University of Amsterdam

Nikhef

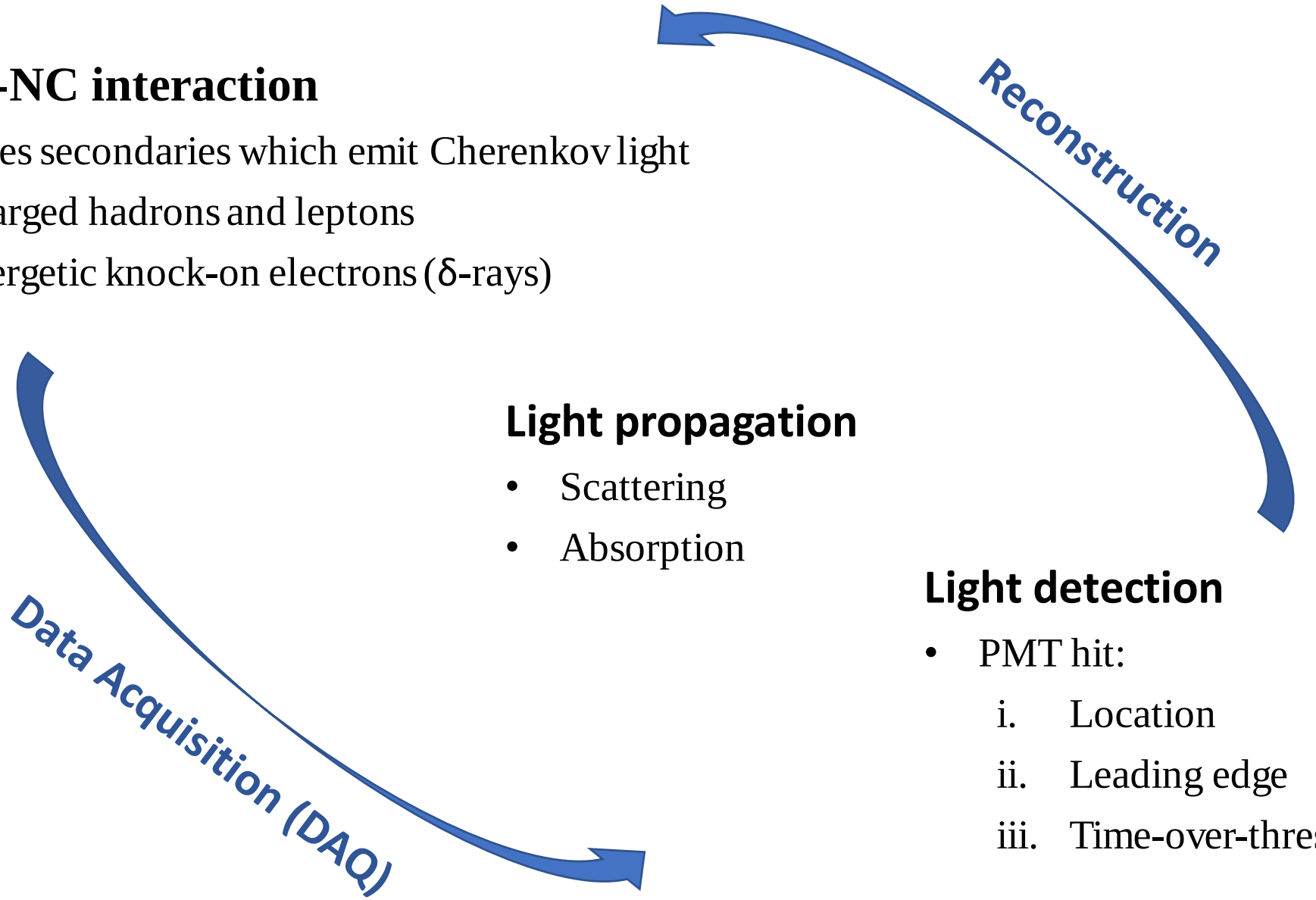
KM3NeT

Jpp Track Reconstruction

Introduction

ν-CC or -NC interaction

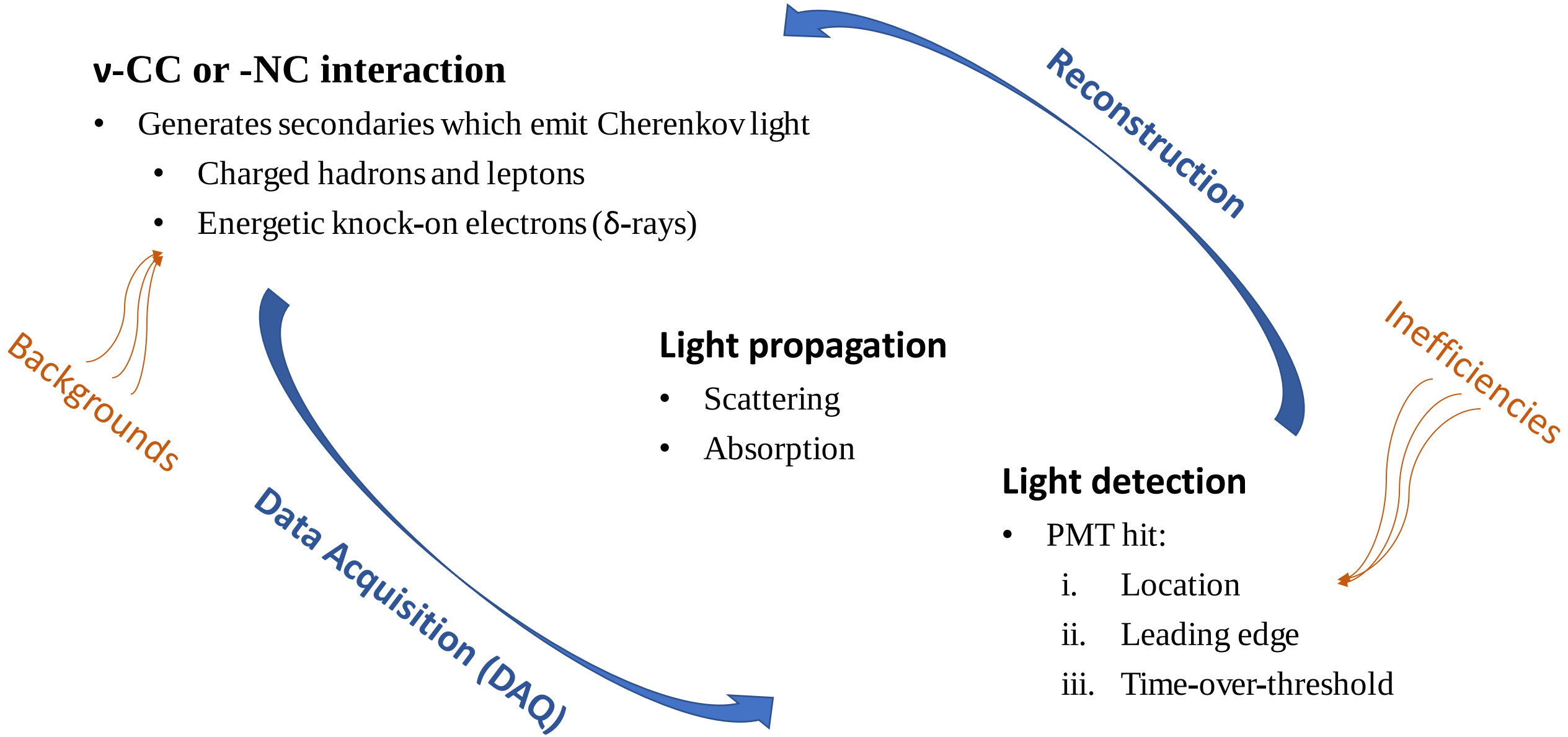
- Generates secondaries which emit Cherenkov light
  - Charged hadrons and leptons
  - Energetic knock-on electrons (δ-rays)



Introduction

ν-CC or -NC interaction

- Generates secondaries which emit Cherenkov light
  - Charged hadrons and leptons
  - Energetic knock-on electrons (δ-rays)



Reconstruction Procedure

- For each event, reconstruct:
  1. Interaction vertex
  2. Track direction
  3. Track energy
  4. Inelasticity

Spatial-temporal hit correlations

↓

Poisson hit probabilities

- Different reconstruction chains for different event topologies:
  1. Showers
     i. Hadronic
     ii. Electro-Magnetic
  2. Tracks
     i. Long-lived charged particles --> muons

JShower-chain

JMuon-chain

Track reconstruction

- The goal: description of muon-trajectory consistent with data

- 8 parameters:
  1. Direction (dx, dy, dz)
  2. Start (x_s, y_s, z_s, t_s)
  3. Initial energy (E_0)

JPrefit --> JSimplex --> JGandalf
(Grid scan --> 1st minimization --> 2nd minimization)

JStart (back-projection of PMT hits)

JEnergy (Inference from PMT hit statistics)

- Note that this optimization problem is non-linear in nature!

