Fast simulation and reconstruction package for TOP counters

Marko Starič

v3.0 (February 23, 2010)

1 Introduction

TOPsimrec is a fast simulation/reconstruction package for TOP counters. The reconstruction is based on an extended likelihood method, which uses analytical expressions for the likelihood function [1]. The simulation and the reconstruction can be used separately from each other; one can use the simulation output as the input to some other reconstruction package or vice verse. Version v3.0 is based on a Fortran 77 code, on top of which a C++ user interface is superimposed. This version enables the simulation/reconstruction for the non-focusing and focusing TOP counters (cylindrical or spherical mirror) as well as the i-TOP counters with or without the focusing mirror.

The package is available as a tar file; Unpacking the tar file creates a directory TOPsimrec-v3.0 with the following subdirectories:

docu	documentation
examples	examples and makefile
hbo	place for the output hbook file(s) from top_simu.cc
include	C++ include files (user interface)
obj	place for compiled (object) files
paw	paw macros to analyze output from top_simu.cc
src	C++ source code
srcF77	F77 source code

The subdirectory examples includes: an example of how to define the TOP counter geometry and set other relevant parameters (TOPconfigure.cc), one simple example (top_test.cc) and one more sophisticated example (top_simu.cc) of how to use simulation/reconstruction, and a makefile. The makefile may eventually need to be tailored to the particular system configuration, like replacing gfortran compiler with the older g77 compiler. Some of the floating point exceptions are enabled (see srcF77/trapfpe.c) by including trapfpe.o in the linking list. They are not handled anywhere but just abort the program execution. To turn them off just remove trapfpe.o from the linking list.

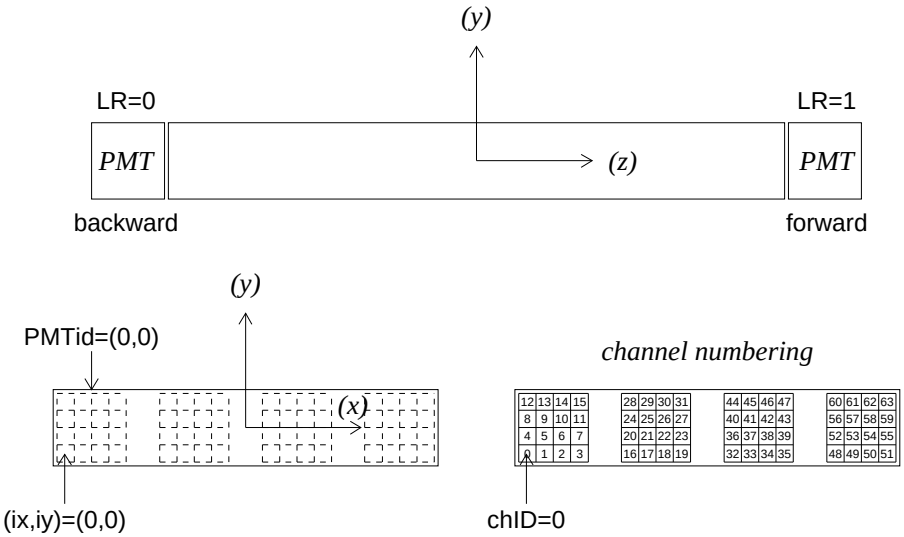


Figure 1: Local (internal Q-bar) frame and numberings.

2 Units, frames and numbering schemes

The quantities are expressed in the following units:

detector geometry, lengths, positions etc.	centimeters [cm]
time, time resolutions etc.	nanoseconds [ns]
group and phase velocity	cm/ns
wavelength	nanometers [nm]
photon energy	electron volts [eV]
magnetic field	Tesla [T]

Two frames are used. MC particles and reconstructed tracks are given in the global (Belle) frame; detected photons are given in the local frame of the corresponding Q-bar. As shown in Fig. 1, the local frame has the origin at the Q-bar center; the z-axis is aligned along the Q-bar and points in the same direction as that of the global frame; y-axis is perpendicular to the Q-bar; the PMT's are positioned along the x-axis. Intersection of a reconstructed track with the Q-bar can be retrieved in both, local and global frames (see `TOPreco::GetHit`).

