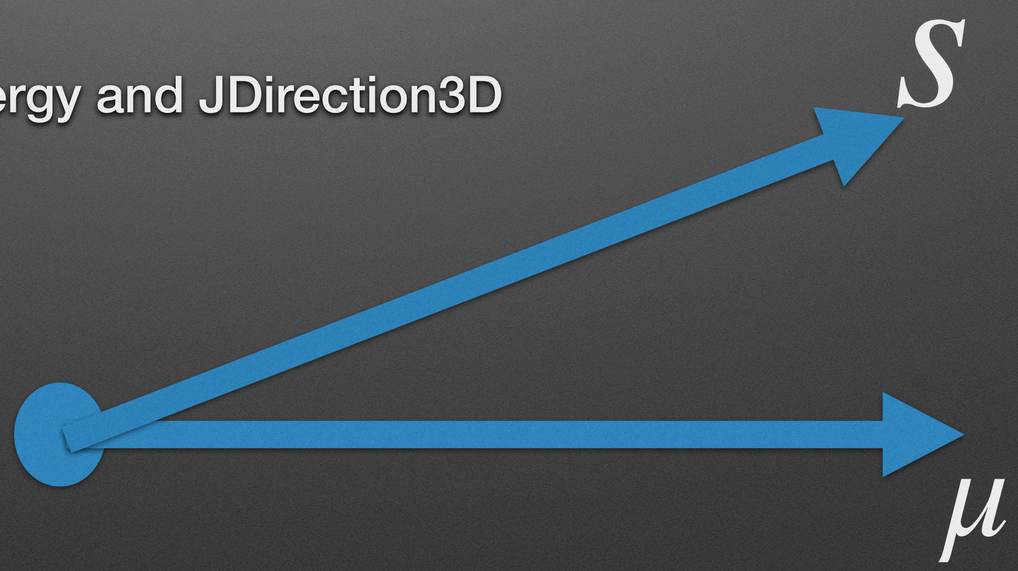
Geometry

- Describe an event vertex:
  3D vertex + time + 2 energies + two 3D directions
- JEventVertex.hh
- JEnergyFlow - class defining a JEnergy and JDirection3D

```
/**
 * 3D vertex.
 */
class JEventVertex :
  public JVertex3D
{
public:

  using JVertex3D::getT;


JEnergyFlowShower shower;
JEnergyFlowMuon   muon;
```



Combine PDFs

- JLoadPDFs.hh: Load and add the muon and shower PDF components
- Calculate the value of the PDF for such an event vertex:
  - for a given hit
  - for a given PMT
- Based on similar code in JLine3ZRegressor / JShower3EZRegressor
- Rotate the coordinate system separately for the muon and the shower

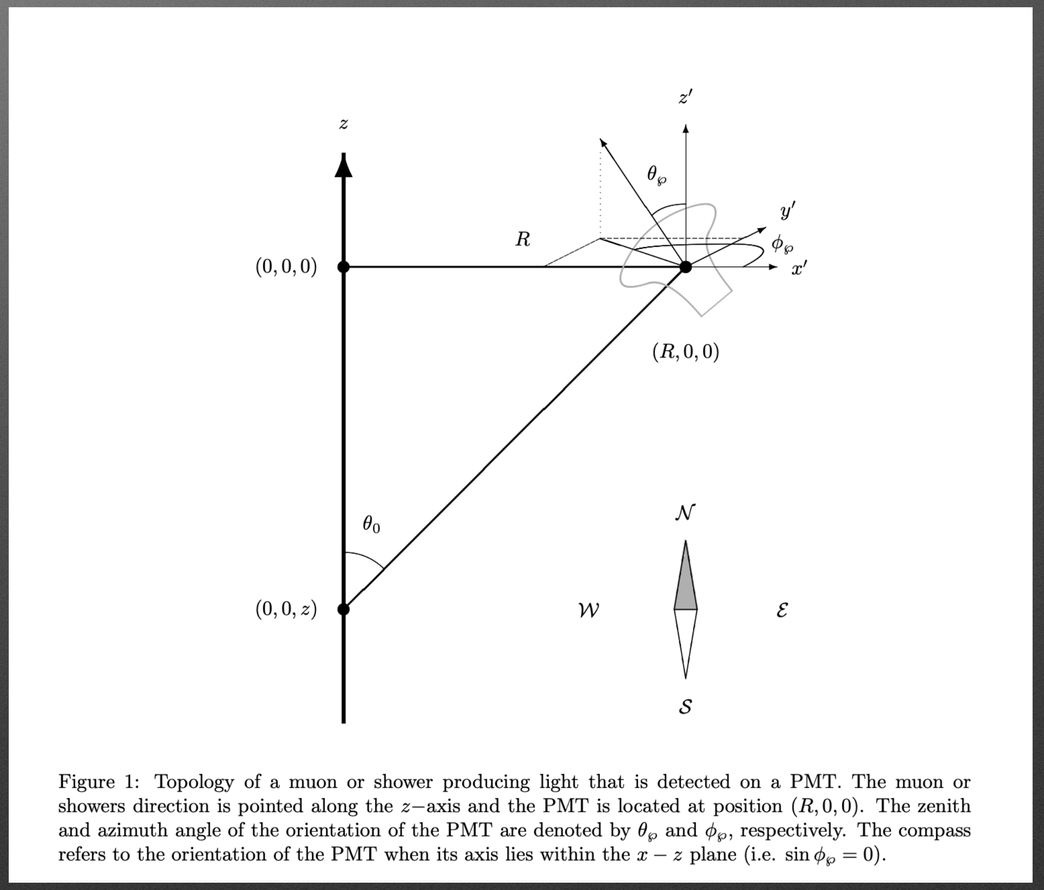


Figure 1: Topology of a muon or shower producing light that is detected on a PMT. The muon or showers direction is pointed along the $z-$axis and the PMT is located at position $(R, 0, 0)$. The zenith and azimuth angle of the orientation of the PMT are denoted by θ_φ and ϕ_φ, respectively. The compass refers to the orientation of the PMT when its axis lies within the $x - z$ plane (i.e. $\sin \phi_\varphi = 0$).

