08/04/2021

Calibration for Calo

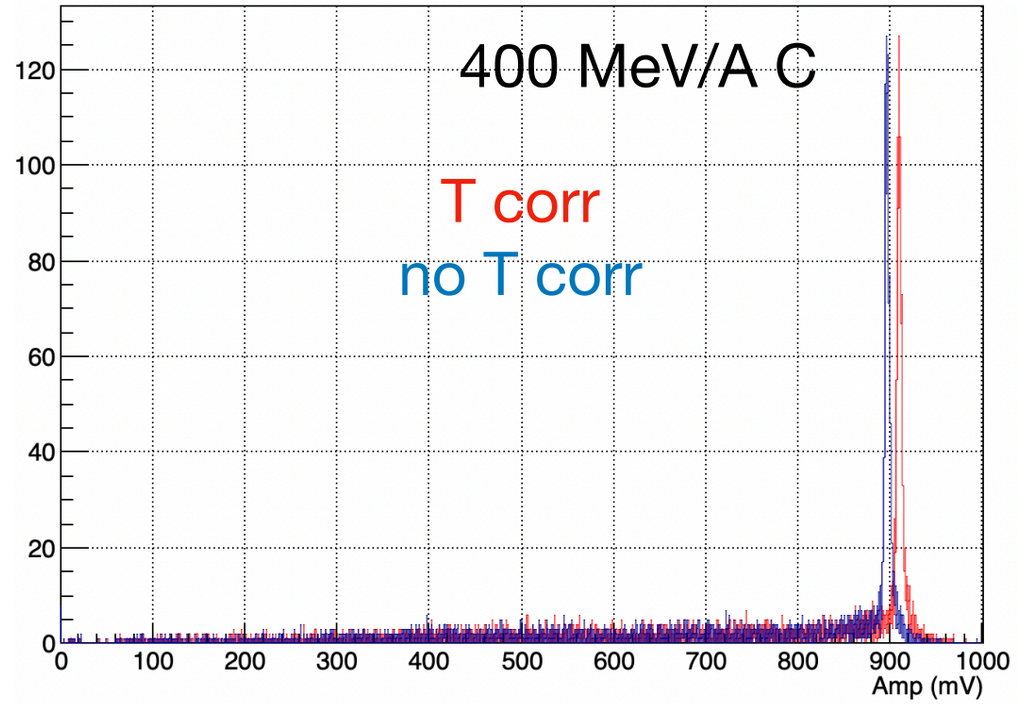
Software Meeting

F. Cavanna, L. Scavarda

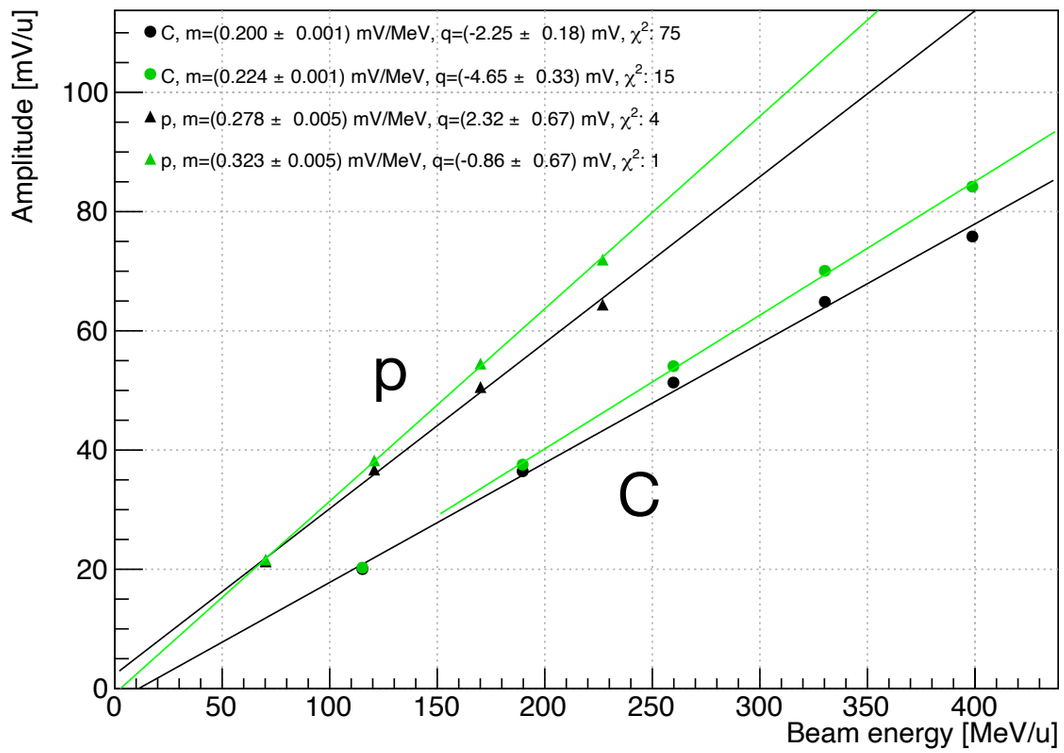
Calibration Steps

Goal: obtain a linear and calibrated response

1) Calibration steps:
   - Temperature corrections
   - Crystal inter-calibration factors
   - Energy calibration (ADC to MeV)



2) Linearity:
   - Position correction
   - Birks effect (?)