The numberings of Q-bars, channels, PMT's etc. start with 0. The Q-bars are numbered in the same order as they've been defined by the calls to `setQbar`. The channel number (`chID`) is composed from a PMT number (`PMTidX, PMTidY`), a channel number within the PMT (`ix,iy`) and an exit window number (`LR`), see Fig. 1, as follows:

- If backward equipped (LR=0)
  `chID=ix+Nx*(iy+Ny*(PMTidX+NumPMTx(LR,QbarID)*PMTidY))`
- If forward equipped (LR=1)

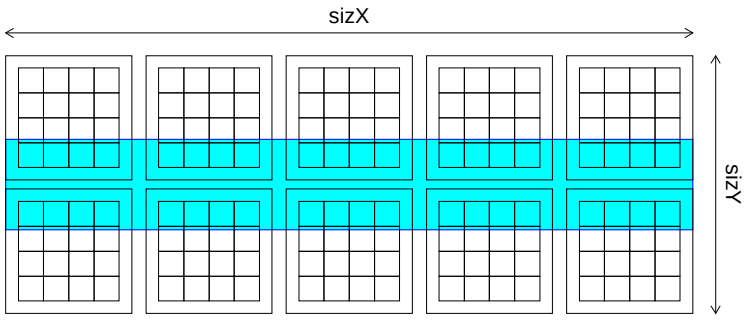


Figure 2: Arrangement of PMT's at the Q-bar exit window (shown in light blue).

```
chID=NumCh(LR=0)+
ix+Nx*(iy+Ny*(PMTidX+NumPMTx(LR,QbarID)*PMTidY))
```

where `Nx` and `Ny` are the number of PMT channels in x and y, and `NumPMTx` is the number of PMT's in x attached to the (single) exit window of a particular Q-bar. Note that the channels are numbered for both Q-bar exit windows in the ascending order as x increases. The conversion functions defined in include/TOPutil.h can be used to cross-check the numbering schemes.

The number of Q-bars is limited to 100 and the number of channels per Q-bar is limited to 1024. One can increase the limits by changing the corresponding parameter values in srcF77/TOP_GEO.fi (`NSIZ_MOD`) and srcF77/TOP_PIK.fi (`NUM_CHA`), clean all object files and recompile the code.

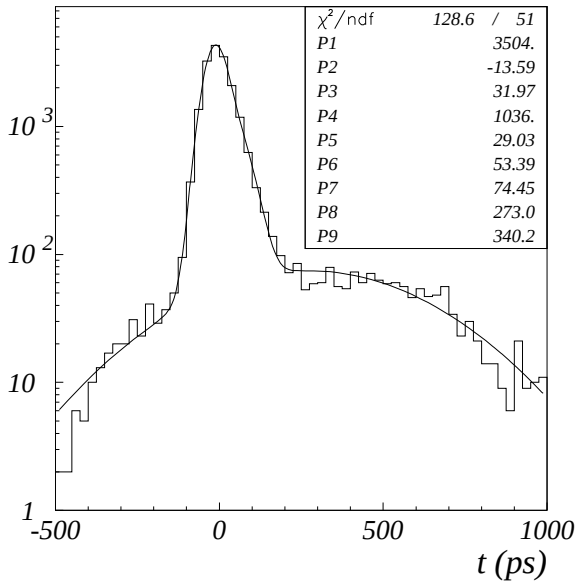
3 Configuring the TOP counter

Functions needed to setup a TOP counter configuration are defined in include/TOPconfig.h file and an example how to do it is given in examples/TOPconfigure.cc. It is possible to define almost any cylindrical geometry, including the overlapped versions*.

