Tau1P3P

* The complementary algorithm for hadronic tau reconstruction and identification, tau1P3P, in the transverse momentum range below ~70 GeV (visible decay products) is track-based, with energy-scale given by energy-flow approach. There are two versions of this algorithm:

- Tau1P3P integrated into tauRec package.
- Tau1P3P stand-alone available as "user analysis" code from ESD.
- parametrisation for tau1p3p performance of Tau1P3P integrated for fast simulation

* A parametrisation of the Tau1P3P performance for fast simulation has been included in ATLFAST for release 12.0.4. Instructions on it suse and other information can be found on the Tau1P3Pfastsimulation page.

Tau1P3P - neural network and advanced cuts

The use of neural network or optimized cuts improves the selection using Tau1p3p package. They both can be used on the AANT or AOD level.

Neural network is created using SNNS package, the trained network is converted into "C" code. There is also a function added, which corrects the NN output in order to obtain the flat signal efficiency in E_T_vis. Also the discriminant is corrected to be flat, so for example a cut on NN discriminant to be greater than 0.3 corresponds to 70% signal efficiency.
The example code is here (Tau1p3pNNandCuts.tar.gz), in the directory snns there are three "C" functions snns_1P.c , snns_2P.c and snns_3P.c containing three neural networks trained on 1P, 2P or 3P events. The functions mySNNS_1P.c , mySNNS_2P.c and mySNNS_3P.c are the correcting functions and they return both the corrected and raw snns output. They should be compiled first doing:

```
cd snns

make

cd ..
```

Then the obtained libraries should be linked with the C++ code. In the supplied example the C++ code is running under root.
The input to the NN function is an array of **floats** containing the discriminating variables**,** so one should be carefull not to supply doubles.

The cuts are optimized using TMVA package. First the data are decorrelated, than many random cuts are applied. The set of cuts with a desired efficiency and maximal background rejection is chosen. The cuts are encoded in the functions: tau1p3pCut1P.C , tau1p3pCut2P.C and tau1p3pCut3P.C, which should be included to the code. The input is a vector of **doubles** containing the discriminating variables:

The example of code using neural network and improved cuts is here:

```
#include "tau1p3pCut1P.C"
#include "tau1p3pCut2P.C"
#include "tau1p3pCut3P.C"

extern "C" float nnOutput1P(float rdata[], float m_snnsOut[2]);
extern "C" float nnOutput2P(float rdata[], float m_snnsOut[2]);
extern "C" float nnOutput3P(float rdata[], float m_snnsOut[2]);

// [........]
```

```
   if ((*tau1p3p_ntrack)[itauR] == 1 && m_tau_prong[matchedTau]==1)
   {


// here we use a Neural network and cuts
      const int size=9;
      float out[size];


 out[0] = (*tau1p3p_ETeflow)[itauR];
 out[1] = (*tau1p3p_nStrip)[itauR];
 out[2] = (*tau1p3p_stripWidth2)[itauR];
 out[3] = (*tau1p3p_IsoFrac)[itauR];
 out[4] = (*tau1p3p_EMRadius)[itauR];
 out[5] = (*tau1p3p_cellsChrgHAD)[itauR]/(*tau1p3p_ptTrack1)[itauR];
 out[6] = ((*tau1p3p_cellsOtherEM)[itauR] + (*tau1p3p_cellsOtherHAD)[itauR])/(*tau1p3p_ETcalo)[it
 out[7] = (*tau1p3p_MVisEflow)[itauR];
 out[8] = (*tau1p3p_NAssocTracksIsol)[itauR];



      nnOutput1P(out,m_snnsOut);


     // cuts
      Double_t  p_arr_d[9];
      for (int i=0; i<9; i++) p_arr_d[i]=out[i];

      m_discriCuts = tau1p3pCut1P(p_arr_d);

      cout<<"1P signal, new cuts: "<<m_discriCuts<<" neural net raw: "<<m_snnsOut[0]<<" corrected




   }

   if ((*tau1p3p_ntrack)[itauR] == 2 &&  m_tau_prong[matchedTau]==3 )
   {


// Neural network
      const int size=11;
      float out[size];


 out[0] = (*tau1p3p_ETeflow)[itauR];
 out[1] = (*tau1p3p_nStrip)[itauR];
 out[2] = (*tau1p3p_stripWidth2)[itauR];
 out[3] = (*tau1p3p_IsoFrac)[itauR];
 out[4] = (*tau1p3p_EMRadius)[itauR];
 out[5] = (*tau1p3p_cellsChrgHAD)[itauR]/
      ((*tau1p3p_ptTrack1)[itauR]+(*tau1p3p_ptTrack2)[itauR]);
 out[6] = ((*tau1p3p_cellsOtherEM)[itauR] + (*tau1p3p_cellsOtherHAD)[itauR])/(*tau1p3p_ETcalo)[it
 out[7] = (*tau1p3p_MVisEflow)[itauR];
 out[8] = (*tau1p3p_NAssocTracksIsol)[itauR];
 out[9]  = (*tau1p3p_MTrk3P)[itauR];
 out[10] = (*tau1p3p_RWidth2Trk3P)[itauR];


//      float m_snnsOut[2];
      nnOutput2P(out,m_snnsOut);


     // cuts
      Double_t  p_arr_d[11];
      for (int i=0; i<11; i++) p_arr_d[i]=out[i];
```

```
      m_discriCuts = tau1p3pCut2P(p_arr_d);

      cout<<"2P signal, new cuts: "<<m_discriCuts<<" neural net raw: "<<m_snnsOut[0]<<" corrected
```

```
   }

   if ((*tau1p3p_ntrack)[itauR] == 3 && m_tau_prong[matchedTau]==3 )
   {
     m_tau_matched[matchedTau]     = 3;
     m_tau_matched_pos[matchedTau] = itauR;
     m_threePrec++;
```

```
// mw added
      const int size=11;
      float out[size];

 out[0] = (*tau1p3p_ETeflow)[itauR];
 out[1] = (*tau1p3p_nStrip)[itauR];
 out[2] = (*tau1p3p_stripWidth2)[itauR];
 out[3] = (*tau1p3p_IsoFrac)[itauR];
 out[4] = (*tau1p3p_EMRadius)[itauR];
 out[5] = (*tau1p3p_cellsChrgHAD)[itauR]/
      ((*tau1p3p_ptTrack1)[itauR]+(*tau1p3p_ptTrack2)[itauR]+(*tau1p3p_ptTrack3)[itauR]);
 out[6] = ((*tau1p3p_cellsOtherEM)[itauR] + (*tau1p3p_cellsOtherHAD)[itauR])/(*tau1p3p_ETcalo)[it
 out[7] = (*tau1p3p_MVisEflow)[itauR];
 out[8] = (*tau1p3p_NAssocTracksIsol)[itauR];
 out[9]  = (*tau1p3p_MTrk3P)[itauR];
 out[10] = (*tau1p3p_RWidth2Trk3P)[itauR];


      nnOutput3P(out,m_snnsOut);


     // cuts
      Double_t  p_arr_d[11];
      for (int i=0; i<11; i++) p_arr_d[i]=out[i];

      m_discriCuts = tau1p3pCut3P(p_arr_d);

      cout<<"3P signal, new cuts: "<<m_discriCuts<<" neural net raw: "<<m_snnsOut[0]<<" corrected
```

```
   }
}
```

RTT

For continuous testing of the tau reconstruction packages RTT has been set up for tauRec and tau1P3P. Test are run in the tauRec package for ESD/CBNT variables as well as in the CBNT_AOD package for AOD variables.
The ESD/CBNT tests run on Z->tautau samples from the rome production (rome.004807).

The AOD tests are performed on a small ttbar sample (mc11.004100) of 100 events and include at the moment only basic plots of the AOD variables for tauRec (TauJetContainer) and tau1P3P (TauJet1p3pContainer). Both, ESD/CBNT and AOD variables are tested after full reconstruction of digit data samples.
All RTT results cab be found here.

How to report bugs for tauRec

Please note that since now only bugs reported in two ways described below will be fixed:

- Savannah - preferred one - if you notice a bug please issue a bug report using CERN's Savannah web service: https://savannah.cern.ch/projects/atlas-reco/
- If you're not sure if what you can see is really a bug please notify us about your concerns by sending e-mail to (include all listed addresses):
  Anna Kaczmarska <ANNA.KACZMARSKA@IFJ.EDU.PL _moz-userdefined="">
  Michael Heldmann <HELDMANN@UNI-MAINZ.DE _moz-userdefined="">
  Lukasz Janyst <LJANYST@CERN.CH _moz-userdefined="">

While sending us bug report please include the following information: jobOption, info about release, info about data sample and anything you think might be helpful. Thank you for your understanding and help.

- tr44vstr42.ps: tr44vstr42.ps
- 1205vs1204.ps: Validation of kit 12.0.5
- Tau1p3pNNandCuts.tar.gz: Example of using Tau1p3p NN and new cuts

---

This topic: Main > Tau1P3P
Topic revision: r1 - 2007-05-16 - AndrewHamilton