- JPrefit constitutes an initial, linearized regression,
  enabled by the initial direction input and the negligance of scattering

```
JFIT::JFit::JFit ( const JHistory &  history,
                   const double      x,
                   const double      y,
                   const double      z,
                   const double      dx,
                   const double      dy,
                   const double      dz,
                   const double      t,
                   const double      Q,
                   const int         NDF,
                   const double      E = 0.0,
                   const int         status = -1
                 )
```

Hit arrival times

- Consider a muon traveling in the +z direction, causing a hit in PMT j

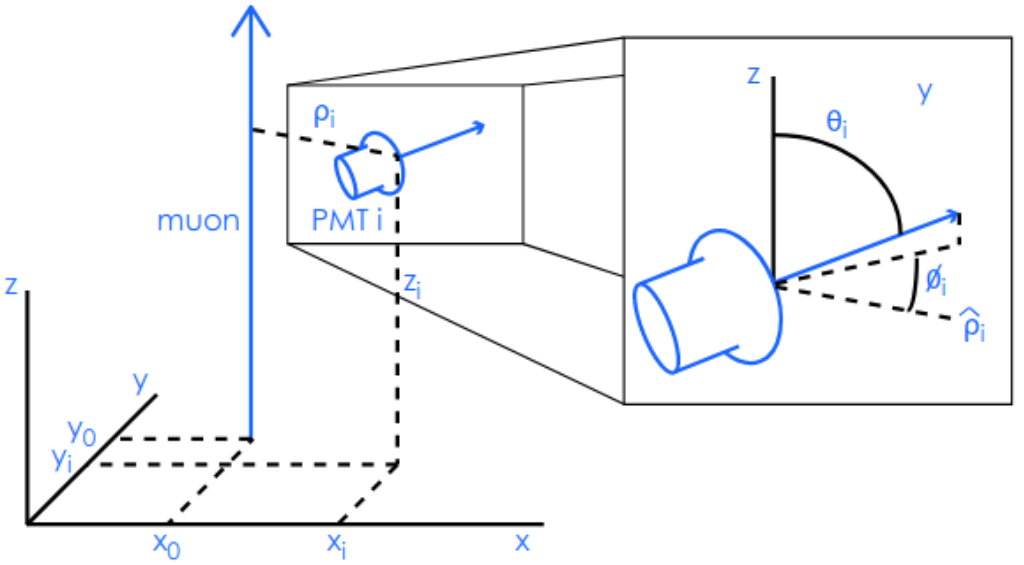
- Neglecting scattering, one would expect a hit at time:

Reference time *Photon propagation time*

$$t_j = t_0 + \frac{z_j}{c} + \tan(\theta_c)\frac{R_j}{c}$$

Muon propagation time

where t_0 defines the time at which the muon passes through the z=0 plane at coordinates (x_0, y_0)



Melis, K.; Heijboer, A.; De Jong, M. (PoS, 2017)

- Assuming we have a direction, we can exploit the arrival time correlations between the hits to find the reference point

JPrefit – Mathematical foundation

- Define the difference between two hit arrival times:

$$t_j'^2 - t_i'^2 - 2(t_j' - t_j')t_0' = x_j^2 - x_i^2 - 2(x_j - x_j)x_0 + y_j^2 - y_i^2 - 2(y_j - y_j)y_0 \,.$$

- Casting this into matrix form, $H\,\Theta = Y$:

$$H = \begin{pmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) & -2(t_2' - t_1') \\ 2(x_3 - x_2) & 2(y_3 - y_2) & -2(t_3' - t_2') \\ . & . & . \\ . & . & . \\ 2(x_1 - x_n) & 2(y_1 - y_n) & -2(t_1' - t_n') \end{pmatrix}, \qquad \Theta = \begin{pmatrix} x_0 \\ y_0 \\ t_0' \end{pmatrix} \longrightarrow \qquad Y = \begin{pmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 - t_2'^2 + t_1'^2 \\ x_3^2 - x_2^2 + y_3^2 - y_2^2 - t_3'^2 + t_2'^2 \\ . \\ . \\ x_1^2 - x_n^2 + y_1^2 - y_n^2 - t_1'^2 + t_n'^2 \end{pmatrix},$$

- To which the least squares solution is:

$$\Theta = \left(H^T V^{-1} H\right)^{-1} \times H^T V^{-1} \times Y$$

with V for the correlation matrix

$$V = J \times \begin{pmatrix} \sigma_1^2 & . & . & . & 0 \\ . & \sigma_2^2 & . & . & . \\ . & . & \sigma_3^2 & . & \\ . & . & . & . & . \\ 0 & . & . & . & \sigma_n^2 \end{pmatrix} \times J^T . \qquad J_{ij} = \delta Y_i \,/\, \delta t_j .$$

JPrefit - Code

- JPrefit takes L1 and/or L0 hit data
- Causally related hits, consistent with a passing muon, are selected among the data
- Using these hits, the ordinary least squares fit, is performed

JEvt JRECONSTRUCTION::JMuonPrefit::operator() (const buffer_type & dataL0,
const buffer_type & dataL1
)

Fit function.

