TAUOLA of $\tau$ lepton decays– framework for hadronic currents, matrix elements and anomalous decays.

M. Chrzaszcz$^{a,b}$, T. Przedziński$^c$, Z. Was$^b$ and J. Zaremba$^b$.

$^a$ Physik-Institute, University of Zürich, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
$^b$ Institute of Nuclear Physics Polish Academy of Sciences, PL-31342 Krakow, Poland
$^c$ Institute of Physics, Jagellonian University, Lojasiewicza 11, PL-30-348 Krakow, Poland

ABSTRACT

We present an update of the Monte Carlo event generator TAUOLA for $\tau$ lepton decays, with substantially increased list of decay channels and new initialization options. The core of the program remains written in FORTRAN but necessary arrangements have been made to allow handling of the user-provided hadronic currents and matrix elements at the execution time. Such solution may simplify preparation of new hadronic currents and may be useful for fitting to the experimental data as well.

We have implemented as default for TAUOLA a set of hadronic currents, which is compatible with the default initialization used by BaBar collaboration. Options for currents available in previous releases are still stored in the code, sometimes left defunct or activated by internal flags only. The new version of the program, includes also implementation of Lepton Flavour Violating $\tau$ decays.

Finally, we present, as an example, a set of C++ methods for handling user-provided currents, matrix elements or complete new decay channels initialization which can be performed at the program execution time.
1 Introduction

The precision of present day $\tau$ leptons measurements is far better than what theoretical models can offer. We have discussed numerical consequences of this point in Ref. [1]. The design of Monte Carlo generators simulating $\tau$ lepton decays, have to be experiment oriented: the models describing hadronic effects of decays, are expected to be modified frequently. The experiment interfaces parts of the code implementing basic physics principles such as of symmetry and finally structures for data handling in simulation programs and in fitting should remain re-usable, to save time for the users and for easier debugging. Some of the functionalities needed for the aforementioned purposes were already present in TAUOLA but required modification to the code and recompilation. Now, we have adopted program structure, that the introduction of user-friendly methods to swap or add currents via pointers to user-defined functions is possible. Information needed to handle these functions is coded in FORTRAN common blocks\footnote{All FORTRAN common blocks can be accessed from C++ as structs. As such, we will use terms FORTRAN common blocks and C++ structs interchangeably, depending on the context.}. This is a good step toward initialization based on the lists of decay channels, which was started already many years ago [2] for the benefit of BaBar collaboration.

The $\tau$ leptons are the heaviest of all observed leptons. Their decays into hadrons constitute large fraction of all decay channels. For their predictions results from intermediate energy scale QCD are needed. For such a regime theoretical predictions are of limited precision only. We will return to this point later, and let us recall our experience with work on Refs. [3, 4] only. Also because of their large mass, $\tau$ leptons may potentially have large Yukawa couplings to New Particles or fields, and therefore be particularly sensitive to effects beyond the Standard Model (SM). One of the cleanest theoretical way to discover New Physics is via so called Lepton Flavour Violation processes. These phenomena were already observed in the neutral lepton sector via neutrino oscillations [5, 6]. In the charged lepton sector no such processes have been observed so far, despite a large experimental effort [7]. Also, sterile neutrinos seem to escape detection \cite{8, 9}.

The violation of charged lepton flavour (CLFV) is predicted in many extensions of the SM (generically named as BSM theories). The inclusion of the CLFV is usually straightforward and follows directly from the model’s assumptions. Among these theories are: supersymmetry (SUSY) \cite{10}, seesaw
models [11], little Higgs scenarios [12] and models with four generations of fermions [13, 14]. That is also why easy introduction of anomalous $\tau$ decay modes becomes an important step of the program development. Our paper describes how such functionality can be achieved and ported for both: FORTRAN environments such as KKMC [15] and for C++ environments like Tauola Universal Interface [16]. Finally, archivization of $\tau$-decay initialization compatible with the one used by BaBar collaboration was a motivation for the presented work.

Our paper is organized as follows. In Section 2 we recall basic assumptions enabling construction of the program. Necessary approximations are presented. Section 3 documents data structures on which architecture of the whole software relies on. Methods essential for the organization of the program are documented there. Rather minor changes with respect to versions of Refs. [2, 17] were necessary. Program remains essentially in FORTRAN only. Section 4 is devoted to the new default TAUOLA initialization for hadronic currents. Minor comments on the possible variants are nonetheless given. Section 5 describes a physics input for the matrix elements of the non-standard-model type. An example of such anomalous $\tau$ decay mode is presented. Thanks to the new software organization, such decay mode can be now easily introduced into TAUOLA by the user. Summary, Section 6, closes the paper.

Appendices cover technical information related to the source code described in this paper. Appendix A documents an example implementation of the interface for adding new decay channels which is provided with the distribution source code. Despite being a prototype all necessary features we expect the user may want are implemented. Appendix B provides instruction how present update of TAUOLA can be installed as well as examples of the program use. Furthermore, all steps required for installation into C++ environment of Tauola Universal Interface and TauSpinner [16] or into KKMC [15] environment, which remains predominantly in F77, are given. Lastly, Appendix C presents the printout of information for all decay channels provided with the distribution. This includes the branching ratios and flavours of stable decay products for the channels of the default program initialization. This includes also information for non initialized channels: placeholder channels available to be substituted by the user.
2 Physics assumption

Event generation in TAUOLA is built out of several steps that involve different types of theory calculations and thus provide different accuracy of the result. In the leptonic decays all calculations are based on QED and Fermi limit of electroweak interaction, which as we will see, is in general case, sufficiently precise. Therefore, in the following description, we will concentrate only on semileptonic decays. The event generation starts with a phase-space parametrization which is exact and depends on masses of final state particles. Formula [1] used directly is inefficient for a Monte Carlo algorithm if sharp peaks are present due to resonances in the intermediate states. The changes of variables affect the program efficiency but the actual density of the phase space remains intact. In this Section we concentrate on \( \tau \) decays to three scalars and neutrino, but principles described here are true for all other decay modes as well. Phase space, \( d\text{Lips} \) (in the following, notation taken from Ref. [18] is used)

\[
d\text{Lips}(P, q_1, q_2, q_3, q_4) = \frac{1}{2^{17} \pi^8} \int_{Q_{\text{min}}^2}^{Q_{\text{max}}^2} dQ^2 \int_{M_{2,\text{min}}^2}^{M_{2,\text{max}}^2} dM_2^2 
\times \int d\Omega_4 \sqrt{\lambda(M_2^2, Q^2, m_1^2)} \int d\Omega_3 \sqrt{\lambda(Q^2, m_3^2, M_2^2)} \int d\Omega_2 \sqrt{\lambda(M_2^2, m_2^2, m_1^2)} 
Q^2 = (q_1 + q_2 + q_3)^2, \quad M_2^2 = (q_1 + q_2)^2, \quad Q_{\text{min}} = m_1 + m_2 + m_3, \quad Q_{\text{max}} = M - m_4 \quad M_{2,\text{min}} = m_1 + m_2, \quad M_{2,\text{max}} = Q - m_3
\]

is calculated independently of the matrix elements Eq. [2]. Matrix element consists of weak and hadronic currents. Weak current is calculated for Fermi-like interaction,

\[
\tau(P, s) \rightarrow \nu_{\tau}(N)X, \quad \mathcal{M} = \frac{G}{\sqrt{2}} \bar{u}(N)\gamma^\mu(v + a\gamma_5)u(P) \cdot J_\mu.
\]