JPDF.pdf - https://common.pages.km3net.de/jpp/JPDF.PDF

Combine PDFs

- Value of muon PDF
- Position, direction of hit, calculate geometry..
- Calculate arrival time assuming Cherenkov hypothesis

$$\Delta t = t_{hit} - time_{eventvertex} - (z + R\tan(\theta_C)) * 1/c$$

```
//*********************************
//given hit assuming muon from the vertex
    template<class JHit_t>
    double operator()(const JEventVertex& eventvertex, const JHit_t& hit) const
    {
      using namespace JPP;
      using namespace JGEOMETRY3D;
      using namespace std;

      JPosition3D  D(hit.getX(),  hit.getY(),  hit.getZ());
      JDirection3D U(hit.getDX(), hit.getDY(), hit.getDZ());

      D.sub(eventvertex.getPosition());
      cout << "D.getposition " << D.getPosition() << endl ;
      cout << "U.getDIrection " << U.getDirection() << endl ;

      const double z  = D.getDot(eventvertex.muon.getDirection());
      const double x  = D.getX()  -  z * eventvertex.muon.getDX();
      const double y  = D.getY()  -  z * eventvertex.muon.getDY();
      const double R2 = D.getLengthSquared() - z*z;
      const double R  = (R2 > Rmin_m*Rmin_m ? sqrt(R2) : Rmin_m);

      const double t1 = eventvertex.getT() + (z + R * getTanThetaC()) * getInverseSpeedOfLight();


      U.rotate(JRotation3Z(-atan2(y,x)));                 // rotate PMT axis to x-z plane

      const double theta = U.getTheta();
      const double phi   = fabs(U.getPhi());

      const double E  = eventvertex.muon.getE();//gWater.getE(E_muon_GeV, z);
      const double dt = T_ns.constrain(hit.getT()  -  t1);

//      cout << " E " << E << "hit " << hit.getX() <<  " hit t " << hit.getT() << " dt " << dt << " theta " << U.getTheta() << "phi" << U.getPhi()

//      JPDF_muon_t::result_type H0muon_hit = getH0muon(hit.getR(), dt);
      JPDF_muon_t::result_type PDFmuon_hit = getPDFmuon(E, R, theta, phi, dt);
```

Combine PDFs

- Similar for the shower PDF
- Note rotating back the coordinate system..
- Muon + Shower PDF values added together at the end

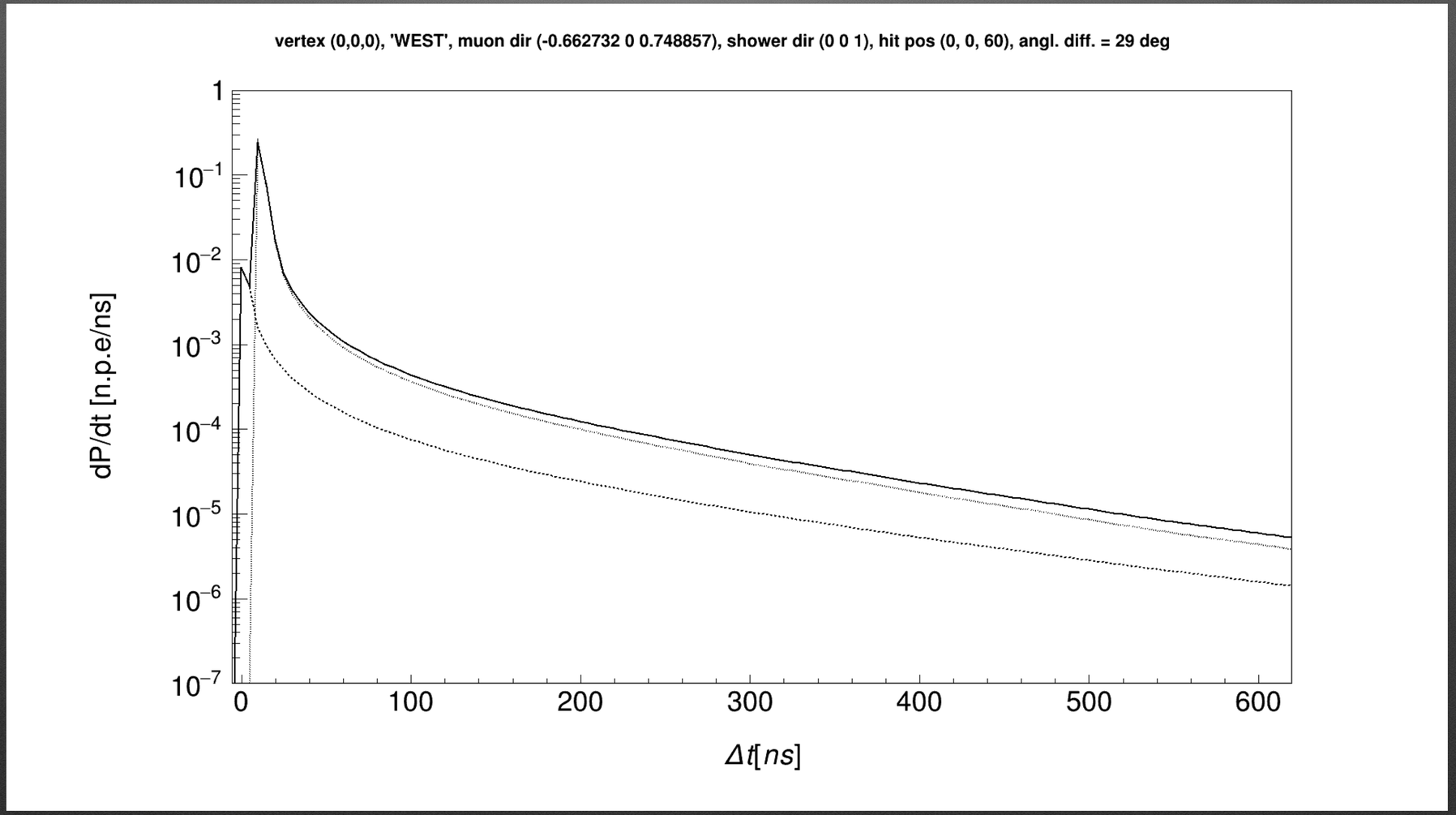
$$\Delta t = t_{hit} - time_{eventvertex} - (Dn) * 1/c$$

```
      U.rotate_back(JRotation3Z(-atan2(y,x)));                 // rotate PMT axis to x-z plane
//      cout << U.getDirection() << endl ;
     // PDF += H0;                                                                    // signal + background

// for shower
      const double z_shower  = D.getDot(eventvertex.shower.getDirection());
      const double x_shower  = D.getX()  -  z_shower * eventvertex.shower.getDX();
      const double y_shower  = D.getY()  -  z_shower * eventvertex.shower.getDY();
      const double cosDelta = z_shower/D.getLength();  // Delta = angle between shower direction and PMT position

//      cout <<  "cosDelta " << cosDelta << " xs " << x_shower << " ys " << y_shower << " zs " << z_shower  <<endl;

      U.rotate(JRotation3Z(-atan2(y_shower,x_shower)));                 // rotate PMT axis to x-z plane

      const double theta_shower = U.getTheta();
      const double phi_shower   = fabs(U.getPhi());

      const double t = eventvertex.getT() + (D.getLength() * getIndexOfRefraction() * getInverseSpeedOfLight());

      const double dt_shower = T_ns.constrain(hit.getT() - t);


      //      JPDF_shower_t::result_type H0 = getH0(hit.getR(), dt);  // getH0 = Get background hypothesis value
      JPDF_shower_t::result_type PDFshower_hit = getPDFshower(D.getLength(), cosDelta, theta_shower, phi_shower, eventvertex.shower.getE(), dt_shower);

      if (PDFshower_hit.V >= Vmax_npe) {
        PDFshower_hit *= Vmax_npe / PDFshower_hit.V;
      }
      //cout << " shower  PDF " << get_value(PDFshower_hit) << endl;

      double combinedPDF = get_value(PDFmuon_hit) + get_value(PDFshower_hit);
```