Parameters

dataL0 L0 hit data

dataL1 L1 hit data

This is where the magic happens

```
284             try {
285
286               (*this)(data.begin(), __end1);
287
288               V.set(*this, data.begin(), __end1, gridAngle_deg, sigma_ns);
289               Y.set(*this, data.begin(), __end1);
290
291               V.invert();
292
293               double y = getChi2(Y, V);
```

JPrefit - Code

- This part of the code is responsible for calculating the covariance matrix of the hit time residuals

- Maybe merits enhanced documentation
  - What is the "kappa of Cherenkov light"?
  - What are v and w?
  - How does the grid angle (alpha) enter the fray? What's the geometrical picture?
  - How does this all translate back to the previous maths?

```
90      {
91        using namespace std;
92        using namespace JTOOLS;
93
94
95        const int N = distance(__begin, __end);
96
97        this ->resize(N);
98        buffer.resize(N);
99
100       const double ta = alpha * PI / 180.0;
101       const double ct = cos(ta);
102       const double st = sin(ta);
103
104
105       // angular resolution
106
107       for (T i = __begin; i != __end; ++i) {
108
109         const double dx = i->getX() - pos.getX();
110         const double dy = i->getY() - pos.getY();
111         const double dz = i->getZ() - pos.getZ();
112
113         const double R  = sqrt(dx*dx + dy*dy);
114
115         double x  = ta * getKappaC() * getInverseSpeedO
116         double y  = ta * getKappaC() * getInverseSpeedO
117         double v  = ta * getInverseSpeedOfLight();
118         double w  = ta * getInverseSpeedOfLight();
119
120         if (R != 0.0) {
121           x *= dx / R;
122           y *= dy / R;
123         }
124
125         x *=  (dz*ct - dx*st);
126         y *=  (dz*ct - dy*st);
127         v *= -(dx*ct + dz*st);
128         w *= -(dy*ct + dz*st);
129
130         buffer[distance(__begin,i)] = variance(x, y, v, w);
131       }
132
133       for (int i = 0; i != N; ++i) {
134
135         for (int j = 0; j != i; ++j) {
136           (*this)(i, j) = getDot(buffer[i], buffer[j]);
137           (*this)(j, i) = (*this)(i, j);
138         }
139
140         (*this)(i, i) = getDot(buffer[i], buffer[i]) +  sigma * sigma;
141       }
142     }
```

$$\kappa = \frac{c}{v_g}\frac{1}{\sin\theta_c} - \frac{1}{\tan\theta_c}$$

double JTOOLS::getKappaC ( )

Get average kappa of Cherenkov light in water.

Returns

kappa

Definition at line 166 of file JConstants.hh.

```
167   {
168     return KAPPA_WATER;
169   }
```

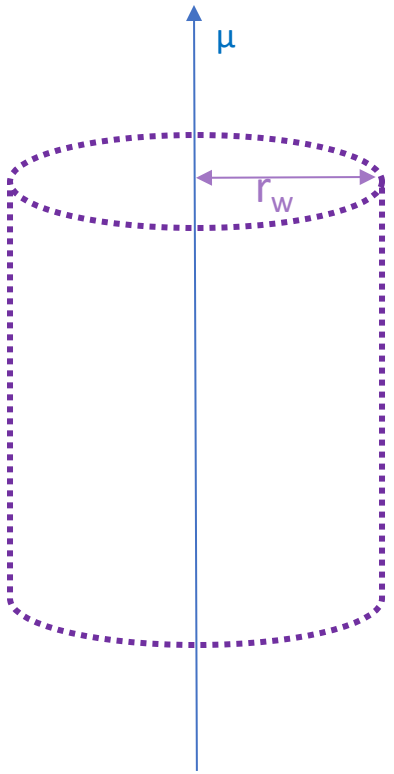
JPrefit

- The prefits for each directional assumption are ranked based on increasing quality:

$$Q = \mathrm{NDF} - \frac{1}{4}\frac{\chi^2}{\mathrm{NDF}} \longrightarrow \chi^2 = \sum_i \frac{\left(t_i - \hat{t}_i\right)^2}{\sigma_i^2} = \mathrm{R}^{\mathrm{T}}\mathrm{V}^{-1}\mathrm{R}$$