Hadronic current \( J_\mu \) depends on the momenta of all hadrons. Such separation is valid for precision of about \( \alpha/\pi \approx 0.2\% \). After straightforward calculation we obtain:
$$|\mathcal{M}|^2 = G^2 \frac{v^2 + a^2}{2} (\omega + H_\mu s^\mu),$$

$$\omega = P^\mu (\Pi_\mu - \gamma_{va} \Pi_\mu^5),$$

$$H_\mu = \frac{1}{M} (M^2 \delta^\nu_\mu - P^\nu_\mu P^\mu) (P^{5}_5 - \gamma_{va} P_{\nu}),$$

$$\Pi_\mu = 2 [(J^* \cdot N) J_\mu + (J \cdot N) J^*_\mu - (J^* \cdot J) N_\mu],$$

$$\Pi^5_{\mu} = 2 \text{ Im } \epsilon^{\mu\nu\rho\sigma} J^*_\nu J_{\rho} N_{\sigma},$$

$$\gamma_{va} = - \frac{2va}{v^2 + a^2}$$

If a more general coupling $\nu + a \gamma_5$ for the $\tau$ current and $\nu_{\tau}$ mass $m_\nu \neq 0$ are expected to be used in Eq. 3, the following terms have to be added to $\omega$ and $H_\mu$:

$$\dot{\omega} = 2 \frac{v^2 - a^2}{v^2 + a^2} m_\nu M (J^* \cdot J), \quad \dot{H}_\mu = -2 \frac{v^2 - a^2}{v^2 + a^2} m_\nu \text{ Im } \epsilon^{\mu\nu\rho\sigma} J^*_\nu J_{\rho} P_{\sigma}$$

Nothing has to be changed with respect to ref. [18].

In case of semileptonic decays hadronic current is of the particular interest for the user, as it is the only source of substantial systematic errors. There is no strict way of calculating it, therefore user might be interested in obtaining them from fitting models to the data. In such application a user-friendly interface is crucial. A typical precision of the models is of order of $1/N_c \approx 30\%$ or $1/N_c^2 \approx 10\%$ while experimental data precision is better than 0.1% in most of the cases.

The differential partial width Eq. 5 for the semileptonic channels reads as product of the phase-space, matrix element squared and flux factor.

$$d\Gamma_X = G^2 \frac{v^2 + a^2}{4M} d\text{Lips}(P; q_i, N) (\omega + \dot{\omega} + (H_\mu + \dot{H}_\mu)s^\mu)$$

With new TAUOLA version modification of hadronic currents for the semileptonic decays should pose no problem. User can focus on physics aspects and work e.g. in C++ environment. Anomalous decays are not covered by the

\textsuperscript{2} Although all arrangements were prepared with C++ in mind, user can work in other programming language instead.

4
above scheme. For those, whole matrix element may need to be prepared and ported to TAUOLA. This is no harder than modifications or replacements of the hadronic currents, details will be shown later in this paper.

3 Data structures and wrappers

Until now, many things were hard-coded in FORTRAN routines. Our intention was to make explicit separation into modules: fixed multiplicity phase space generators, hadronic currents, ME calculators, initialization and interface to event records. At the same time we aimed to make changes in the code as small as possible. Our present work is a follow-up on the work started in [2] and continued in [17], which became a technical start for the hadronic current parametrizations used by BaBar collaboration. Our present arrangements profit from the opportunities available in mixed F77/C++ environment.

Our aim was to structurize program in the object-oriented manner but at the same time retain the use of static data structures instead of switching to dynamic memory allocations to preserve performance and minimize memory usage. Such arrangement is a step towards introduction of parallelization methods while still facilitating backward-compatibility checks.

The first step of our work was to prepare FORTRAN common blocks designated to keep all information regarding the generation process. The changes introduced to the code are in FORTRAN, but as a result, each segment can be translated into C++ independently.

We use specialized routines to generate phase-space for final states of fixed multiplicity. This is the reason why the decay channels are grouped accordingly to multiplicity. The constants \( NM_i \), where \( i = 1 \ldots 5 \), denote number of memory slots reserved for channels of final states with multiplicity \( i+1 \). For \( NM_6 \), this convention is overruled. This group collects decay channels with final states of any multiplicity, but for these decay channels an iterative phase space generator is used and as a consequence approximations on matrix element must be imposed. Two additional variables should be mentioned before data structures are explained. The \( NLT=2 \) denotes number of leptonic \( \tau \) decay channels. The \( NMODE \) denotes the total number of memory slots prepared for non-SM or non-leptonic decay channels. It’s value is equal to \( NM_1+NM_2+NM_3+NM_4+NM_5+NM_6 \). In the source code provided with this paper \( NMODE=196 \). If needed, this value can be increased up to \( 500-NLT \).
3.1 Data structures

First, let us present these data structures that, once initialized, usually remain constant during the remainder of the program execution.

```
extern "C" struct taubra_ {
    float GAMPRT[500];
    int JLIST[500];
    int NCHAN;
};
```

**GAMPRT[]**

The branching ratios used to define probabilities with which choice of particular decay channel is made.

**JLIST[]**

The channel number assigned within FORTRAN part of the code. Usually JLIST(i) == i, where i=1,...,500.

```
NCHAN = NMODE + NLT
```

Number of all available decay channels. Parameters NLT, NMODE, NM1, NM2, NM3, NM4, NM5, NM6 are initialized in two places TAUDCDsize.inc and in TauolaStructs.h. It has to remain consistent whenever changes would be introduced.

```
extern "C" struct metyp_ {
    int KEY0[NLT];
    int KEY1[NM1];
    int KEY2[NM2];
    int KEY3[NM3];
    int KEY4[NM4];
    int KEY5[NM5];
    int KEY6[NM6];
};
```

---

3. Note that it is possible to change channel branching probability dynamically during program execution. It is also possible, however undesirable and error-prone, to change the decay channels definition more than once during program execution.

4. It can be set with any non-negative numbers. User should ensure correct proportions for channels. For example, if user generates two decay modes only, one of which is 100-times more likely to occur, it makes no difference for the program if respective entries in GAMPRT are set to 1. and 100. or 0.001 and 0.1.
In this structure information on the type of the matrix elements used by each decay channel is stored. As usual, decay channels are grouped by multiplicity. The following types of matrix element calculation are possible:

0 - channel is not initialized,
1 - constant matrix element (flat phase space),
2 - default TAUOLA matrix element and current,
3 - default TAUOLA matrix element, but for case when one, stable particle of spin>0 in final state is present,
4 - default matrix element but with user defined hadronic current,
5 - user defined matrix element.

extern "C" struct tauccd_ {
    int IDFFIN[NMODE][9];
    int MULPIK[NMODE];
    char NAMES [NMODE][31];
};

NAMES[i][31]
List of names for the decay modes to be used for the output printouts. Up to 31 letters can be used to describe each decay channel.

MULPIK[i]
Multiplicity of stable decay products ($\nu_{\tau}$ excluded). An overall multiplicity for the channel $i$ is thus $MULPIK[i]+1$.