The configuration code must begin with the call to `TOPvolume` to reset the number of Q-bars to zero and must end with the call to `TOPfinalize`. The PMT must be defined prior to Q-bars (using `setPMT`), since its size is needed to arrange the PMT's at the Q-bar exit window(s). By default they are arranged so that the left-side edge of the first PMT and the right-side edge of the last PMT are aligned with the edges of the Q-bar. For the i-TOP the upper and lower PMT edges are aligned with the edges of expansion volume exit window as well. One can re-arrange PMT's with `arrangePMT` using the values of `sizX` and `sizY` different from the exit window width and hight (Fig. 2). Note that `sizY=0` alignes PMT's in one row. The whole PMT array can also be shifted by `Dx` and `Dy` in respect to the exit window.

The quantum efficiency is specified by `setQE` functions. The data points can be either read from an ascii file (see examples/qe_*.dat) or specified with two arrays holding the predefined

*the reconstruction for overlapped geometry is not entirely supported in v3.0



fraction	mean [ps]	sigma [ps]
0.5815	-13.59	31.97
0.2870	29.03	53.39
0.1315	273.0	340.2

Figure 3: TTS distribution [2] fitted with a sum of three Gaussians.

data points. The data points do not need to be equidistant. In addition there are two functions with predefined QE for GaAsP and multi-alkali photo cathodes with the values provided by Inami-san [2]. In all cases the electron collection efficiency (`CE`) is a separate input parameter.

The time transition spread (TTS) distribution is specified by `setTTS`. To be able to express the likelihood function analytically, it is parametrized as a sum of `ng` Gaussians ($\texttt{ng} \leq 8$); the one provided by Inami-san [2] is found to be fitted well with the sum of three Gaussians (Fig. 3).

The TDC specifications (number of bits, channel width and offset) and the magnetic field[†] must also be provided. They are set by `setTDC` and `setBfield`.

The Q-bars are specified by `setQbar`; the function arguments are defined in Fig. 4. The default mirror radius for focusing Q-bar is set to $R = 2c$, where c is the Q-bar length; the optical axis (mirror center) is set to $x_c = 0, y_c = b/2$, where b is the Q-bar thickness. They can be changed by calling `setMirrorRadius` and `setMirrorCenter`.

The expansion volume of i-TOP is added to the Q-bar by `addExpansionVolume` function. One has to specify `QbarID` and the side to which it is attached, its shape (*Prism* or *Box*, see fig. 5), the length (`Dz`) and the y coordinates (Q-bar frame) of the upper and lower exit window edges (`Yup, Ydown`)[‡].

4 Simulation

The interface to the F77 code is provided by the class `TOPsimu` defined in include/TOPsimu.h. The constructor has two arguments to specify T_0 jitter (in nanoseconds) and the average number