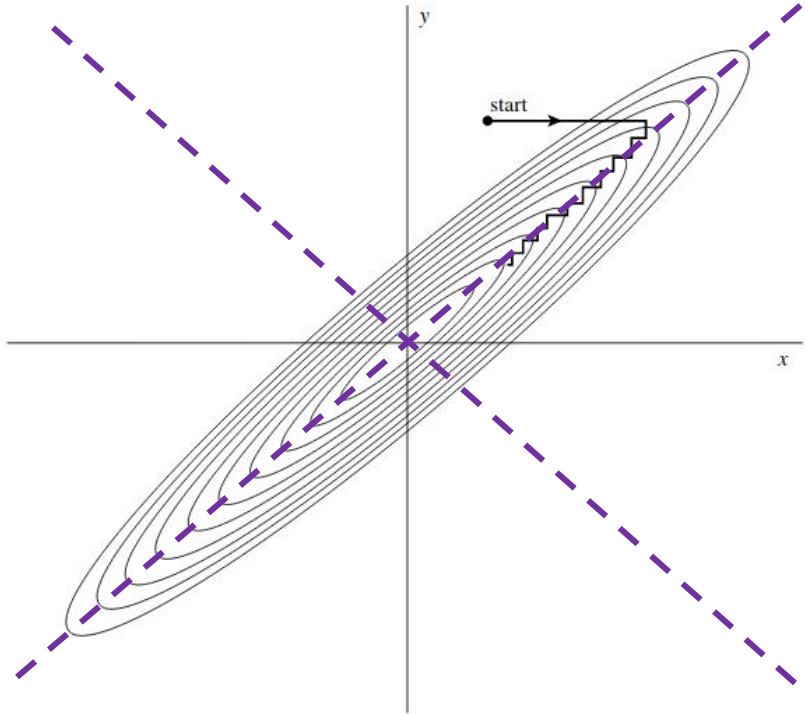
- **Number of degrees** of freedom scales with the number of hits, which is dependent on the **roadwidth**

- Of all the evaluated directions, the best N (12 default) are stored and passed on to JSimplex



JMuonSimplex

- JMuonSimplex minimizes the arrival time residuals, via a minimization algorithm called **Powell's method:**
  - Iteratively finds a set of N mutually **conjugate directions**, whith which to minimize your likelihood landscape
  - Does not require any gradients!
- The conjugate directions for JMuonSimplex are embedded in the position-direction parameter space (x_0, y_0, z_0, t_0, dx, dy, dz)
- For each step, the chi2 is re-evaluated
- Terminate when fractional improvement becomes negligible (default 1e-4)



10.7.2 Powell's Quadratically Convergent Method

Powell first discovered a direction set method that does produce N mutually conjugate directions. Here is how it goes: Initialize the set of directions $\mathbf{u}_i$ to the basis vectors,

$$\mathbf{u}_i = \mathbf{e}_i \qquad i = 0, \ldots, N-1 \tag{10.7.6}$$

Now repeat the following sequence of steps ("basic procedure") until your function stops decreasing:

- Save your starting position as $\mathbf{P}_0$.
- For $i = 0, \ldots, N-1$, move $\mathbf{P}_i$ to the minimum along direction $\mathbf{u}_i$ and call this point $\mathbf{P}_{i+1}$.
- For $i = 0, \ldots, N-2$, set $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$.
- Set $\mathbf{u}_{N-1} \leftarrow \mathbf{P}_N - \mathbf{P}_0$.
- Move $\mathbf{P}_N$ to the minimum along direction $\mathbf{u}_{N-1}$ and call this point $\mathbf{P}_0$.

Powell, in 1964, showed that, for a quadratic form like (10.7.1), k iterations of the above basic procedure produce a set of directions $\mathbf{u}_i$ whose last k members are mutually conjugate. Therefore, N iterations of the basic procedure, amounting to

Press, WH et al. (2007); Numerical Recipes: The Art of Scientific Computing (3rd ed.)

JMuonGandalf

- The fits from JPrefit and JSimplex should now be sufficiently close to the true optimum, to allow for a final, full MLE, including non-linear effects such as scattering

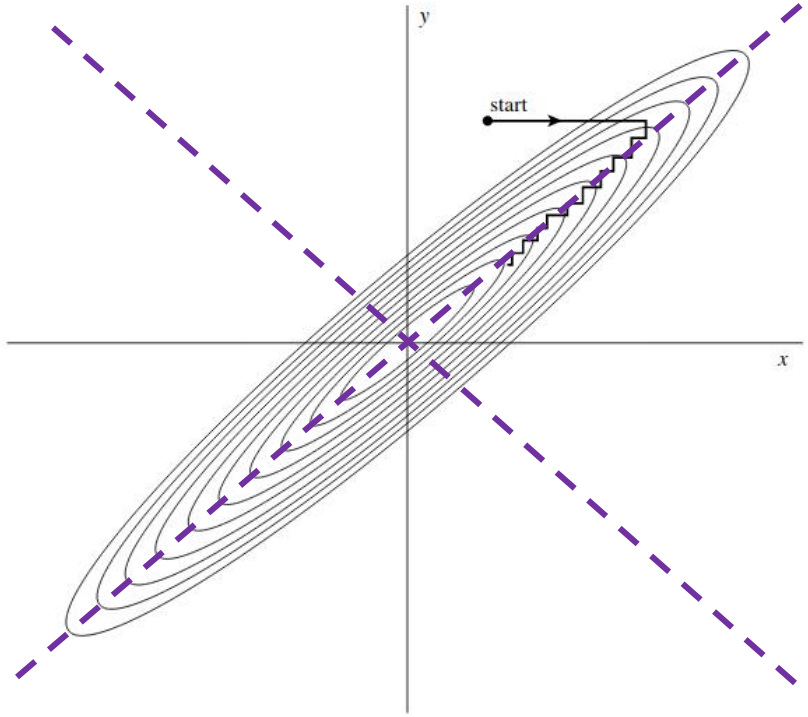
- JMuonGandalf constitutes this final minimization step