IDFFIN[i][j]
Identifiers of the consecutive stable decay products (except $\nu_{\tau}$). Entries for $j \geq MULPIK[i]+1$ should be set to zero. An exception is for the anomalous decay channels where $\nu_{\tau}$ is absent. In such cases non-zero IDFFIN[i][j] for $j = 3$ or $j = 4$ denotes identifier for the particle to be placed into decay final state instead of $\nu_{\tau}$. Entry for $j = 3$ is to be

---

5 In practice, it is used for $\tau \rightarrow \pi^-\pi^0\gamma$ decay channel only.
6 Note that variable NAMES is defined in FORTRAN as a variable of type CHARACTER*31. As such, it is not a null-terminated C string.
used when identifier is to remain the same for $\tau^-$ and $\tau^+$. Entry for $j = 4$ should be used when the sign should differ.$^7$

extern "C" struct sampl2_ {
    float PROB1[NM2];
    float PROB2[NM2];
    float AM2[NM2];
    float GAM2[NM2];
    float AM3[NM2];
    float GAM3[NM2];
};

extern "C" struct sampl3_ {
    float PROB1[NM3];
    float PROB2[NM3];
    float AMRX[NM3];
    float GAMRX[NM3];
    float AMRA[NM3];
    float GAMRA[NM3];
    float AMRB[NM3];
    float GAMRB[NM3];
};

extern "C" struct sampl4_ {
    float PROB1[NM4];
    float PROB2[NM4];
    float AMRX[NM4];
    float GAMRX[NM4];
    float AMRA[NM4];
    float GAMRA[NM4];
};

extern "C" struct sampl5_ {
    float PROBa2[NM5];
    float PROBOM[NM5];

$^7$Note that this arrangement implies, that anomalous, neutrinoless channels can be initialized only for 3 or 2 particles final states. We think it is sufficient for foreseeable future.
float ama2[NM5];
float gama2[NM5];
float AMOM[NM5];
float GAMOM[NM5];

The structures collect parameters for the presamplers and channels of multiplicity corresponding respectively from 2 to 5 hadrons in the final state. Those parameters are probabilities for generation sub-channels or coefficients, such as masses and widths of intermediate peaks. Modification of a presampler affects efficiency of generation only. It may be necessary if substantially modified hadronic current is used, for example, if user plans to check the code in intermediate resonances narrow width limit.

3.2 Wrappers

Let us document functions that have to be written by the user when an interface to the new software structure is introduced. These wrappers are used to pass parameters from FORTRAN part of the code to the user-provided functions. That is why the format of these functions must be strictly followed. Each function name consist of name of the current (or matrix element) as used in older pure FORTRAN versions of TAUOLA but with _wrap added at the end. For example, wrapper for user routine to replace DAM2PI function is declared as:

extern "C" void dam2pi_wrap_
    (int *MNUM, float *PT, float *PN, float *PIM1, float *PIM2,
     float *AMPLIT, float *HV)

These wrappers depend on multiplicity of final states but in most cases are prepared accordingly to the same scheme. The exceptions are listed at the end of this Section.

8An implementation of these functions have been presented in tauola-c/channel_wrappers.c file. This implementation demonstrates how parameters of the functions for calculation of matrix element or hadronic currents should be used. If the C++ interface is to be removed, e.g. as the first step of writing another one, possibly to different language, use of code of tauola-no-c/ directory, instead of tauola-c may be useful.
Note that such wrappers are executed only if user have registered a custom channel (with matrix element or hadronic current, i.e. $\text{KEY2}[\text{mnum}]=4 \text{ or } 5$) for particular multiplicity and particular $\text{mnum}$ ($\text{mnum}$ denotes position on the sub-list of decays corresponding to the multiplicity).

The FORTRAN common blocs are not sufficient to describe initialization of TAUOLA. The table, like Tauolapp::tauabra_userChannels[channel] of our interface, described in Appendix A is needed. Let us recall essential points only, leaving description of the Interface to the Appendix. The function RegisterChannel of this interface can be used. The position on the list of all channels is calculated from the position of channels with two scalars in final state $\text{MNUM}$ as $\text{NM4}+\text{NM5}+\text{NM6}+\text{NM3}+\text{mnum}$. Then, a pointer to the user routine is taken from the table:

$$\text{Tauolapp::tauabra_userChannels[channel]}.$$  
This pointer is set to user routine executed in place of function:

$$\text{xsec(pt, pn, pim1, pim2, amplit, hv)}$$

If for some reason this pointer is not set, program will exit with an error:

ERROR: dam2pi_wrap: pointer to xsec for channel 3 (mnum 3) not set!''

The remaining part of each function-wrapper is devoted to debugging. It will activate if any of the calculated quantities, matrix element squared or polarimetric vector is of a NaN value.

The following subsubsections list all wrappers used in the project. We use the C/C++ declaration to describe these wrappers as it is less intuitive than the original FORTRAN declaration, which can be deducted much easier.

### 3.2.1 Wrappers for squared Matrix Elements

**extern "C" void dam1pi_wrap**

(int *MNUM, float *PNU, float *AMF0, float *PKK, float *AMF1, float *GAMM, float *HV)

**extern "C" void dam2pi_wrap**

(int *MNUM, float *PT, float *PN, float *PIM1, float *PIM2, float *AMPLIT, float *HV)

**extern "C" void dam3pi_wrap**

(int *MNUM, float *PT, float *PN, float *PIM1, float *PIM2,
float *PIM3, float *AMPLIT, float *HV)

extern "C" void dam4pi_wrap_
    (int *MNUM, float *PT, float *PN, float *PIM1, float *PIM2,
     float *PIM3, float *PIM4, float *AMPLIT, float *HV)

extern "C" void dam5pi_wrap_
    (int *MNUM, float *PT, float *PN, float *PIM1, float *PIM2,
     float *PIM3, float *PIM4, float *PIM5, float *AMPLIT, float *HV)

3.2.2 Wrappers for Hadronic Currents

extern "C" void curr2_wrap_
    (int *MNUM, float *PIM1, float *PIM2, complex *HADCUR)

extern "C" void curr3pi_wrap_
    (int *MNUM, float *PIM1, float *PIM2, float *PIM3, complex *HADCUR)

extern "C" void curr4_wrap_
    (int *MNUM, float *PIM1, float *PIM2, float *PIM3, float *PIM4,
     complex *HADCUR)

extern "C" void curr5_wrap_
    (int *MNUM, float *PIM1, float *PIM2, float *PIM3, float *PIM4,
     float *PIM5, complex *HADCUR)

3.2.3 Special cases

extern "C" void dampry_wrap_
    (int *ITDKRC, double *XKODEC, double *XK, double *XA, double
     *QP, double *XN, double *AMPLIT, double *HV)
Matrix element calculation for leptonic decays, including possibly 4-momentum for bremsstrahlung photon and its phase space limit $\textbf{XKODEC}$.

```c
extern "C" float sigee_wrap_(float *AMX2, int *MNUM)
```

In some decays approximation is used. Only distribution of invariant mass for all hadrons is generated accordingly to function $\text{sigee}$ as if it was vector state. The final states of hadronic system fragmentation are generated accordingly to flat phase space.

```c
extern "C" float fconst_wrap_(int *MNUM)
```

In this case no matrix element is used at all. Only overall constant is passed to generation. Flat phase space (all mass effects included) distribution is generated.