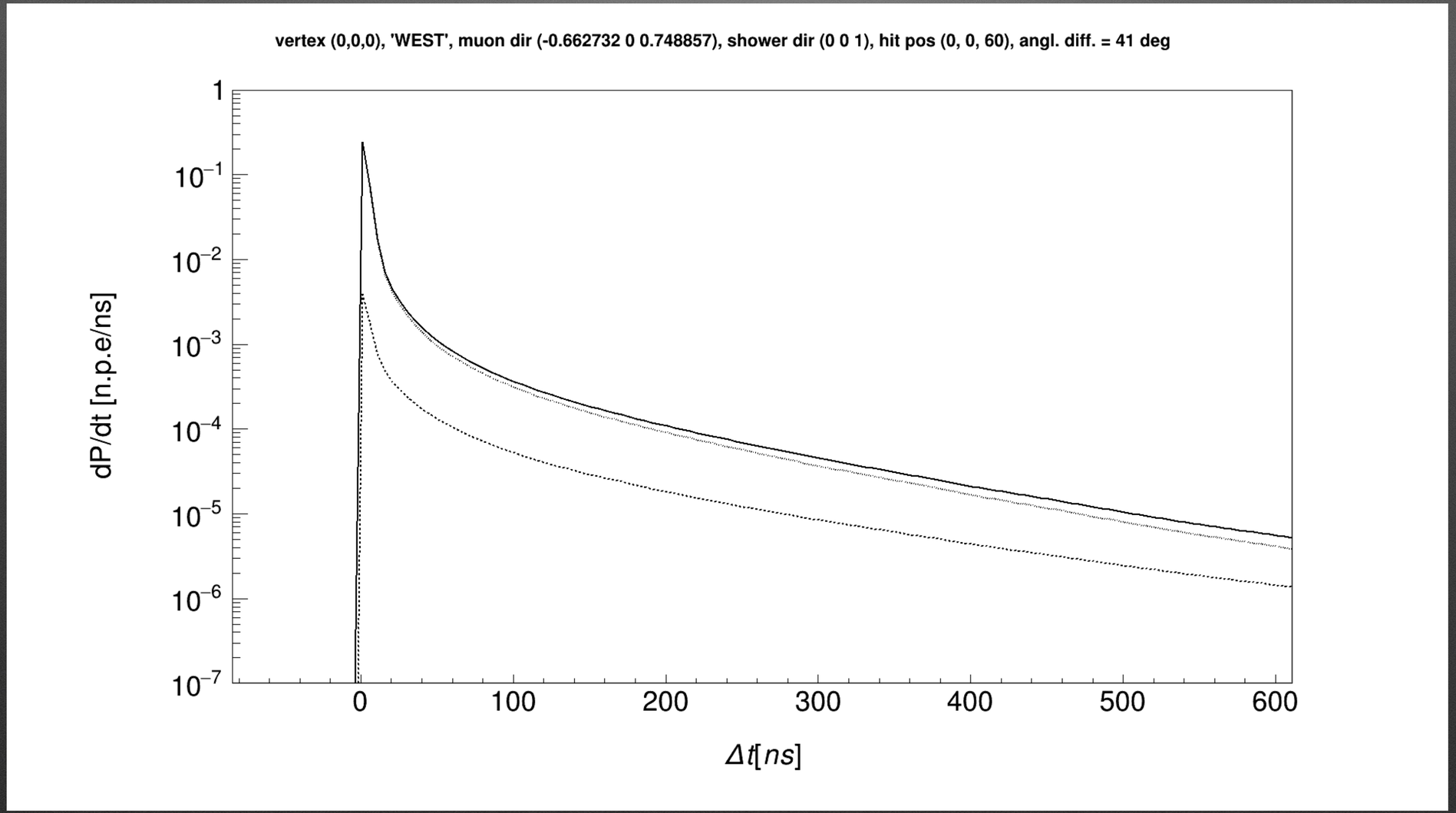
Combine PDFs

- Rotate the muon w.r.t shower and plot PDFs.
- Check: narrow angle, Cherenkov angle, wide angle between the two components
- NB: time residual on the plot is that w.r.t. muon hypothesis