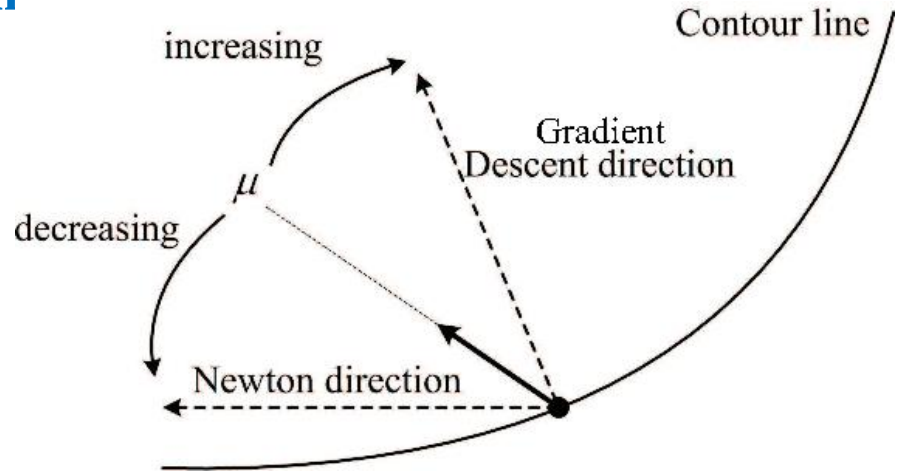
  - Exploits the analytical arrival time PDFs encoded in Jpp

- The method is based on the **Levenberg-Marquard Algorithm**

  - Exploits the strengths of both gradient descent and Gauss-Newton minimization

  - Requires analytical calculation of the Hessian matrix





Press, WH et al. (2007); Numerical Recipes: The Art of Scientific Computing (3rd ed.)

Arrival Time PDFs

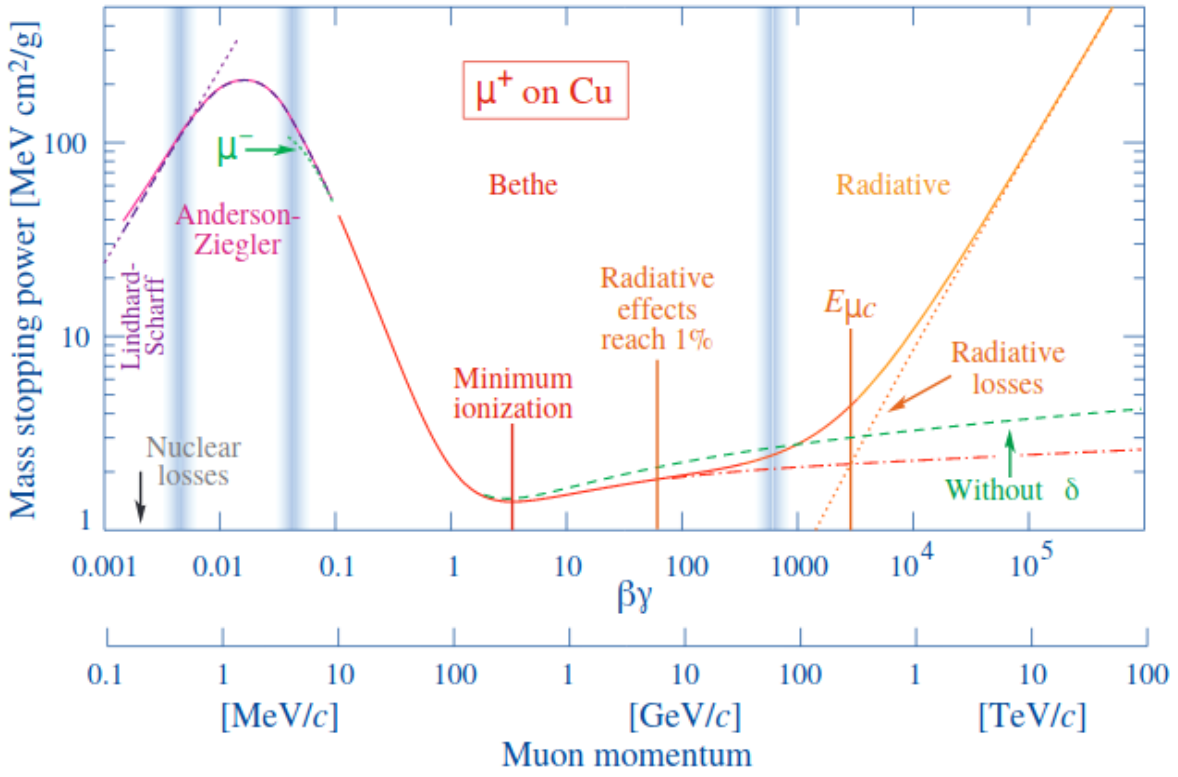
- Three main ingredients:
  1. Flux per unit wavelength as a function of closest distance of approach, R
  2. Light propagation (scattering, absorption, dispersion)
  3. Detector response (effective photocathode area, quantum efficiency)