### 3.3 Generation monitoring variables

All structures presented so far, were for variables initialized at the very beginning of initialization and left unchanged over the whole generation run. Wrappers are not expected to store information on the whole sample as well. This Subsection presents the variables used to access information which is continuously updated during the program run. Information on generation weights is stored in local variables of routine $\text{DADNEW}$ for each semileptonic decay separately, and in routines $\text{DADMEL}$ and $\text{DADMMU}$ for leptonic decays:

- `REAL*4 WTMAX(NMODE)`
- `REAL*8 SWT(NMODE),SSWT(NMODE)`
- `INTEGER*8 NEVRAW(NMODE),NEVOVR(NMODE),NEVACC(NMODE)`

The variables denote:

- **WTMAX** - maximum weight for the given channel,
- **SWT** - sum of all event weights for the given channel,
- **SSWT** - sum of all event weights squared for the given channel,
- **NEVRAW** - number of raw generated events for the given channel,
- **NEVOVR** - number of even of weights larger than the maximum estimated at the initialization for the given channel,
NEVACC - number of accepted events for the given channel.

Further variables regarding the program run are stored in common block TAUBMC:

```fortran
COMMON / TAUBMC / GAMPMC(500), GAMPER(500), NEVDEC(500)
```

The variables denote:

- **GAMPMC(500)** - width for the given decay channel calculated from the Monte Carlo generation,
- **GAMPER(500)** - error of the **GAMPMC**,
- **NEVDEC(500)** - number of generated decays.

Parts of the code and variables of the present Subsection will have to be adjusted at later steps of the program evolution, but as these parts do not affect the use of the presented methods for matrix element replacement we left them in **FORTRAN**.

This setup has to be modified if a multi-threaded parallelization process is to be used. At this moment it does not seem to be necessary, because with the present day matrix elements **TAUOLA** is sufficiently fast. Furthermore, applying parallelization solution as used by KK Monte Carlo is straightforward. It would only require to store run information, described in the present Subsection, into histograms.

### 4 Default **TAUOLA** initialization

The main purpose of the present paper is description of the means for user alterations of hadronic currents to be used in $\tau$ decays. However, these modifications have to base on the default initialization of **TAUOLA**. In this Section we describe such default initialization available for the user prior to his main work.

For present distribution of **TAUOLA** we decided that the initialization compatible with the one used by BaBar collaboration, from now on will be used as default. It contains currents which even with very large statistics reproduce results of BaBar collaboration version of **TAUOLA**. This is the version used by
collaboration for their default simulations. We have compared results of our new initialization of TAUOLA with BaBar collaboration production files. We have reproduced BaBar setup with a great detail to the best of our knowledge. However, we still have missed fine-tuning of minimum photon energy for PHOTOS and TAUOLA for generation of bremsstrahlung in decays. Therefore we did not have reproduced exactly results of BaBar production files. However, the differences were minuscule. Agreement was checked with MC-TESTER with samples of 1,617,945,000 $\tau$ decays. The MC-TESTER-based tests accounted for over 133 decay channels (including those with multiple photons in final state generated by PHOTOS). Dissimilarity coefficients calculated by MC-TESTER (see Ref. Section 6.1) were: $T_1=0.033235\%$, $T_2=0.058574\%$. Further details on the test and how to reproduce its results are given in Appendix B.2.

4.1 Extensions

In principle, there are several extensions that can be applied to this initialization. The distribution tarball includes e.g. RChL currents for 3 pions, Novosibirsk currents for 4 pions, and 5 pion currents described in [24]. Those additional currents are of secondary importance so we will not describe them here. Information on usage is given in respective publications, README files and comments in the code.

We would like to point out that this distribution does not provide CLEO initialization of TAUOLA used since Ref. [2], even though the code for this parametrization is not removed but left inactive. That is the reason why some comments in the code are outdated. We feel that since the present version is prepared to start completely new initialization, archiving BaBar initialization and setting it as the default brings more benefit than retaining CLEO and/or other initializations used so far. If the need arises user may adapt the existing initialization to match CLEO setup at his own discretion. To decide whether to remove corresponding code or make it available for the user will rely on discussions with community of users.

---

9 We have no detailed knowledge when and which other currents parametrizations were used for specific measurements/studies. Nonetheless we think that this version deserves archivization.
5 Example of the user-defined hadronic current

In this Section we present an example of a user-defined matrix element, which we exploit to extend the default setup of TAUOLA described in previous Section. For this purpose, anomalous channels are good candidates because of large interest from the users. Moreover, these channels are not as challenging for fitting as the other ones. Their analytic form is simple. Present Section describes the physics aspects of these matrix elements. The distribution tar-ball contains a demo-lfv directory demonstrating how such channels can be added into TAUOLA using the new setup. A practical example is given in Appendix B.

5.1 Effective field theory approach as applied to $\tau \to \mu\mu\mu$ decay

CLFV processes resulting from BSM theories can be described in a model-independent way in terms of operators. If new physics exist at a mass scale $\Lambda$, it can manifest itself at an electroweak scale in the form of a higher order operators which, however, do not spoil the $SU(2)_L \times U(1)_Y$ symmetry. In the EFT approach the right-handed singlets are written as the following isospin doublets [25]:

$$R_e = \frac{1 - \gamma_5}{2} \begin{pmatrix} 0 \\ \psi_e \end{pmatrix}, \quad R_\mu = \frac{1 - \gamma_5}{2} \begin{pmatrix} 0 \\ \psi_\mu \end{pmatrix}, \quad R_\tau = \frac{1 - \gamma_5}{2} \begin{pmatrix} 0 \\ \psi_\tau \end{pmatrix}. \quad (6)$$

Taking into account Eq. (6) and the matrix of Higgs fields from [25], one can derive the following relevant dimension six operators:

$$O_1 = (\bar{L}\gamma_\mu L)(\bar{L}\gamma^\mu L), \quad (7)$$
$$O_2 = (\bar{L}\tau^a_\mu\gamma_\mu L)(\bar{L}\tau^a_\gamma\gamma^\mu L), \quad (8)$$
$$O_3 = (\bar{R}\gamma_\mu R)(\bar{R}\gamma^\mu R), \quad (9)$$
$$O_4 = (\bar{R}\gamma_\mu R)(\bar{R}\gamma^\mu L), \quad (10)$$
$$R_1 = g' (\bar{L}H\sigma_{\mu\nu}R)B^{\mu\nu}, \quad (11)$$
$$R_2 = g (\bar{L}\tau^a H\sigma_{\mu\nu}R)W^{\mu\nu}. \quad (12)$$
As defined above, $B_{\mu\nu}$ and $W_{\mu\nu,a}$ are the electroweak gauge fields, $g$ and $g'$ are the coupling constants of $SU(2)_L$ and $U(1)_Y$, $H$ denotes the matrix of Higgs fields, $L(R)$ are the left(right)-handed fields and $\sigma^{\mu,\nu} = \frac{i}{4} [\gamma^\mu, \gamma^\nu]$.

According to S. Turszyk et. al. [26], higher order operators are suppressed by small lepton Yukawa couplings. We will not consider them in this paper.

In the effective field theory the most general Hamiltonian that describes the discussed process is formed as the sum of the operators 7 to 12. For the process $\tau \to \mu\mu\mu$ the operators $O_1$ and $O_2$ are identical after projecting them on charged leptons. The $O_3$ corresponds to a purely right-handed current and is completely analogous to $O_1$. The $O_4$ operator corresponds to the mix four fermion operator. For radiative operators $R_1$ and $R_2$, the latter is suppressed by small Yukawa coupling of $\tau$. Only photonic operator $R_1$ is relevant.