[†] $B = 0$ is also supported now

[‡] note the changes with respect to previous versions

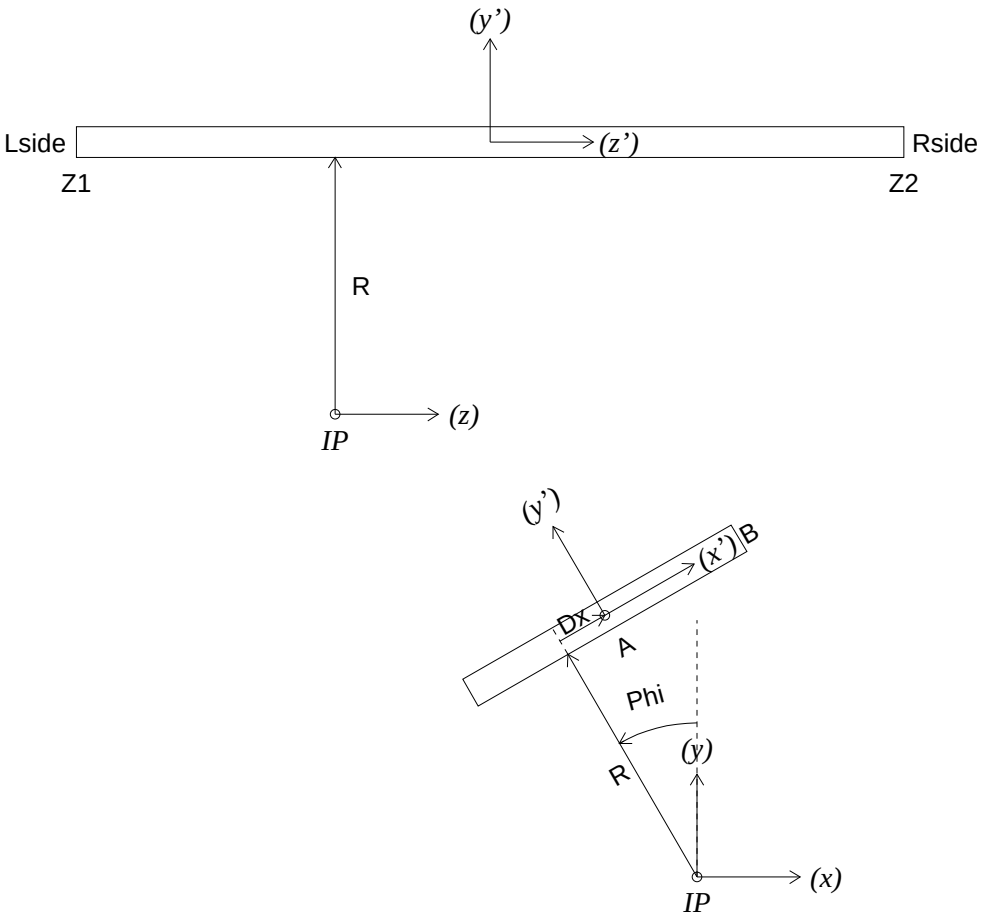


Figure 4: Arguments of the `setQbar` function.

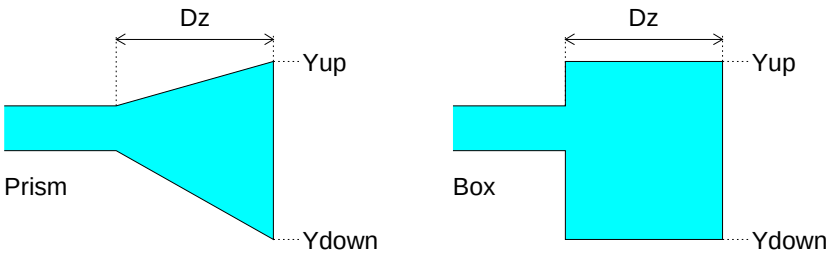


Figure 5: Expansion volume shapes.

of background hits in the TDC time range per exit window.

Particles are added to the simulation by a member function `AddParticle`, specifying for each one the starting point [cm], the starting time [ns] in respect to the primary interaction time, the momentum vector [GeV/c] and the LUND code, and optionally a reference label (positive integer). Particles with a LUND codes different from that of (charged) e, μ, π, K, p are ignored.

The simulation is called with a member function `Simulate`. The functions for retrieving the simulation output are also provided; with `GetNphot` or with `NphExit` and `NphDigi` one can get for a given particle the number of detected photons; calling `GetData` in a loop[§] one can get the simulated TOP counter output.

5 Reconstruction

The interface to the F77 code is provided by the class `TOPreco` defined in include/TOPreco.h. The mass hypotheses are specified with the class constructor; the expected average number of background hits per exit window per TDC time range (assumed to be uniformly distributed) is also a constructor argument. Optionally one can specify also the scaling factor for N_0.

The TOP counter data (detected photons) are input to the reconstruction by a member function `AddData` using the numbering schemes defined in section 2. One can use functions defined in include/TOPutil.h to cross-check or convert. The reconstruction is run for each (reconstructed) track by a member function `Reconstruct`. The track is specified by the point (`X`, `Y`, `Z`) , the length of the track from IP to that point (`Tlen`)[¶] , the momentum vector (`Px`, `Py`, `Pz`) and the charge (`Q`). It is recomended to give the track parameters at or close to the track intersection with the Q-bar.

The member function `Flag` returns the reconstruction status (1=OK, 0=out of acceptance, -1=inside a gap between Q-bars); log likelihoods, the expected number of photons and the measured number of photons are returned by `GetLogL`; `GetHit` returns the track parameters at the intersection with the Q-bar in a local or global frame.

In order to obtain proper results from the reconstruction, the likelihood function (PDF) must match the TOP data. To check the matching, a member function `GetPull` is provided; it returns the arguments, if the true track identity in the call to `Reconstruct` (argument `HYP`> 0) is given. The default track identities are labeled as 1=e, 2=μ, 3=π, 4=K and 5=p. If masses are not given in this order the identities must be specified explicitly with `SetHypID`. How to use `GetPull` is shown in examples/top_simu.cc; the histograms from this example are plotted in Fig. 6 and can serve as reference plots. A function to retrieve the PDF for a given mass hypethesis is also provided (see examples/get_pdf.cc).

[§] the list counter begins with 0

[¶] note the change with respect to the previous versions

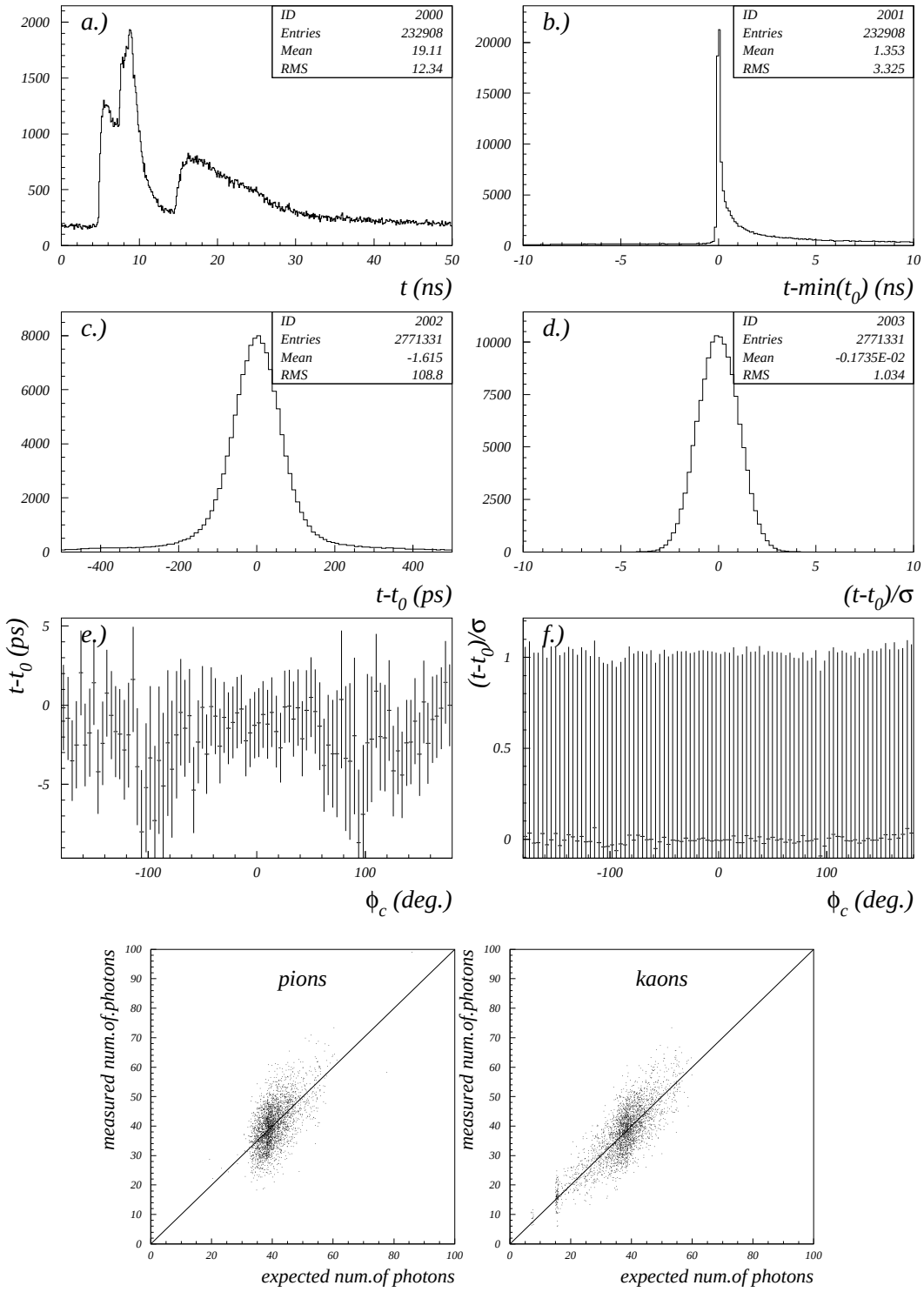


Figure 6: Example of histograms to check matching between data (t) and likelihood function (t_0, σ). (a) time distribution of TOP data, (b)-(f) time differences and pulls, should peak at zero; pulls should have r.m.s close to one; ϕ_c is the azimuthal Cherenkov angle. At bottom: measured versus expected number of photons (including background) for pions (left) and kaons (right). Events should be distributed around the diagonal line.

6 Examples

6.1 Defining 2-readout f-TOP configuration

```
#include "TOPconfig.h"