$$\frac{d\mathcal{P}}{dt} = \Phi_0(R,\lambda)\, A \left(\frac{\partial t}{\partial \lambda}\right)^{-1} \varepsilon(\cos\theta_{\oslash})\, QE(\lambda)\, e^{-d/\lambda_{abs}}\, e^{-d/\lambda_s}$$

- For muons, the photon flux comprises:
  - Direct Cherenkov radiation
  - Ionization photons
  - Radiative effects (bremsstrahlung)

- Additionally there is light from energetic knock-on electrons (delta-rays)

Particle Data Group, Passage of particles through matter (2018)



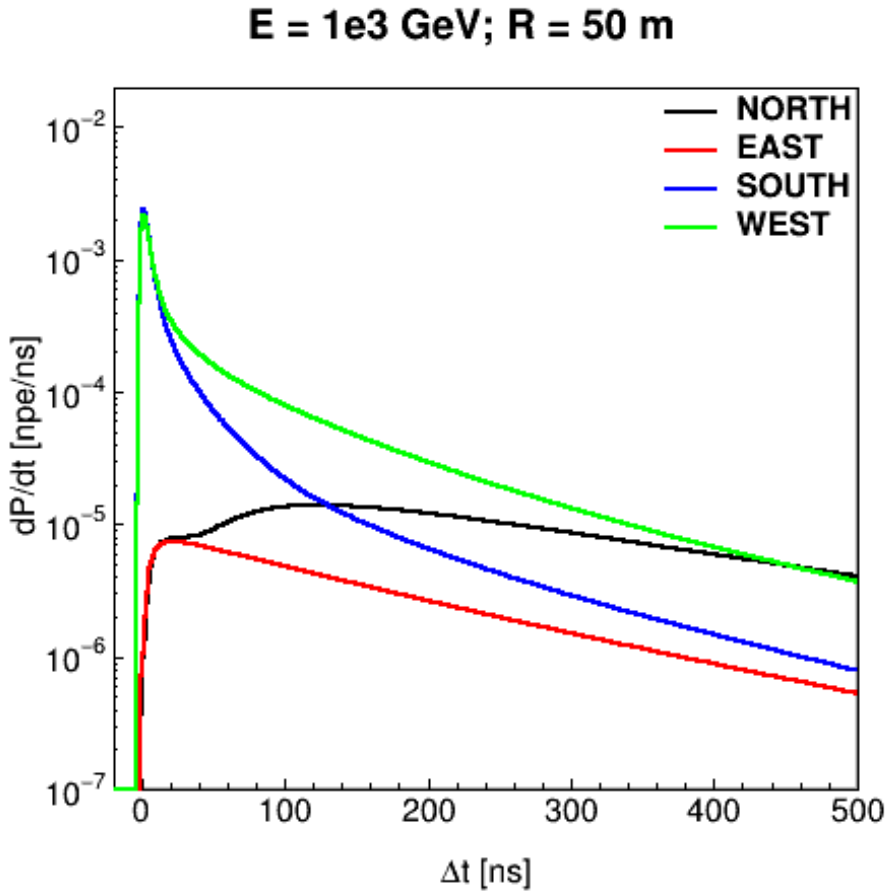
Arrival Time PDFs

- Three main ingredients:
  1. Flux per unit wavelength as a function of closest distance of approach, R
  2. Light propagation (scattering, absorption, dispersion)
  3. Detector response (effective photocathode area, quantum efficiency)

$$\frac{d\mathcal{P}}{dt} = \Phi_0(R,\lambda)\, A \left(\frac{\partial t}{\partial \lambda}\right)^{-1} \varepsilon(\cos\theta_{\odot})\, QE(\lambda)\, e^{-d/\lambda_{abs}}\, e^{-d/\lambda_s}$$

- For muons, the photon flux comprises:
  - Direct Cherenkov radiation
  - Ionization photons
  - Radiative effects (bremsstrahlung)

- Additionally there is light from energetic knock-on electrons (delta-rays)



JStart

- Takes the causally related hits and back-projects them onto the muon track under the Cherenkov angle

- The first (last) projection point corresponding to a hit, whose hit probability exceeds the background is determined to correspond to the start (end)

```
if (!data.empty()) {

  sort(data.begin(), data.end(), make_comparator(&JNPEHit::getZ));

  vector<JNPEHit>::const_iterator         track_start = start.find(data. begin(), data. end());
  vector<JNPEHit>::const_reverse_iterator track_end   = start.find(data.rbegin(), data.rend());

  if (track_start == data. end()) { track_start = data. begin(); }
  if (track_end   == data.rend()) { track_end   = data.rbegin(); }
```

JEnergy

- The energy estimation is based on a binomial hit/no-hit likelihood estimation

  - I.e.: If a hit was registered at PMT i, what is probability for this hit having been caused by the traversal of a muon of initial energy E, starting at position given by JStart and with direction given by JGandalf?