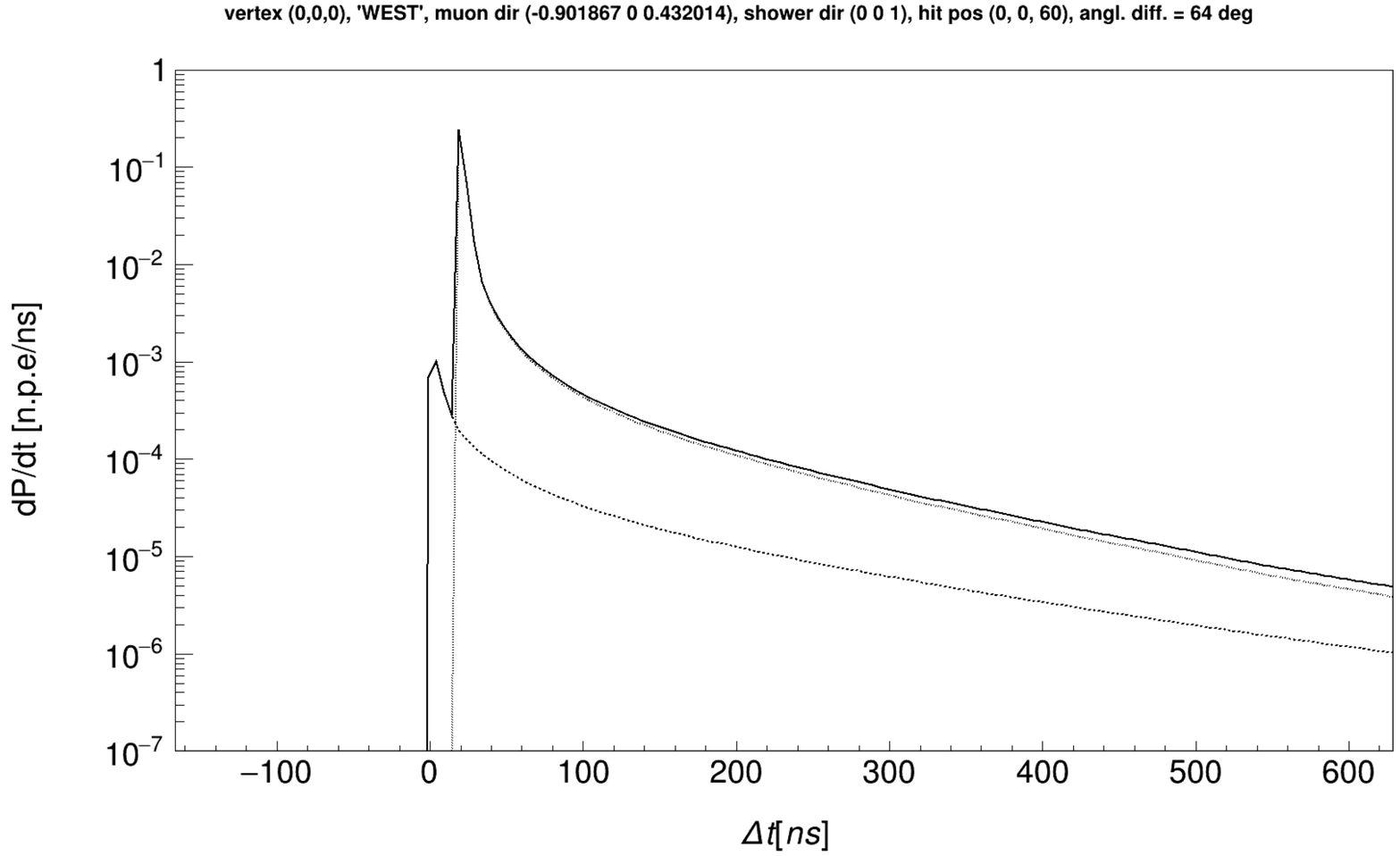
Combine PDFs



Combine PDFs



Combine PDFs

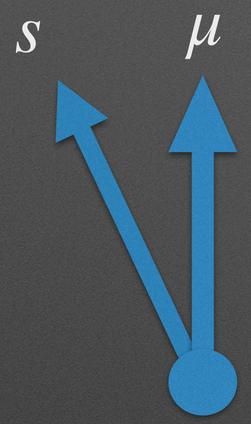


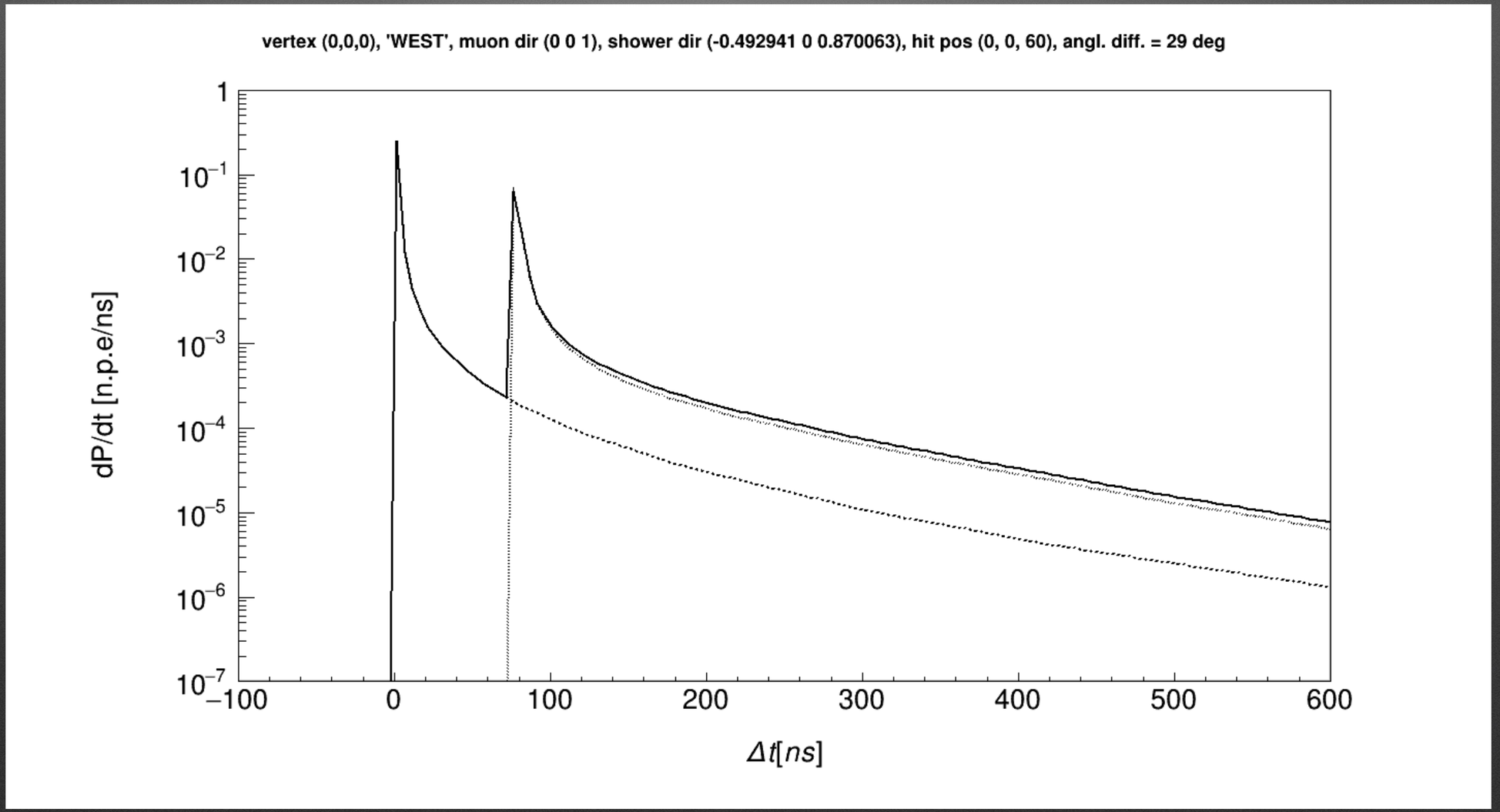
Combine PDFs

- Rotate the shower w.r.t muon and plot PDFs.
- Check: narrow angle, Cherenkov angle, wide angle between the two components
- NB: time residual on the plot is that w.r.t. muon hypothesis