The operators of the present paper were also interpreted in terms of the BSM operators. The respective decay widths can be presented in the form of Dalitz distributions [27], which were derived in the following five cases, corresponding to different lepton chirality structures:

- **Four left-handed leptons ($O_1$ operator):**

$$
\frac{d^2\Gamma^{(LL)(LL)}_V}{dm^2_{23}dm^2_{12}} = \frac{\left| g^{(L_\mu R)}_V \right|^2}{\Lambda^4} \frac{(m_\tau^2 - m_\mu^2)^2 - (2m_{12}^2 - m_\tau^2 - 3m_\mu^2)^2}{256\pi^3 m_\tau^3}. \tag{13}
$$

- **Two left-handed, two right-handed leptons ($O_4$ operator):**

$$
\frac{d^2\Gamma^{(LL)(RR)}_V}{dm^2_{23}dm^2_{12}} = \frac{\left| g^{(L_\mu R')}_V \right|^2}{\Lambda^4} \left[ \frac{(m_\tau^2 - m_\mu^2)^2 - 4m_\mu^2(m_\tau^2 + m_\mu^2 - m_{12}^2)}{512\pi^3 m_\tau^2} - \frac{(2m_{13}^2 - m_\tau^2 - 3m_\mu^2)^2 + (2m_{23}^2 - m_\tau^2 - 3m_\mu^2)^2}{1024\pi^3 m_\tau^2} \right]. \tag{14}
$$

- **Radiative right-handed $\tau$ leptons ($R_1$ operator):**

$$
\frac{d^2\Gamma^{(LR)}_{rad}}{dm^2_{23}dm^2_{12}} = \alpha_{em}^2 \frac{\left| g^{(L_\mu R')}_{rad} \right|^2 \nu^2}{\Lambda^4} \left[ \frac{m_\mu^2(m_\tau^2 - m_\mu^2)^2}{128\pi^3 m_\tau^2} \left( \frac{1}{m_{13}^4} + \frac{1}{m_{13}^4 + m_{23}^4} \right) + \frac{m_\mu^2(m_\tau^2 - 3m_\mu^2)m_\mu^2 + 2m_\mu^2}{128\pi^3 m_\tau^2 m_{23}^2 m_{13}^2} + \frac{2m_{12}^2 - 3m_\mu^2}{128\pi^3 m_\tau^2} + \frac{(m_{13}^2 + m_{23}^2)(m_{12}^4 + m_{13}^4 + m_{23}^4 - 6m_\mu^2(m_\mu^2 + m_\tau^2))}{256\pi^3 m_\tau^2 m_{23}^2 m_{13}^2} \right]. \tag{15}
$$
• Interference between $O_1$ and $R_1$:

$$\frac{d^2\Gamma^{(LL)(RR)}_{\text{mix}}}{dm_{12}^2 dm_{23}^2} = \alpha_\text{em}^2 \frac{2\nu \text{Re} \left[ g_V^{(L\mu L^\tau)(L\mu L^\nu)} g_{\nu \text{rad}}^{L\mu R^\tau} \right]}{\Lambda^4} \left[ \frac{m_{12}^2 - 3m_{12}^2}{64\pi^3m_{\tau}^2} + \frac{m_{\mu}^2(m_\tau^2 - m_{\mu}^2)^2(m_{13}^2 + m_{23}^2)}{128\pi^3m_{23}^2m_{13}^2} \right].$$  \hspace{1cm} (16)

• Interference between $O_4$ and $R_1$:

$$\frac{d^2\Gamma^{(LL)(RR)}_{\text{rad}}}{dm_{12}^2 dm_{23}^2} = \alpha_\text{em} \frac{2\nu \text{Re} \left[ g_V^{(L\mu L^\tau)(R\mu R^\nu)} g_{\nu \text{rad}}^{L\mu R^\tau} \right]}{\Lambda^4} \left[ \frac{m_{\tau}^2 - m_{12}^2 - 3m_{\mu}^2}{256\pi^3m_{\tau}^2} + \frac{m_{\mu}^2(m_\tau^2 - m_{\mu}^2)^2(m_{13}^2 + m_{23}^2)}{256\pi^3m_{23}^2m_{13}^2} \right].$$  \hspace{1cm} (17)

In distributions 13 - 17 the following dimuon masses are defined:

$$m_{--} = m_{12}^2 = (p_{\mu}^- + p_{\mu}'^+)^2, \quad m_{+-} = m_{23}^2 = (p_{\mu}^+ + p_{\mu}'^-)^2, \quad m_{13}^2 = m_{\tau}^2 + 3m_{\mu}^2 - m_{--}^2 - m_{+-}^2.$$

(18)

and $m_\ell$ is the mass of corresponding lepton, $g_V$ is the corresponding coupling constant and $\nu$ is the element from the Higgs matrix. The Dalitz distributions can be found in Fig. 1, we were able to reproduce them with the help of TAUOLA Monte Carlo.

Note that in contrary to phase-space parametrization of Eq. 1, in case of Dalitz parametrization, $dm_{23}^2 dm_{12}^2$, no phase space, kinematic-dependent, contribution to Jacobians appear. That is why, right hand sides of Eqs. 13 to 17 are proportional, up to numerical overall constant to matrix element squared and averaged/summed over the spin. This is true, because no dependence on angular variables is present in these formulae. We have implemented these formulae into our program, neglecting sometimes terms of order of $\sim \frac{m_{\mu}^2}{m_\tau^2} < 0.01$. 

17
Figure 1: Dalitz distributions simulated in the effective field approach for the five different BSM operators corresponding to different lepton chirality structures [25]. The normalized to unit area distributions, implemented in the TAUOLA package.
6 Summary

The purpose of the present paper was to present an installation package for essentially **FORTRAN** update of $\tau$ leptons decay library **TAUOLA**. The package is ready to be merged with Tauola Universal Interface [16] and KKMC [15]; necessary patches are provided. The distribution corresponds nearly in all details to the version presented at TAU14 conference [28]; minor modifications are explicitly listed.

In the code distributed with the present paper, maximum number of $\tau$ decay channels is 500, most of them are left not-initialized and represent reserved static memory only. Changes introduced into **FORTRAN** part of **TAUOLA** have been described in a great detail. All initialization information has been moved to **FORTRAN** common blocks so that they can be accessed from other parts of the code, for example as C structs.

The default initialization of the package provides hadronic currents and corresponding branching fractions numerically equivalent to the ones used by the BaBar collaboration for their default simulations. Extended set of decay channels with anomalous nature was also prepared. For the anomalous decay channels of 2 and 3 decay products $\nu_\tau$-less channels can be introduced.

The backward compatibility to CLEO initialization was not assured, but it is rather easy to recover if the need would arise. The corresponding **FORTRAN** code is not removed yet. This distribution completes the first step towards the migration of the whole project to **C++** and towards preparation of convenient structure for the parallelization. Further steps will require collaboration with the program users, in particular with the authors of fitting programs of the future experiments. Also, at this moment, we can not decide if the **FORTRAN** code for the variants of hadronic current parametrization can be removed.

Acknowledgments

This project was partially supported by the funds of Polish National Science Center under decision UMO-2014/15/B/ST2/00049.

M. Chrzaszcz is grateful for support of the Polish National Science Center under the "Sonata" grant: UMO-2015/17/D/ST2/03532.
References

[1] Zbigniew Was and Jakub Zaremba. Study of variants for Monte Carlo generators of $\tau \rightarrow 3\pi\nu$ decays. Eur. Phys. J., C75(11):566, 2015. [Erratum: Eur. Phys. J.C76,no.8,465(2016)].