- At > GeV energies, the energy deposition of a muon can be approximated well using two terms, one of which encodes **ionization energy loss**, the other of which describes **radiative energy loss**

$$-dE/dx = a(E) + b(E)\,E\,.$$

  - a and b vary only very slowly as function of energy and can be assumed constant, to good approximation

- Given the distance of a PMT to the muon track, R, as well as PMT looking-angles, θ and ϕ, the total energy loss of the muon can be converted into an expected number of hits, λ, using the integrated arrival time PDFs

JEnergy

- The energy estimation is based on a binomial hit/no-hit lik

  - I.e.: If a hit was registered at PMT i, what is probabili
    having been caused by the traversal of a muon of initi
    starting at position given by JStart and with direction

- At > GeV energies, the energy deposition of a muon can b h
  encodes **ionization energy loss**, the other of which describ

$$-dE/dx = a(E) + b(E)\,E\,.$$

  - a and b vary only very slowly as function of energy and can be assumed constant, to good approximation

- Given the distance of a PMT to the muon track, R, as well as PMT looking-angles, θ and ϕ,
  the total energy loss of the muon can be converted into an expected number of hits, λ,
  using the integrated arrival time PDFs

double JFIT::JNPE::getH1 (const double E_GeV) const

Expected number of photo-electrons for muon hypothesis as a function of muon energy.

Parameters

E_GeV energy [GeV]

Returns

light yield [npe]

Definition at line 126 of file JNPE.hh.

```
127     {
128       using namespace JPP;
129
130       const double E = gWater.getE(E_GeV, this->getZ());
131
132       if (E >= MASS_MUON * INDEX_OF_REFRACTION_WATER)
133         return this->getYA() + E * this->getYB();
134       else
135         return 0.0;
136     }
```

JEnergy

- Given the expected number of photon hits, the probability for a no-hit or a hit follows as:

$$P(0) = e^{-\lambda}$$
$$P(n > 0) = 1 - e^{-\lambda}$$

- JEnergy computes the logarithm of this quantity for each PMT

- A chi2-equivalent is defined as the quadratic sum of these logarithmic probabilities

$$\chi^2 = \frac{1}{2} \sum_i \ln^2 [P_i]$$

- This quantity is minimized as a function of the energy,
  using an iterative 5-point search between an upper and a lower limit

JEnergy

- Given the expected number of photon hits, the probability for a no-hit or a

$$P(0) = e^{-\lambda}$$
$$P(n > 0) = 1 - e^{-\lambda}$$

- JEnergy computes the logarithm of this quantity for each PMT

- A chi2-equivalent is defined as the quadratic sum of these logarithmic prob

$$\chi^2 = \frac{1}{2} \sum_i \ln^2 [P_i]$$

- This quantity is minimized as a function of the energy,
  using an iterative 5-point search between an upper and a lower limit

```
// 5-point search between given limits

const int      N  =  5;

JResult result[N];

for (int i = 0; i != N; ++i) {
  result[i].x = EMin_log + i * (EMax_log - EMin_log) / (N-1);
}

do {

  int j = 0;

  for (int i = 0; i != N; ++i) {

    if (!result[i]) {
      result[i].chi2 = fit(result[i].x, data.begin(), data.end());
    }

    if (result[i].chi2 < result[j].chi2) {
      j = i;
    }
  }

  // squeeze range

  switch (j) {

  case 0:
    result[4] = result[1];
    result[2] = JResult(0.5 * (result[0].x + result[4].x));
    break;

  case 1:
    result[4] = result[2];
    result[2] = result[1];
    break;

  case 2:
    result[0] = result[1];
    result[4] = result[3];
    break;

  case 3:
    result[0] = result[2];
    result[2] = result[3];
    break;

  case 4:
    result[0] = result[3];
    result[2] = JResult(0.5 * (result[0].x + result[4].x));
    break;
  }

  result[1] = JResult(0.5 * (result[0].x + result[2].x));
  result[3] = JResult(0.5 * (result[2].x + result[4].x));

} while (result[4].x - result[0].x > parameters.resolution);
```

Questions

- Do no-hits not contain information about the muon direction / track lenght as well?
- What extra information do the time-over-threshold values yield about the muon energy?
- How does the light yield from the hadronic final state influence the reconstructed muon energy?