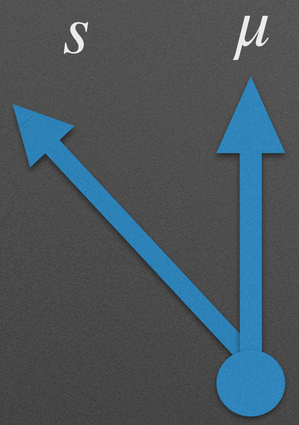
-> resolved issues

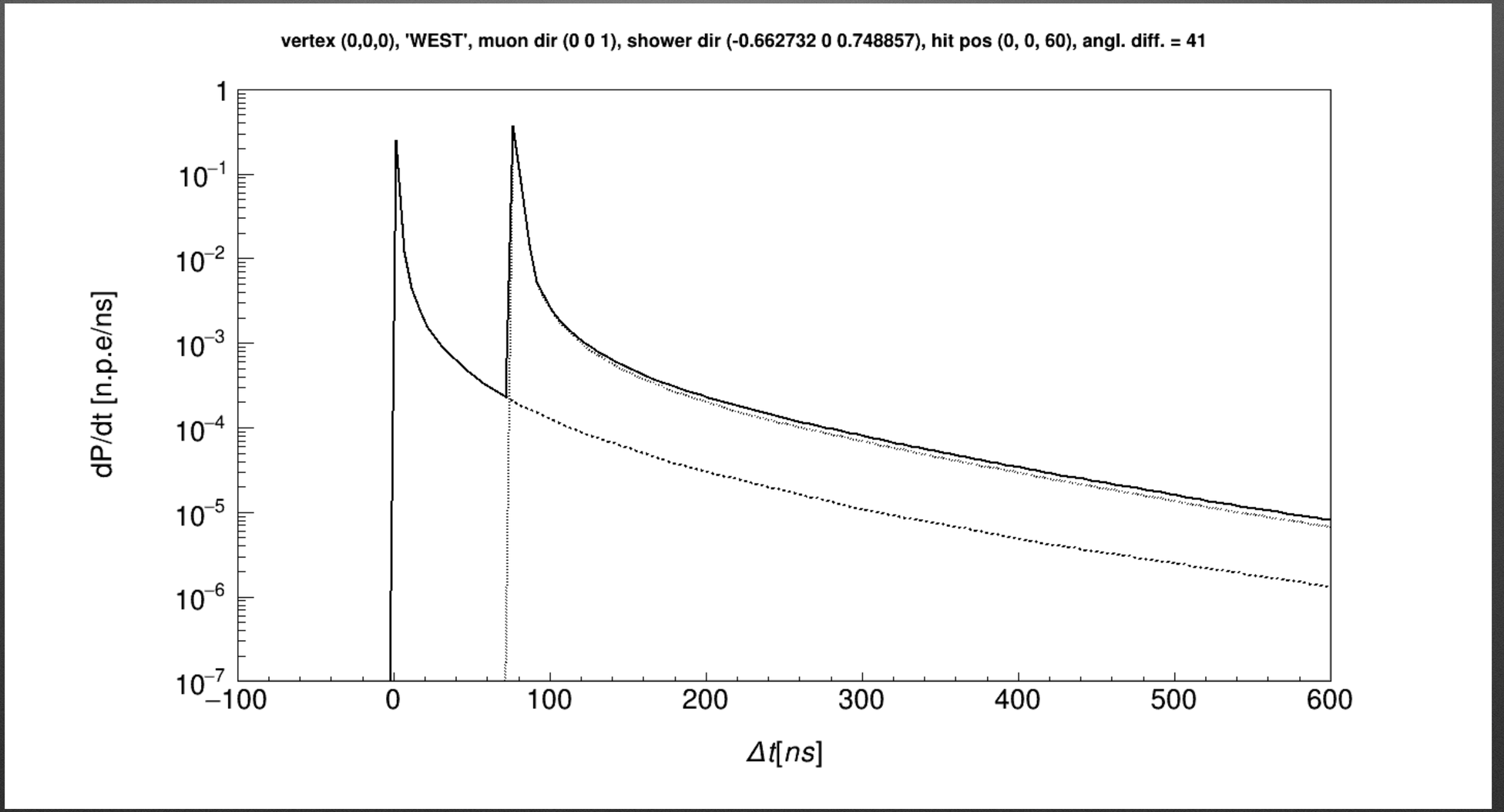
Combine PDFs



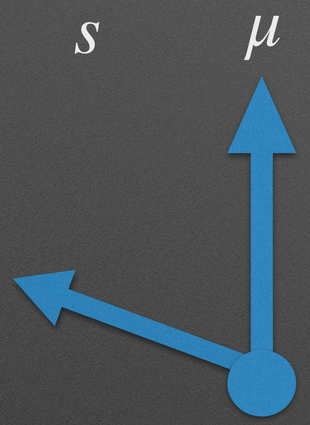


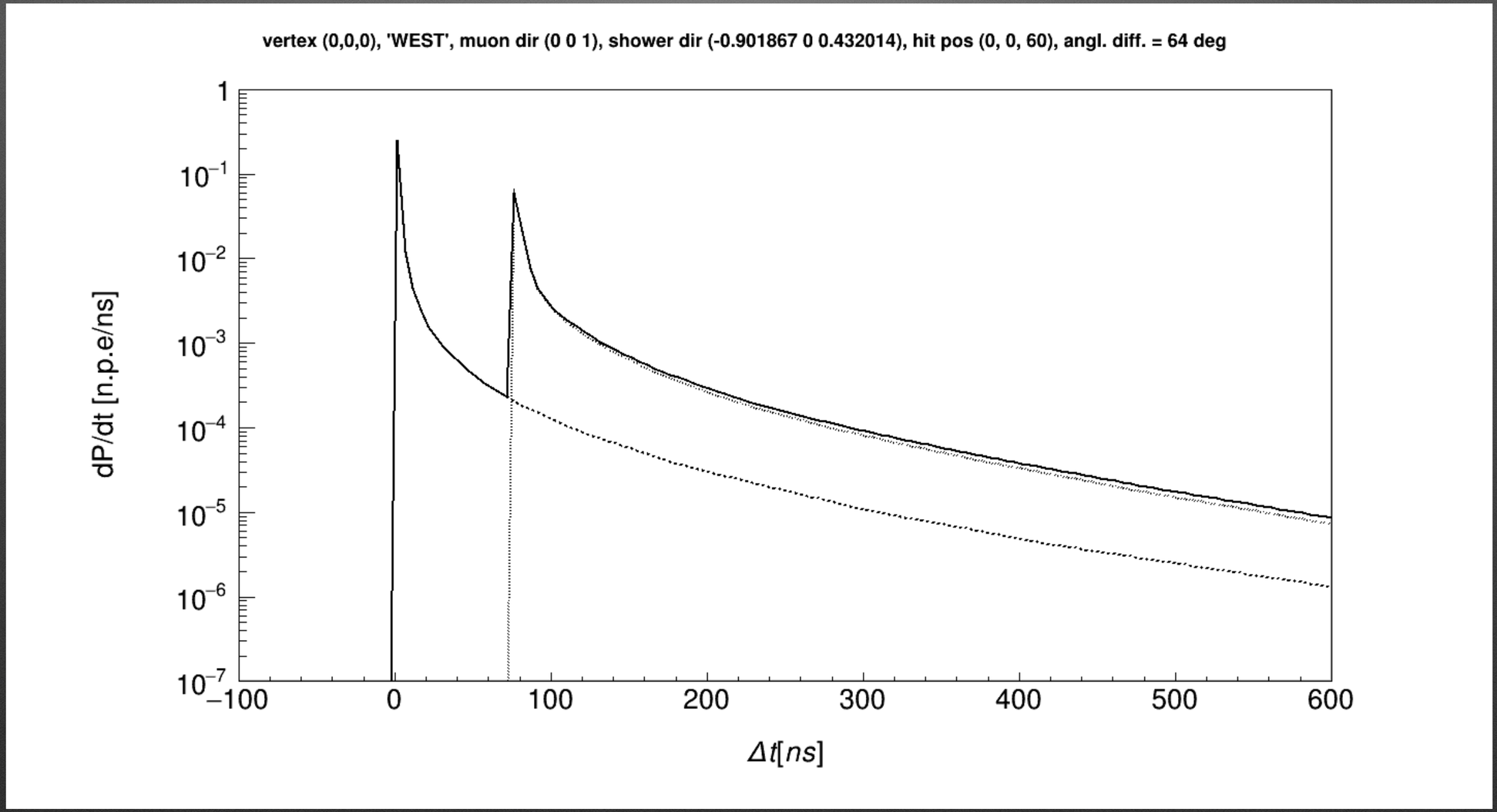
Combine PDFs



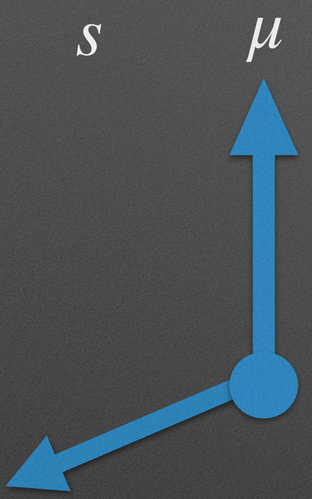


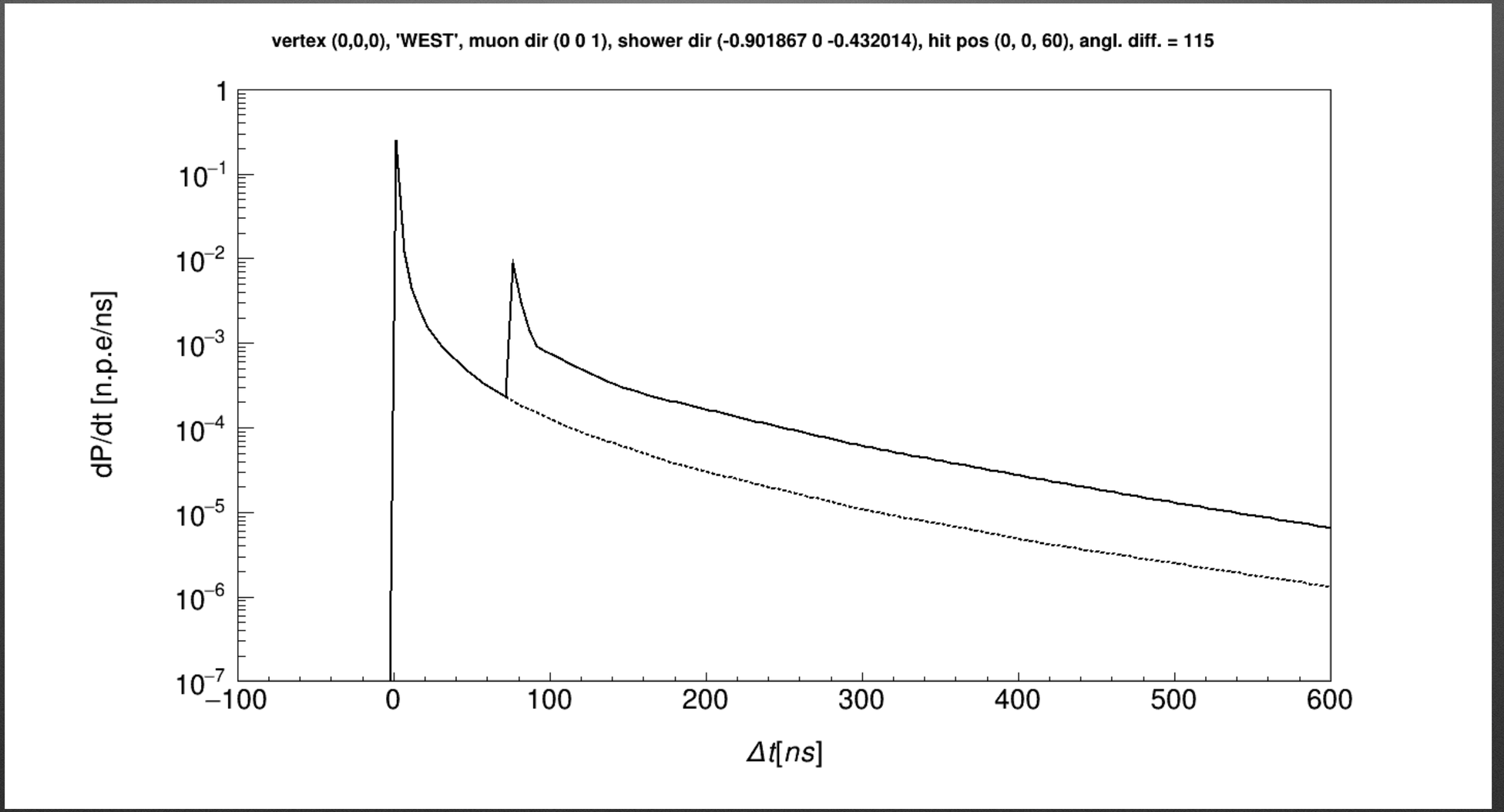
Combine PDFs





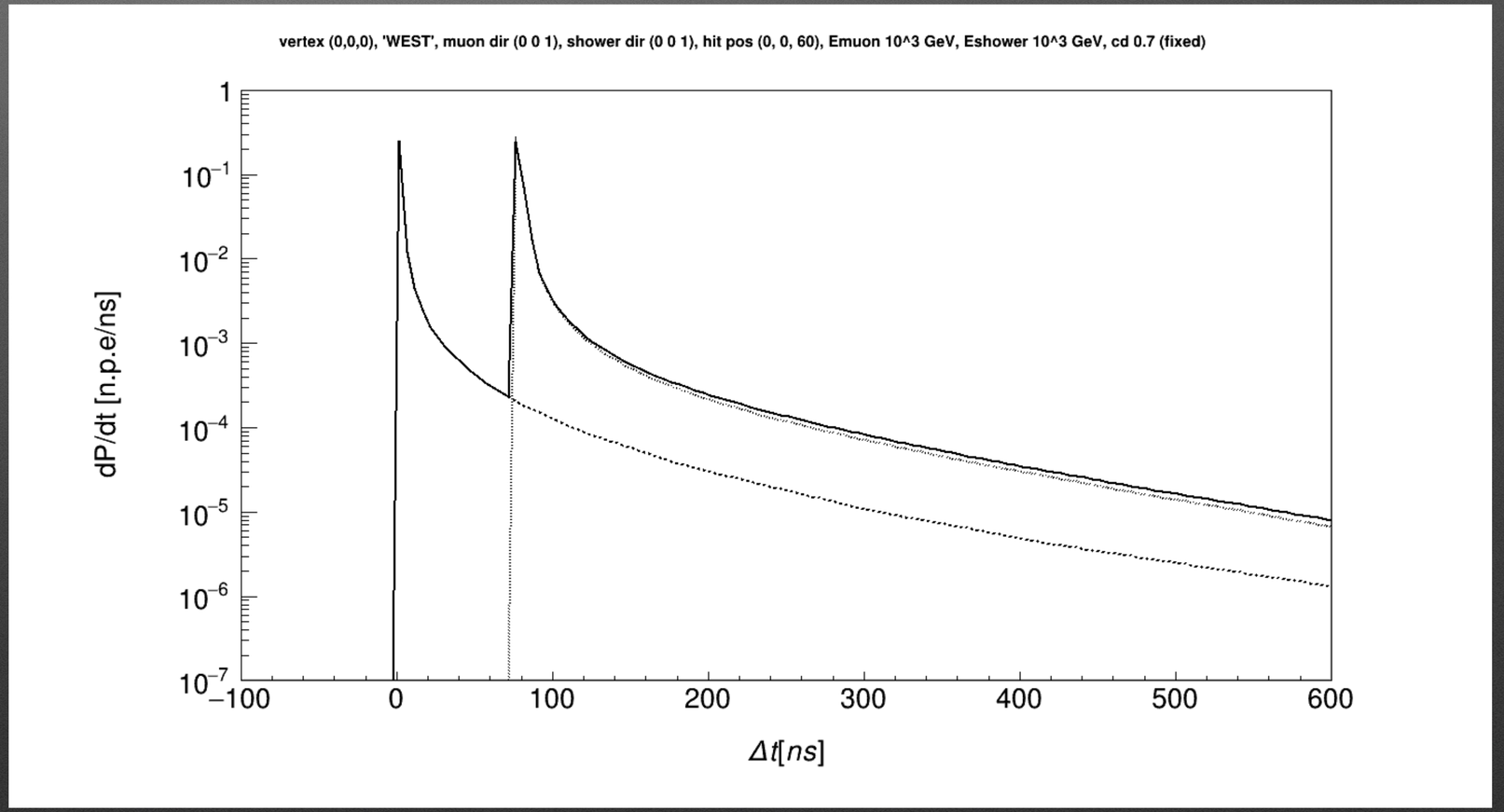
Combine PDFs





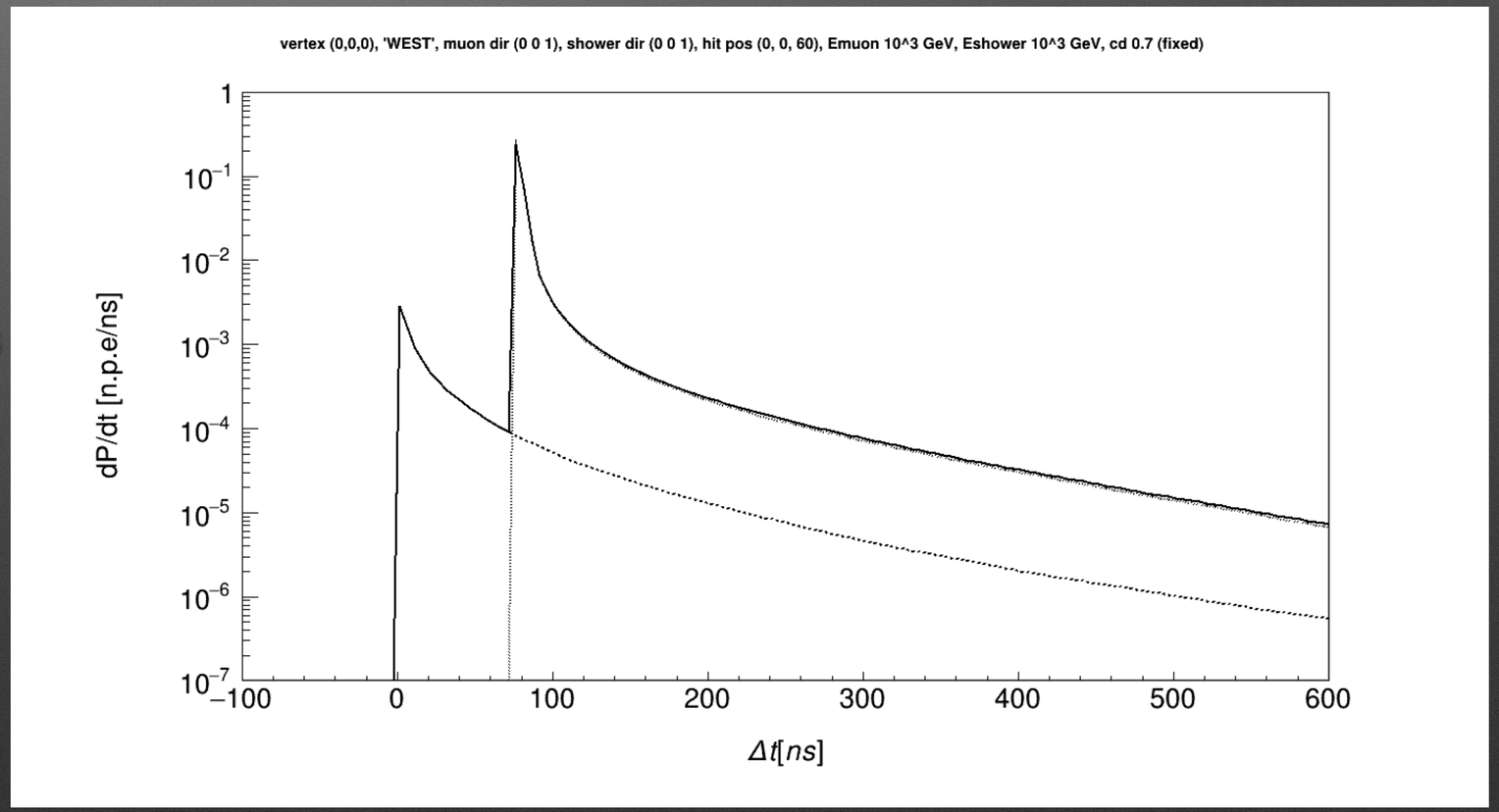
Combine PDFs

- Check E_mu / E_sh = 0 GeV cases
- Here, both at same energy, fixed direction



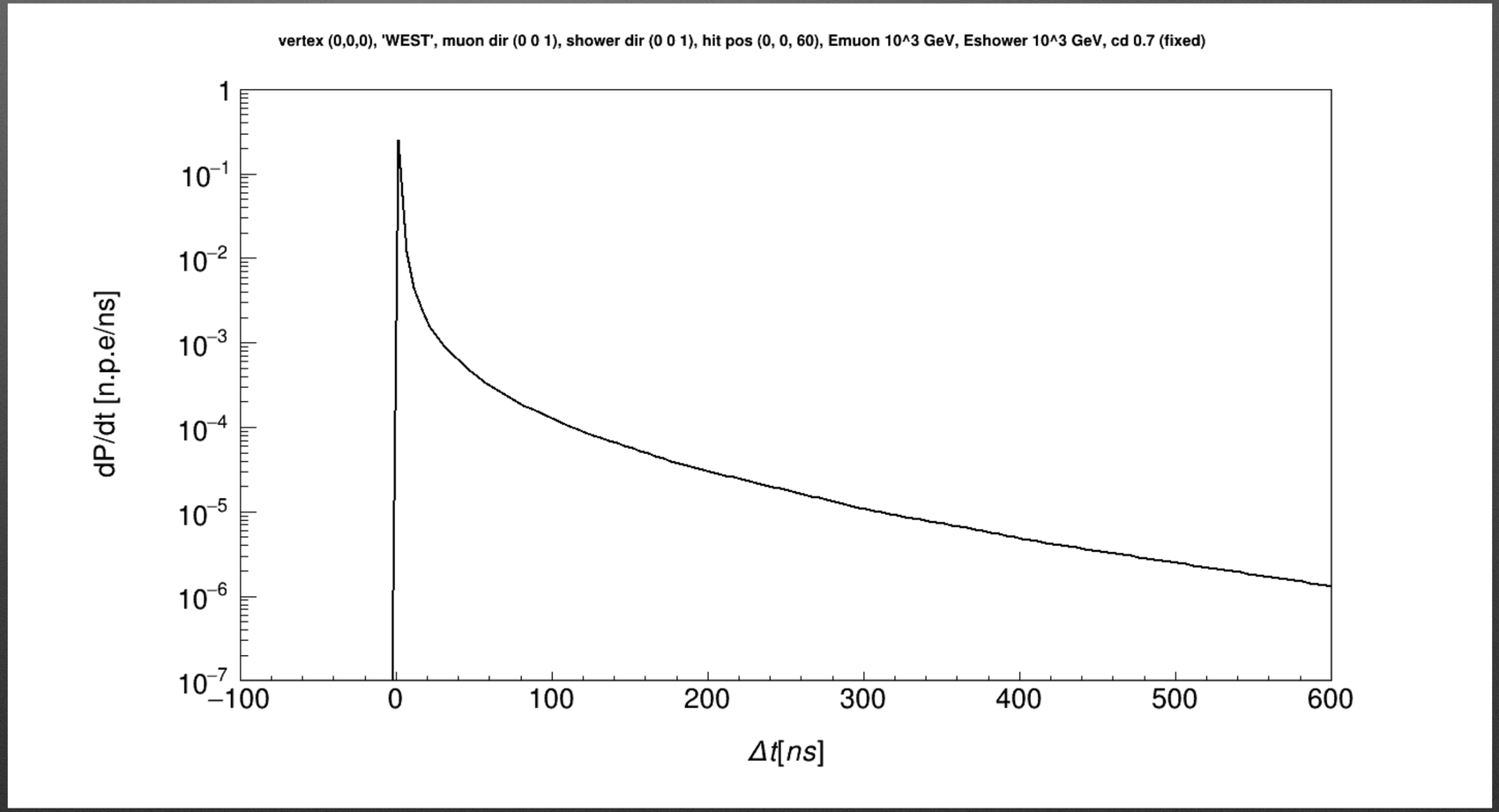
Combine PDFs

- E_mu = 0 GeV
- As Maarten said,
  need to check prior to evaluating the energy loss, whether a track particle is actually above the minimum ionization threshold or not.



Combine PDFs

- E_sh = 0 GeV - only see muon light



Combine PDFs

- Comparison using JDrawPDX tools
- Apply conditions of model to the PDFs - no muon light from behind the vertex, E threshold for a MIP… *even* in drawing stages
- Add PDF blurring, time-slewing correction
- LIKELIHOOD