News in the newgeom branch

Thanks to Giacomo the WaveDream classes also for the calo are now implemented

↓

The calibration chain is under development

↓

A new campaign was created in the *newgeom* branch for this study: *testCalo*

↓

In testCalo the data acquired at GSI with channel *7* of board *B27* of the WaveDream were attributed to the calorimeter (it was *ST* during the campaign)

↓

No physical results

Calibration Steps in SHOE

Reconstruction/level0/config/testCALO/

TACAdetector.map

```
#number of crystal present
CrystalsN: 9

#crysid crymodule channelID crysBoard(HW) activecrys
0       0        0         27            1
1       0        1         27            1
2       0        2         27            1
3       0        3         27            1
4       0        4         27            1
5       0        5         27            1
6       0        6         27            1
7       0        7         27            1
8       0        8         27            1
```

TACAparMap.cxx

```
//------------------------------------------+-----------------------------------
//! Read mapping data from file \a name .
Bool_t TACAparMap::FromFile(const TString& name)
{
  Clear();

  if (!Open(name))
    return false;

  // read for parameter
  Double_t* para = new Double_t[5];
  // Int_t nCrys = 0;

  // number of crystal
  ReadItem(nCrys);

  if (FootDebugLevel(1)) {
    printf("CrystalsN: %d\n", nCrys);
    printf("CrystalId ModuleId ChannelId BoardId ActiveCrystal \n");
  }

  // nCrystals = nCrys;
  // cout << "n crys: " << nCrys << endl;
  for (Int_t i = 0; i < nCrys; ++i) { // Loop over crystal

    // read parameters (boardId chId, crysId)
    ReadItem(para, 5, ' ', false);

    // fill map
    Int_t crysId      = TMath::Nint(para[0]);
    Int_t moduleId    = TMath::Nint(para[1]);
    Int_t channelId   = TMath::Nint(para[2]);
    Int_t boardId     = TMath::Nint(para[3]);
    Int_t activeCrys  = TMath::Nint(para[4]);
```

TACAparMap.hxx

```
Int_t           GetCrystalId(Int_t boardId, Int_t channelId);
Int_t           GetCrystalsN() const {return nCrys;}
Int_t           GetBoardId(Int_t cryId) {return fBoardId[cryId]; }
Int_t           GetChannelId(Int_t cryId) {return fChannelId[cryId]; }
Int_t           GetModuleId(Int_t cryId) {return fModuleId[cryId]; }
```

Calibration Steps in SHOE

Reconstruction/level0/calib/testCALO/

TACA_Calibration.cal

```
#CrId T int_calib
0 290 1
1 290 0.9
2 284 0.8
3 299 0.9
4 300 0.9
5 291 0.8
6 294 0.9
7 294 0.8
8 300 0.9
```

BaseReco::ReadParFiles( )

```
    parFileName = fCampManager->GetCurCalFile(TACAparGeo::GetBaseName(), fRunNumber);
    parCal->FromCalibFile(parFileName.Data());
  }
}
```

TACAparCal.cxx/hxx

```
//--------------------------------------------------+----------------------------------
Bool_t TACAparCal::FromCalibFile(const TString& name)
{

  Clear();

  TString name_calib_temp_cry = name;

  gSystem->ExpandPathName(name_calib_temp_cry);

  fMapCal->LoadCryCalibrationMap(name_calib_temp_cry.Data());

  Info("FromCalibFile()", "Open file %s for calibration\n", name_calib_temp_cry.Data());

  return kFALSE;
}
```

TACAcalibrationMap.cxx

```
//_____________________________________________________________________
void TACAcalibrationMap::LoadCryCalibrationMap(std::string FileName)
{

  if (gSystem->AccessPathName(FileName.c_str()))
    {
    Error("LoadCryCalibrationMap()","File %s doesn't exist",FileName.c_str());
    }

  ///////// read the file with Charge calibration

  ifstream fin;
  fin.open(FileName,std::ifstream::in);

  Int_t nCrystals = fpCalMap->GetCrystalsN();

  Int_t cryId[nCrystals];  // Id of crystal
  Double_t temp[nCrystals];   // temperature
  Double_t equalis[nCrystals];   // equalis factor

  if(fin.is_open()){

    int  cnt(0);
    char line[200];

    // loop over all the slat crosses ( nSlatCross*nLayers ) for two TW layers
    while (fin.getline(line, 200, '\n')) {

      if(strchr(line,'#')) {
        if(FootDebugLevel(1))
          Info("LoadCryCalibrationMap()","Skip comment line:: %s\n",line);
        continue;
      }

      sscanf(line, "%d %lf %lf",&cryId[cnt],&temp[cnt],&equalis[cnt]);
```

TACAactNtuRaw

GetTemperatureCorrection( )

```
//------------------------------------------+-----------------------------------
Double_t TACAactNtuRaw::GetTemperatureCorrection(Double_t charge, Int_t  crysId)
{

  SetTemperatureFunctions();
  SetParFunction();
  Double_t T0 = f_parcal->getCalibrationMap()->GetTemperatureCry(crysId);
  Double_t m1 = fTcorr1->Eval(charge);
  Double_t m2 = fTcorr2->Eval(charge);

  Double_t m0 = m1 + ((m2-m1)/(T2-T1))*(T0-T1);

  Double_t delta = (T1 - T0) * m0;

  Double_t charge_tcorr = charge + delta;

  return charge_tcorr;

}
```

GetEqualisationCorrection( )

```
//------------------------------------------+-----------------------------------
Double_t TACAactNtuRaw::GetEqualisationCorrection(Double_t charge_tcorr, Int_t  crysId)
{

  Double_t Equalis0 = f_parcal->getCalibrationMap()->GetEqualiseCry(crysId);
  Double_t charge_equalis = charge_tcorr*Equalis0;

  return charge_equalis;

}
```

Action( )

```
Bool_t TACAactNtuRaw::Action() {

   TACAdatRaw*   p_datraw = (TACAdatRaw*) fpDatRaw->Object();
   TACAntuRaw*   p_nturaw = (TACAntuRaw*) fpNtuRaw->Object();
   TACAparMap*   p_parmap = (TACAparMap*) fpParMap->Object();

  int nhit = p_datraw->GetHitsN();


  int ch_num, bo_num;

  for(int ih = 0; ih < nhit; ++ih) {
    TACArawHit *aHi = p_datraw->GetHit(ih);

    Int_t ch_num     = aHi->GetChID();
    Int_t bo_num     = aHi->GetBoardId();
    Double_t time    = aHi->GetTime();
    Double_t timeOth = aHi->GetTimeOth();
    Double_t charge  = aHi->GetCharge();

    // here needed mapping file
    Int_t crysId = p_parmap->GetCrystalId(bo_num, ch_num);

    if (crysId == -1) // pb with mapping
      continue;

    Double_t type=0; // I define a fake type (I do not know what it really is...) (gtraini)

    // here we need the calibration file
    Double_t charge_tcorr = GetTemperatureCorrection(charge, crysId);
    Double_t charge_equalis = GetEqualisationCorrection(charge_tcorr, crysId);
    Double_t energy = GetEnergy(charge_equalis, crysId);
    Double_t tof    = GetTime(time, crysId);
    p_nturaw->NewHit(crysId, energy, time,type);

  }
```

Conclusions & Next Steps

Conclusions:

- The WD classes are now implemented also for the calorimeter
- Added a campaign map file for the calorimeter setup
- Added a calibration file
- The first two steps of the calibration chain are implemented

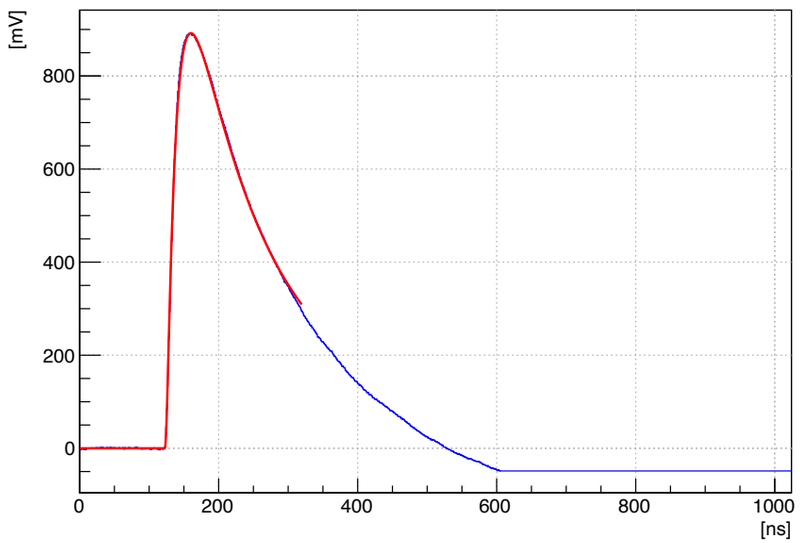
Next Steps:

- It would be good have “real” calorimeter data to validate calibration chain. How we can do? Any suggestions?
- Implement the energy calibration (ADC to MeV)
- Customise the computation of charge/amplitude for calo purpose.

```
double TACArawHit::ComputeCharge(TWaveformContainer *w){
  return TAGbaseWD::ComputeCharge(w);
}


double TACArawHit::ComputeAmplitude(TWaveformContainer *w){
  return TAGbaseWD::ComputeAmplitude(w);
}
```

Question:



With our standalone code we perform a fit on each pulse and from the fit we extrapolate the maximum amplitude and the integral (charge). Can we implement it also in SHOE or is too “heavy”?