[2] Z. Was. TAUOLA the library for tau lepton decay, and KKMC / KORALB / KORALZ /... status report. Nucl. Phys. Proc. Suppl., 98:96–102, 2001. [,96(2000)].

[3] O. Shekhovtsova, T. Przedzinski, P. Roig, and Z. Was. Resonance chiral Lagrangian currents and $\tau$ decay Monte Carlo. Phys. Rev., D86:113008, 2012.

[4] I. M. Nugent, T. Przedzinski, P. Roig, O. Shekhovtsova, and Z. Was. Resonance chiral Lagrangian currents and experimental data for $\tau^- \rightarrow \pi^-\pi^+\nu_\tau$. Phys. Rev., D88:093012, 2013.

[5] Y. Fukuda et al. Evidence for oscillation of atmospheric neutrinos. Phys. Rev. Lett., 81:1562–1567, 1998.

[6] Q. R. Ahmad et al. Direct evidence for neutrino flavor transformation from neutral current interactions in the Sudbury Neutrino Observatory. Phys. Rev. Lett., 89:011301, 2002.

[7] Y. Amhis et al. Averages of $b$-hadron, $c$-hadron, and $\tau$-lepton properties as of summer 2014. 2014.

[8] S. Fukuda et al. Tau neutrinos favored over sterile neutrinos in atmospheric muon-neutrino oscillations. Phys. Rev. Lett., 85:3999–4003, 2000.

[9] Roel Aaij et al. Search for Majorana neutrinos in $B^- \rightarrow \pi^+\mu^-\mu^-$ decays. Phys. Rev. Lett., 112(13):131802, 2014.

[10] Ian Aitchison. Supersymmetry and the MSSM: An Elementary introduction. 2005.

[11] Alejandra Melfo, Miha Nemevsek, Fabrizio Nesti, Goran Senjanovic, and Yue Zhang. Type II Seesaw at LHC: The Roadmap. Phys. Rev., D85:055018, 2012.
[12] Martin Schmaltz. Physics beyond the standard model (theory): Introducing the little Higgs. *Nucl. Phys. Proc. Suppl.*, 117:40–49, 2003. [40(2002)].

[13] Andrzej Buras et al. Lepton Flavour Violation in the Presence of a Fourth Generation of Quarks and Leptons. *JHEP*, 1009:104, 2010.

[14] Linda Carpenter, Arvind Rajaraman, and Daniel Whiteson. Searches for Fourth Generation Charged Leptons. 2010.

[15] S. Jadach, B. F. L. Ward, and Z. Was. The Precision Monte Carlo event generator K K for two fermion final states in e+ e- collisions. *Comput. Phys. Commun.*, 130:260–325, 2000.

[16] N. Davidson, G. Nanava, T. Przedziński, E. Richter-Was, and Z. Was. Universal Interface of TAUOLA Technical and Physics Documentation. *Comput. Phys. Commun.*, 183:821–843, 2012.

[17] Z. Was and P. Golonka. TAUOLA as tau Monte Carlo for future applications. *Nucl. Phys. Proc. Suppl.*, 144:88–94, 2005. [88(2004)].

[18] S. Jadach, Z. Was, R. Decker, and Johann H. Kuhn. The tau decay library TAUOLA: Version 2.4. *Comput. Phys. Commun.*, 76:361–380, 1993.

[19] Elisabetta Barberio and Zbigniew Was. PHOTOS: A Universal Monte Carlo for QED radiative corrections. Version 2.0. *Comput. Phys. Commun.*, 79:291–308, 1994.

[20] Piotr Golonka and Zbigniew Was. PHOTOS Monte Carlo: A Precision tool for QED corrections in Z and W decays. *Eur. Phys. J.*, C45:97–107, 2006.

[21] N. Davidson, P. Golonka, T. Przedzinski, and Z. Was. MC-TESTER v. 1.23: A Universal tool for comparisons of Monte Carlo predictions for particle decays in high energy physics. *Comput. Phys. Commun.*, 182:779–789, 2011.

[22] P. Golonka, T. Pierzchała, and Z. Wąs. Mc-tester: a universal tool for comparisons of monte carlo predictions for particle decays in high energy physics. *Computer Physics Communications*, 157(1):39 – 62, 2004.
A. E. Bondar, S. I. Eidelman, A. I. Milstein, T. Pierzchala, N. I. Root, Z. Was, and M. Worek. Novosibirsk hadronic currents for tau \(\rightarrow\) 4 pi channelsof tau decay library TAUOLA. *Comput. Phys. Commun.*, 146:139–153, 2002.

[24] Johann H. Kuhn and Z. Was. tau decays to five mesons in TAUOLA. *Acta Phys. Polon.*, B39:147–158, 2008.

[25] Sascha Turczyk. Model-independent analysis of \(\tau \rightarrow \ell\ell\ell'\) decays. *Nucl. Phys. Proc. Suppl.*, 189:140–145, 2009.

[26] B.M. Dassinger, Th. Feldmann, Th. Mannel, and S. Turczyk. Model-independent analysis of lepton flavour violating tau decays. *JHEP*, 0710:039, 2007.

[27] R. H. Dalitz. Decay of \(\tau\) mesons of known charge. *Phys. Rev.*, 94:1046–1051, May 1954.

[28] Zbigniew Was. The \(\tau\) leptons theory and experimental data: Monte Carlo, fits, software and systematic errors. *Nucl. Part. Phys. Proc.*, 260:47–51, 2015.

[29] Twiki webpage for kkmc version for fcc. [https://twiki.cern.ch/twiki/bin/view/FCC/Kkmc](https://twiki.cern.ch/twiki/bin/view/FCC/Kkmc). Accesed: 2017-04-19.

[30] Twiki webpage for tauola version for fcc. [https://twiki.cern.ch/twiki/bin/view/FCC/Tauola](https://twiki.cern.ch/twiki/bin/view/FCC/Tauola). Accesed: 2017-05-05.

### A Example of the interface implementation

The main goal of new TAUOLA version was to prepare FORTRAN common blocks to facilitate addition of new currents and modification of existing ones\[^{10}\] in cases when user main program is in FORTRAN or in C++. New setup allows for initialization or re-initialization of the currents definition at runtime. It also makes it easier to understand in principle modular structure of the algorithms used in the project.

\[^{10}\]In principle, there is no need for any interface to modify hadronic currents used in TAUOLA. Nonetheless it may be convenient for some users.
Before describing the implementation of the interface provided with the source code we want to point out that most of the arrangements presented in this Section can be modified or replaced. All of presented methods, have to rewrite information stored in the data structures described in Section 3. We have prepared solutions on the basis of discussion with users, but alternative solutions may be found to be more convenient. The code presented in this Section may thus be replaced rather easily.

The C++ interface for adding and modifying decay channels in TAUOLA is defined in `tauola-c/ChannelForTauolaInterface.c`. Most of the functions provided by this interface work on the instances of the class `ChannelForTauola` (defined in `tauola-c/ChannelForTauola.h`). Functions invoked from the FORTRAN code are defined in `tauola-c/channels_wrappers.c`. These functions are used by FORTRAN code of TAUOLA to access `ChannelForTauola` objects and in this way pass parameters further to user-initialized channels. In particular, to user routines for matrix elements or hadronic current calculation, accessed at the execution time through pointers. This approach allows to program in any language as long as proper wrappers are provided. Scheme of calling functions written in other languages from FORTRAN depends on a particular programming language and a set of compilers used to combine these languages.