void TOPconfigure()
{
  TOPvolume(115, 125, -80, 190);
  setBfield(1.5);
  setPMT(2.75, 2.75, 2.2, 2.2, 1, 4);
  double frac[3]={0.5815, 0.2870, 0.1315};
  double mean[3]={-13.59e-3, 29.03e-3, 273.0e-3};
  double sigma[3]={31.97e-3, 53.39e-3, 340.2e-3};
  setTTS(3, frac, mean, sigma);
  setQE("qe_GaAsP400nm.dat", 0.35);
  setTDC(10, 50.E-3);

  int n=18;
  double Pi=3.1415926535
  double Dphi=2*Pi/n; double Phi=0; int id;
  for(int i=0; i<n; i++){
    id=setQbar(40, 2, -78, 107, 118, 0, Phi, PMT, SphericM);
    setMirrorRadius(id, 500);
    id=setQbar(40, 2, 108, 183, 118, 0, Phi, None, PMT);
    Phi+=Dphi;
  }
  TOPfinalize();
}
```

6.2 Defining focusing i-TOP with an asymmetric expansion volume

```
int n=16;
double Dphi=2*Pi/n; double Phi=0; int id;
for(int i=0; i<n; i++){
  id=setQbar(44, 2, -80, 190, 115.8, 0, Phi, PMT, SphericM);
  setMirrorRadius(id, 720);
  addExpansionVolume(id, Left, Prism, 7.89, 1.00, -4.15);
  Phi+=Dphi;
}
```

6.3 Using `TOPreco` class for reconstruction

```
#include "TOPreco.h"

double Masses[3]={.13957, .49368, .93827}; int Nhyp=3;
double Bkg=1.5; // expect this number of background hits

void RecEvent()
{
  TOPreco reco(Nhyp, Masses, Bkg);
  reco.Clear();
  for(...){ //loop over your digitized data (all bars)
    ...
    reco.AddData(QbarID, chID, TDC);
  }
  for(...){ //loop over tracks in the event
    ...
    reco.Reconstruct(x,y,z,trklen,px,py,pz,q);
    if(reco.Flag() == 1) {
      double LogL[Nhyp], ExpPhot[Nhyp]; int Nphot;
      reco.GetLogL(Nhyp, LogL, ExpPhot, Nphot);
      ...
    }
  }
}
```

References

[1] M. Starič *et.al.*, Nucl. Instr. Meth. A**595**, 252 (2008).

[2] K. Inami, [pid_up:0160] TOP simulation (e-mail, 10 Apr 2009)