Initialization of user defined code is performed in two steps. First, one has to call routine `Tauolapp::setUserRedefinitions(RedefExample);`\[11\]. Then `iniofc` routine (defined in `tauola-c/channels_wrappers.c`) has to be called to reinitialize TAUOLA with new user-updated currents.

Note that useful information can be found also in comments of the header files located in `tauola-c` subdirectory and in the example programs provided with the distribution. Most notably, an extensive example of use of the interface described in this Appendix is available in `demo-redefine` subdirectory. File `demo-redefine/iniofc.c` contains verbose comments about available functions, their effects and possible uses. All FORTRAN common blocks used by the interface are declared as external C structures in file `tauola-c/TauolaStructs.h`. Note that in this file parameters NLT, NMODE, NM1, NM2, NM3, NM4, NM5 and NM6 are defined. In order for the interface to work correctly, these definitions must coincide with the definitions located in `TAUDCDSIZE.inc`.

\[11\]In example `demo-redefine` it has been shown how this routine can be called from FORTRAN via subroutine `TAUOLAREDEF` defined in `demo-redefine/iniofc.c`
A.1 Functions defined in ChannelForTauolaInterface.h

Let us describe functions presented in tauola-c/ChannelForTauolaInterface.h which form an example of such interface. Note that, as pointed before, this solution should serve as a prototype and can be easily replaced if the user wishes to introduce a new one. Especially if some of the functionality of this prototype is not needed. For example, significant part of the code in this implementation is devoted to preventing the coding errors when redefining or adding new channels through strict differentiation between channels of different multiplicity. Strict type checks are applied on such channels and function pointers passed to these channels. Because of that, our solution may be hard to read and it might hinder understanding the main purpose of each function.

On the other hand, understanding the interface details may not be necessary to perform basic operations and the following explanations should be enough to make a good use of this interface.

extern ChannelForTauola* taubra_userChannels[500];
Holds pointers to user defined ChannelForTauola objects. It is a supplement to struct taubra_ defined in tauola-c/TauolaStructs.h.

extern void* leptonChannelsMEpointers[2];
Pointers to matrix elements for leptonic channels.

extern void (*channelRedefinitionFunction)();
Pointer to tauola channel redefinition function.

void PrintChannelInfo(int channel);
Prints current definition of a channel: channel number, branching ratio, multiplicity of hadrons in final state, sub-channel number, matrix element type, and identifiers of final state particles.

void SetUserRedefinitions(void (*function)());
Sets pointer to function reinitializing Tauola channels. This function is the only place where channels can be reinitialized.

int RegisterChannel(int channel, ChannelForTauola *pointer);
Changes information about selected channel or adds a new channel. Parameter channel defines place on the list of channels where new one is to be added. Setting it to -1 means that new channel has to be
added on the first empty slot. Parameter pointer is a pointer to an object defining new or modified channel. Function returns added or modified channel number or 0 in case of failure. If function succeeds, it takes ownership of the object pointed by the pointer. Same channel number cannot be registered twice. Use OverwriteChannel instead.

```c
int OverwriteChannel(int channel, ChannelForTauola *pointer);
```
Overwrites information about selected channel with new information provided by the pointer. Parameter channel defines place on the list of channels where new one is to be added. Parameter pointer is a pointer to an object defining new or modified channel. If function succeeds, it takes ownership of the object pointed by the pointer. Function returns added or modified channel number or 0 in case of failure.

```c
int ModifyLeptonic(int channel, int me, float br, void *xsec = NULL);
```
Changes information about selected leptonic channel. This functionality is by far more delicate than that of function RegisterChannel, and user is requested to perform detailed tests of its performance. Parameter channel can be chosen only as 1 or 2. Parameter me is a matrix element type. See README for the meaning. Parameter br is a branching ratio of the selected channel. Parameter current is a pointer to function calculating the cross-section or current. Pointer xsec has to be of type DAMPRY_POINTER_TYPE defined in tauola-c/ChannelForTauola.h. Function returns modified channel number or 0 in case of failure.

```c
ChannelForTauola* GetChannel(int channel);
```
Get channel information from FORTRAN common block. Returns a pointer to ChannelForTauola object with channel information filled, ready to register without any changes or to update and register. Such ChannelForTauola object can be only registered back on their original place, unless matrix element calculation is reduced to constant (flat phase-space), then it can be registered on empty slot.

```c
int SetPresampler2(ChannelForTauola *pointer, float prob1, float prob2, float am2, float gam2, float am3, float gam3);
```
Set parameters used to optimize efficiency of 2-scalar mode phase space generator. First argument is a pointer for ChannelForTauola for which presampler parameters will be changed. All of the following arguments
are new numerical values of presampler parameters. Note: 0<=prob1; 0<=prob2; prob1+prob2<=1.

int SetPresampler3(ChannelForTauola *pointer, float prob1, float prob2, float amrx, float gamrx, float amra, float gamra, float amrb, float gamrb);
Set parameters used to optimize efficiency of 3-scalar mode phase space generator. First argument is a pointer for ChannelForTauola for which presampler parameters will be changed. All of the following arguments are new numerical values of presampler parameters. Note: 0<=prob1; 0<=prob2; prob1+prob2<=1.

int SetPresampler4(ChannelForTauola *pointer, float prob1, float prob2, float amrx, float gamrx, float amra, float gamra);
Set parameters used to optimize efficiency of 4-scalar mode phase space generator. First argument is a pointer for ChannelForTauola for which presampler parameters will be changed. All of the following arguments are new numerical values of presampler parameters. Note: 0<=prob1; 0<=prob2; prob1+prob2<=1; only prob1+prob2 is used.

int SetPresampler5(ChannelForTauola *pointer, float proba2, float probom, float ama2, float gama2, float amom, float gamom);
Set parameters used to optimize efficiency of 5-scalar mode phase space generator. First argument is a pointer for ChannelForTauola for which presampler parameters will be changed. All of the following arguments are new numerical values of presampler parameters. Note 0<=proba2<=1; 0<=probom<=1.

A.2 Key functionality of the ChannelForTauola class

Constructors
Class ChannelForTauola provides eleven specialized constructors, one for each type of the user functions that can be passed to TAUOLA\textsuperscript{12}. Each of these constructors accept the following parameters:

float br - branching ratio

\textsuperscript{12}Matrix elements for channels of multiplicity from 1 to 5, hadronic currents for channels of multiplicity from 2 to 5, function for a leptonic current and a constant function.
const vector<int> &ki - list of the ID of the decay products
string name - name of the decay channel
* POINTER_TYPE pointer - pointer to user function

The possible POINTER_TYPES are listed at the top of the tauola-c/ChannelForTauola.h file. Initialization of parameters mulpik and me_type is done accordingly to this pointer type. For channels that are constructed without a pointer to function, a separate constructor has been provided that allows setting mulpik and me_type manually.

Accessors
Get and set methods have been provided for all of the parameters of this class. The most notable are the accessors for me_type. The me_type can be set to following values:

0 - channel is not initialized,
1 - channel is reduced to constant matrix element, i.e. flat phase space is used,
2 - default matrix element is used,
3 - default matrix element is used but one stable particle is of spin>0,
4 - default matrix element is used with user hadronic current function,
5 - user-provided function is used for matrix element calculation.

Note that setter for this parameter can only set it to values 0,1,2 or 3. Values 4 and 5 are reserved for the class constructors and are set based on the function pointer provided to the constructor.

ChannelForTauola::print()
Prints information about this channel.

B Installation and examples

The installation procedure of the tauola-bbb is similar to the one of original tauola. All relevant source code is located in TAUOLA-FORTRAN/tauola-bbb directory. Executing make in this directory builds all relevant sub-modules and produces the glib.a library. Executing make in any demo subdirectory

\[13\] Make sure that HEPEVT definition located in TAUOLA-FORTRAN/include/HEPEVT.h is exactly the same as used in target environment.
builds corresponding example. Running ./go script in prod subdirectory of any demo directory runs the example with the default setup provided with the example.

There are three demo subdirectories prepared:

- demo-babar
- demo-lfv
- demo-redefine

These examples are basically copies of each other with minor differences only. We decided to split examples into three folders, for user convenience to grasp properties of new version.

We also provide folders:

- patch-tauolapp
- patch-KK-face
- patch-babar-validation

These Patch-folders include necessary information on how to use new tarball in other projects. The last one is with instruction how to reproduce results of validation for the new initialization.

In the following Subsections we describe briefly the content of the directories.

### B.1 Default BaBar initialization example

Directory demo-babar is to demonstrate the default initialization of tauola-bbb. This example, produces results validated against BaBar KKMC with BaBar TAUOLA. It is the simplest example provided with the distribution. It does not use C++ interface for adding channels. This example is to show how the default TAUOLA output looks like.

### B.2 BaBar initialization used for validation against BaBar data

Directory patch-babar-validation contains README with a list of modifications that have to be applied to the tauola-bbb in order to allow comparison against BaBar data.
File \texttt{babar.root} contains an MC-TESTER analysis of the 1600 MEvents sample taken from the production files of BaBar experiment. File \texttt{tauola-bbb.root} contains an analysis of the sample of the same size, generated using \texttt{tauola-bbb} with the modifications described in this directory. Comparison output \texttt{babar.vs.tauola-bbb.pdf} is the result of the MC-TESTER comparison of these two samples. The instructions located in this directory, show how to reproduce generation of the \texttt{tauola-bbb.root} file. See MC-TESTER documentation on how to perform comparison against \texttt{babar.root} sample to produce the pdf output.

### B.3 Example of adding LFV current to tauola-bbb

Directory \texttt{demo-lfv} expands the \texttt{demo-babar} example with the current described in Section 5, README located in this directory, describes modifications needed to add this new current to the project. In the directory, generation is set for sample of this single decay channel.

### B.4 Example of adding new channels and channel re-definition

Directory \texttt{demo-redefine} demonstrates options for the interface implementation described in Appendix A. File \texttt{iniofc.c} located in this directory contains extensive description of changes that can be introduced using this interface. This example does not introduce anything valuable in terms of physics content. Its use is to present the technical side of the project.

#### B.4.1 Example of $\tau \rightarrow \pi\pi\nu\tau$ current/matrix element replacement

The most useful example for learning how the new interface can be used is to replace one FORTRAN channel with it’s C++ version. In directory \texttt{demo-redefine} files \texttt{pipi0.c, pipi0.h, MEutils.c MEutils.h} contain C++ version of $\tau \rightarrow \pi\pi\nu\tau$ channel and its current. It is a copy of FORTRAN code\textsuperscript{14}. User can replace FORTRAN hadronic current or the whole matrix element with the C++ version. We have performed test to check that the C++ current gives exactly the same results as its FORTRAN counterpart. With this, user

\textsuperscript{14}We tried to make it as similar to FORTRAN counterpart as possible, so user could compare it step by step. Note that there are some solutions that do not work in C++ as a straightforward copies, e.g. filling 2-dim tables through simple equation, had to be replaced by two for loops.
explores baseline to edit the currents, as well as insight on how hadronic
current should look like, if they are written from scratch. We believe, that
this example should prepare user for developing his own channels. To turn
this example on or off one should inspect demo-redefine/iniofc.c file.

B.5 Using tauola-bbb with Tauola Universal Interface

Directory patch-tauolapp contains README file for the procedure needed
to import tauola-bbb into the C++ Interface of Tauola [10]. The steps
are fairly simple. They boil down to copying the tauola-bbb subdirectory
into TAUOLA/tauola-fortran subdirectory of the C++ interface and applying
small patches that allow the directory to be used from the C++ interface.

B.6 Using tauola-bbb with KKMC

Similarly, directory patch-KK-face includes README file for the procedure
needed to use tauola-bbb with KKMC [15]. The key element is to correctly
pass the branching ratio for the new channels provided by tauola-bbb. To
achieve this, modifications described in README have to be applied to the
default KKMC setup.

Let us provide as an example installation of FCC project [29, 30]. This
version is of importance because of plans for the Future Circular Colider
of CERN. In principle tauola-bbb/patch-KK-face/README contains all of
the required information, however complication due to evolution of software
environments have to be taken into account. In particular new Makefile sys-
tem of KKMC requires use of command alias kmake='make -f KKMakefile',
which changes use of all old command from make command to kmake command.

B.7 New tar-ball modifications

The distribution tarball discussed in this paper is almost exactly the same as
the version announced in [28]. Only minor changes were introduced, mostly
affecting documentation (README files and examples of program use) but
not the actual code.

In particular:

1. README-changelog in main tauola-bbb directory has been updated

and one incorrect printout in tauola-c/ChannelForTauola.h has been
fixed.
2. In file `pkorb.f`, lines 47-48 were added to initialize BRA1 and BRKS to zero. It was to avoid undefined variables, which for some compilers may not be set to zero anyway, and resolve potential problem with CLEO parametrization. Note that this parametrization is presently defunct.

3. Instead of single `demo-standalone` of older versions of TAUOLA, several new, but similar examples described in previous Subsection have been prepared.

4. We have renamed folders to make clear distinctions between run examples (demo-*), and instructions for patching and validations (patch-*). README files were updated and some comments were added to account for the above changes.

5. Prints for outputs have been updated: new version number and release date has been provided.

6. Minor bugfixing, e.g. prints from `makefile`'s, `KK defaults adendum.txt`.

C Technical prints of default initialization

In previous Appendix an example of how to run the demo programs provided with the distribution was given. For the sake of debugging, a function `Tauolapp::PrintChannelInfo(int i)` have been provided it is declared in `tauola-c/ChannelForTauolaInterface.h` file. It prints the information, related to the decay channel `i`, and stored in the structs described in Section 3. One can use this function to monitor the initialization.

The first column of the print, denotes consecutive channel number, the next its name which will appear on TAUOLA printouts. The `!nu_tau` denotes that \( \nu_\tau \) is absent from \( \tau \) decay products. The `b.ratio` denotes probability for the given channel to be generated. The sum of `b.ratio` over all channels, does not need to be equal 1. It is normalized before event generation anyway. The relative position on the sub-list of decay channels for the particular phase-space generator follows. Finally type of the matrix element as explained in Section 3 and identifiers of final state \( \tau \) decay products are given. Zero denotes that particular identifier is not used. This is usually the case of all entries after vertical line `|`. An exception, necessary for anomalous decays, are position 3 and 4 which are identifiers for particles to replace default \( \nu_\tau \) among \( \tau \) decay products, valid if they follow `|` in the printout of course.
Whenever a pointer to user function for matrix element or current calculation was properly set, a line: Pointer to user function set correctly.

C.1 Default BaBar initialization

The printout of the default initialization of BaBar currents, no user modifications introduced:
