The cluster uncertainties

Mikael Berggren[1]

[1]DESY, Hamburg

ILD phone meeting, 28 Oct 2015

Outline

1. Introduction
2. Error propagation
3. Checking the math
4. Classes and processors
5. Conclusions

Introduction

The problem:

- Clusters are measured
- Hence, their properties should be assigned errors
- Specifically, they have a
  - Total magnitude (ie. Energy)
  - Position
  - Direction
- The error on the energy will come from parametrisation of simulation and test-beam observations.
- The Position and Direction error - in contrast - are measurable in each cluster

Cluster direction

The problem:

- A Cluster is a set of points in space (possibly with an attached weight).
- The direction of the cluster - or the cluster axis - would naturally be the direction that minimises the distance between the axis and the points.
- This can be shown to be the eigen-vector corresponding to the largest eigen-value of the estimated covariance matrix C^*. (This is *non-trivial* to prove !!!)
- It can also be shown that the covariance matrix and the tensor of inertia have the same eigen-vectors, but eigen-values in the opposite order (This is *a bit less non-trivial* to prove !!!)
- Also: $C^*/"n"$ *is* the correlation matrix of the position of the cluster. ($1/"n"$ because it's a bit more complicated than just that for weighted points...)

Cluster direction

The problem:

- A Cluster is a set of points in space (possibly with an attached weight).
- The direction of the cluster - or the cluster axis - would naturally be the direction that minimises the distance between the axis and the points.
- This can be shown to be the eigen-vector corresponding to the largest eigen-value of the estimated covariance matrix C^*. (This is *non-trivial* to prove !!!)
- It can also be shown that the covariance matrix and the tensor of inertia have the same eigen-vectors, but eigen-values in the opposite order (This is *a bit less non-trivial* to prove !!!)
- Also: $C^*/"n"$ *is* the correlation matrix of the position of the cluster. ($1/"n"$ because it's a bit more complicated than just that for weighted points...)

Error propagation

The problem:

- In general: $V = \Delta C' \Delta^T$ where Δ is the derivatives of the variables V pertains to wrt those actually measured. C' is the covariance between the measurements.
- In our case: Δ are the derivatives of θ and ϕ wrt. the *elements* of C^*.
- C' is in our case the "covariance-of-covariance", denoted by ζ.
- C^* is a symmetric 3×3 matrix, ie. it has 6 unique elements.
- ζ is thus 6×6, still symmetric $\Rightarrow$ 21 unique elements.
- Δ is 6×2, so that V indeed is 2×2

Error propagation: Δ

Each row i of Δ is

$$\begin{pmatrix} \frac{\cos\psi}{\sin\theta(\lambda-\lambda_2)}\bar{u}_2^T\delta C_i\bar{u} - \frac{\sin\psi}{\sin\theta(\lambda-\lambda_3)}\bar{u}_3^T\delta C_i\bar{u} & \ldots \\ \frac{\sin\psi}{(\lambda-\lambda_2)}\bar{u}_2^T\delta C_i\bar{u} + \frac{\cos\psi}{(\lambda-\lambda_3)}\bar{u}_3^T\delta C_i\bar{u} & \ldots \end{pmatrix}$$

which is *highly non-trivial* to show.
$\bar{u}$ and λ is the main axis and the corresponding eigen-value, while $\bar{u}_{2,3}$ and $\lambda_{2,3}$ are the other two eigen-vectors/values.
At least, δC_i, which is the derivative of C^* wrt. each of the six unique elements of C^* *is* is trivial = 3$\times$3 matrices with one or two “1”:s, the rest are “0”:es....

Error propagation: ζ

The 21 elements of ζ:

- These are the variances of the estimate of each of the six unique elements of C^* along the diagonal.
- The off-diagonal elements contains things like $Cov(V^*_x, C^*_{yz})$ etc.
- Under the assumption that different points in the cluster are independent (ie. no cross-talk), this can be calculated.
- The elements of ζ will be combinations of different elements of C^* with each other, and with all possible fourth-moments of the distribution of points (M_4).

Error propagation: ζ

In the "simple" case with un-weighed points:

$$\tilde{\zeta}_{ij} = (C_{ind(i,1),ind(j,1)}C_{ind(i,2),ind(j,2)} + C_{ind(i,1),ind(j,2)}C_{ind(i,2),ind(j,1)})\frac{1}{n-1}$$

where $\tilde{\zeta}$ is the approximation of ζ when the fourth moments are not available (= the exact answer if the distribution is a 3D Gaussian). If they are, then

$$\zeta_{ij} = \frac{1}{n}\left(\tilde{\zeta}_{ij} + M_{ind(i,1),ind(i,2),ind(j,1),ind(j,2)} - C_{ind(i,1),ind(i,2)}C_{ind(j,1),ind(j,2)}\right)$$

The index array (*ind*) is

$$ind^T = \begin{pmatrix} x & y & z & x & x & y \\ x & y & z & y & z & z \end{pmatrix}$$

Error propagation: Weights

In fact, the points have weights. Weights can be of two types:

- Importance weights.
- Sampling weights.

The latter $\Leftrightarrow$ points are entries in a 3D histogram doesn't change anything, except that the bin-width uncertainty needs to be added in quadrature when looking at pulls.
In the former case, the weights must be tracked through the calculations.

Error propagation: Weights

The result:

$$C^*_{ij} = \frac{\Sigma(\xi^{(i)} - \langle \xi^{(i)} \rangle)(\xi^{(j)} - \langle \xi^{(j)} \rangle) w_i}{f_1(n-1)}$$

$$\zeta_{mnrv} = \frac{(f_4 + \frac{f_3 - f_4}{n-1})(M_{mnrv} - C_{mn}C_{rv}) + f_3\,(C_{mr}C_{nv} + C_{mv}C_{nr})\,\frac{1}{(n-1)}}{n(f_1 f_2)^2}$$

$$M_{mnrv} = \frac{\Sigma(\xi_i^{(m)} - \langle \xi^{(m)} \rangle)(\xi_i^{(n)} - \langle \xi^{(n)} \rangle)(\xi_i^{(r)} - \langle \xi^{(r)} \rangle)(\xi_i^{(v)} - \langle \xi^{(v)} \rangle) w_i + f_5 \frac{1}{3}(C_{mn}C_{rv} + C_{mr}C_{nv} + C_{mv}C_{nr})}{f_6(n-4)}$$

where the factors are

$$f_1 = \frac{\Sigma w_i - \frac{\Sigma w_i^2}{\Sigma w_i}}{n-1} \qquad f_2 = \frac{\Sigma w}{n}$$

$$f_3 = \frac{2[((\Sigma w^2)^2 - \Sigma w^4)]}{2n(n-1)} \qquad f_4 = \frac{4(\Sigma w^2 (\Sigma w)^2 - 2\Sigma w^3 \Sigma w + 2\Sigma w^4 - (\Sigma w^2)^2)}{4n(n-1)(n-2)}$$

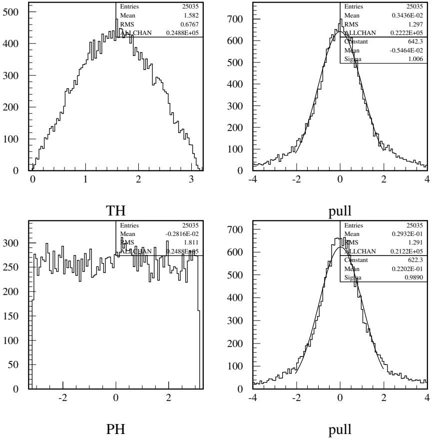
$$f_5 = \frac{1}{2(\Sigma w)^3}(2\Sigma w(\Sigma w \Sigma w^2 - \Sigma w^3) - 3(\Sigma w^2)^2 + 3\Sigma w^4) \qquad f_6 = \frac{1}{(\Sigma w)^3}\frac{(\Sigma w)^4 - 4(\Sigma w)^2(\Sigma w^2) + 6\Sigma w \Sigma w^3 - 3\Sigma w^4)}{n-4}$$

Checking the math

- All coded in Fortran95 (10 statements ...)
- Check with sets of points from known distributions.
  - Gaussian or not
  - High stat/low stat
  - Binned or un-binned.
  - Almost spherical/cigar-shaped
  - Different directions.
- All OK, both pulls on θ and ϕ, on the elements and correlations of C^*.
- Check clusters in DBD simulation:
  - Run `RecoMCTruthLinker` with all options switched on - Most notably `CalohitMCTruthLink`.
  - Write out true id and momentum for all γ:s and K_L^0 from generator that creates hits, followed by xyz and E of each hit.
  - Read back and calculate errors, plot θ and ϕ pulls.

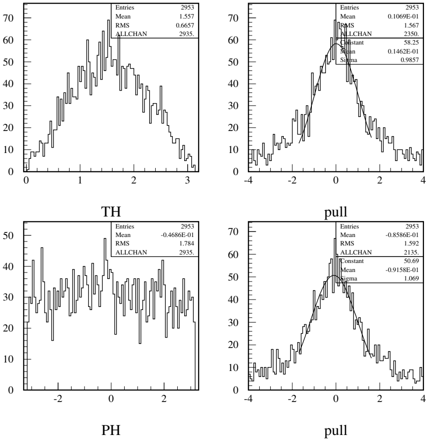
Checking the math: pulls on real clusters

- Pulls for the γ clusters.
- Pulls for the K_L^0 clusters.



Checking the math: pulls on real clusters

- Pulls for the γ clusters.
- Pulls for the K_L^0 clusters.



Classes and processors

This implemented into MARLIN as:

- One MARLINUTIL class : WEIGHEDPOINTS3D.
- One MARLINRECO processor : ADDCLUSTERPROPERTIES (in ANALYSIS)

They can be found in the SVN head of the marlin project, as of last week-end.

Classes and processors: WEIGHEDPOINTS3D

WEIGHEDPOINTS3D has methods to return the covariance matrix of the C.O.G., all eigen-values and -vectors with errors.
It has two c'tor:s:

```
WeightedPoints3D(int nhits, double* a,
      double* x, double* y, double* z);

WeightedPoints3D(const std::vector<double> &cog,
      const std::vector<double> &cov,
      const std::vector<double> &mayor_axis_error =
            std::vector<double>() ,
      int npnt = 0 ,double wgtsum =0.0 ,
      double wgt2sum=0.0 , double wgt4sum=0.0 );
```

The first calculates using the hits ⇒ Needs the calo-hits.
The second one takes a cluster position with covariance + possibly other shape-descriptors, and can return other stuff (eigen-values and -vectors with errors ...) ⇒ Does *not* need calo-hits.

Classes and processors: ADDCLUSTERPROPERTIES

- ADDCLUSTERPROPERTIES is a MARLIN processor. Only inputs: PFO and Cluster collection-names (defaults: PandoraPFOs and PandoraClusters), but the calo-hits must be in the event (throws `StopProcessingException` if not)
- It uses WEIGHEDPOINTS3D to calculate the cluster C.O.G. and θ and ϕ, with their covariance matrices . Then:

```
clu->setPosition(Position);
clu->setPositionError(&PositionError[0]);
clu->setITheta(theta);
clu->setIPhi(phi);
clu->setDirectionError(&DirectionError[0]);
```

- In addition, three shape-parameters are added to any pre-existing ones (`"npoints"`,`"sum_wgt2̂"` and `"sum_wgt4̂"`).
- **NB** If `clu->getEnergyError()` returns 0, it also calculates this from assumed numbers for ECal and HCal, and the seen E_{ECal} and total E.

Classes and processors: ADDCLUSTERPROPERTIES

- ADDCLUSTERPROPERTIES is a MARLIN processor. Only inputs: PFO and Cluster collection-names (defaults: PandoraPFOs and PandoraClusters), but the calo-hits must be in the event (throws `StopProcessingException` if not)
- It uses WEIGHEDPOINTS3D to calculate the cluster C.O.G. and θ and ϕ, with their covariance matrices . Then:

```
clu->setPosition(Position);
clu->setPositionError(&PositionError[0]);
clu->setITheta(theta);
clu->setIPhi(phi);
clu->setDirectionError(&DirectionError[0]);
```

- In addition, three shape-parameters are added to any pre-existing ones (`"npoints"`,`"sum_wgt^2"` and `"sum_wgt^4"`).
- **NB** If `clu->getEnergyError()` returns 0, it also calculates this from assumed numbers for ECal and HCal, and the seen E_{ECal} and total E.

Classes and processors: AddClusterProperties

- AddClusterProperties is a Marlin processor. Only inputs: PFO and Cluster collection-names (defaults: PandoraPFOs and PandoraClusters), but the calo-hits must be in the event (throws `StopProcessingException` if not)
- It uses WeighedPoints3D to calculate the cluster C.O.G. and θ and ϕ, with their covariance matrices . Then:

```
clu->setPosition(Position);
clu->setPositionError(&PositionError[0]);
clu->setITheta(theta);
clu->setIPhi(phi);
clu->setDirectionError(&DirectionError[0]);
```

- In addition, three shape-parameters are added to any pre-existing ones (`"npoints"`,`"sum_wgt^2"` and `"sum_wgt^4"`).
- **NB** If `clu->getEnergyError()` returns 0, it also calculates this from assumed numbers for ECal and HCal, and the seen E_{ECal} and total E.

Classes and processors: ADDCLUSTERPROPERTIES

- It then loops all PFOs to add the covariance matrix of the 4-momentum to neutrals. **NB**: Only for "typical" neutrals, ie. made of one cluster, no tracks.
- It does this assuming the neutral originates at (0,0,0), ie. the 3-momentum is in the direction of the vector to the cluster-C.O.G. The uncertainty on the direction is from `clu->getPositionError()`.
- The magnitude of the momentum is obtained by `clu->getEnergy()`, with error from `clu->getEnergyError()`.
- The total error from C.O.G. position and energy is propagated to the covariance of the 4-momentum.
- Then finally:

```
part->setMomentum(mom);
part->setCovMatrix(p_cov_v);
```

Classes and processors: ADDCLUSTERPROPERTIES

- It then loops all PFOs to add the covariance matrix of the 4-momentum to neutrals. **NB**: Only for “typical” neutrals, ie. made of one cluster, no tracks.
- It does this assuming the neutral originates at (0,0,0), ie. the 3-momentum is in the direction of the vector to the cluster-C.O.G. The uncertainty on the direction is from `clu->getPositionError()`.
- The magnitude of the momentum is obtained by `clu->getEnergy()`, with error from `clu->getEnergyError()`.
- The total error from C.O.G. position and energy is propagated to the covariance of the 4-momentum.
- Then finally:

```
part->setMomentum(mom);
part->setCovMatrix(p_cov_v);
```

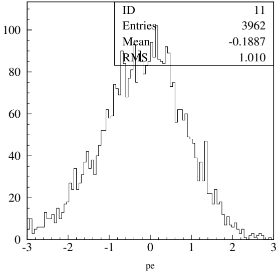
Classes and processors: AddClusterProperties

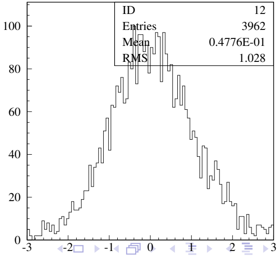
- It then loops all PFOs to add the covariance matrix of the 4-momentum to neutrals. **NB**: Only for “typical” neutrals, ie. made of one cluster, no tracks.
- It does this assuming the neutral originates at (0,0,0), ie. the 3-momentum is in the direction of the vector to the cluster-C.O.G. The uncertainty on the direction is from `clu->getPositionError()`.
- The magnitude of the momentum is obtained by `clu->getEnergy()`, with error from `clu->getEnergyError()`.
- The total error from C.O.G. position and energy is propagated to the covariance of the 4-momentum.
- Then finally:

```
part->setMomentum(mom);
part->setCovMatrix(p_cov_v);
```

Checking Classes and processors: pulls PFO 4-mom

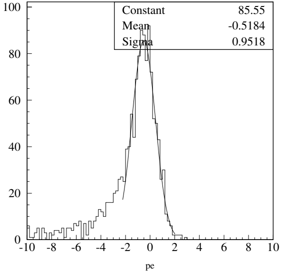
- Pulls PFO energy of γ:s.
- Pulls PFO p$_x$ of γ:s.
- Pulls PFO energy of K_L^0:s.
- Pulls PFO p$_x$ of K_L^0:s.
- **NB**: A number of tricks needed to figure out what the true value should be!

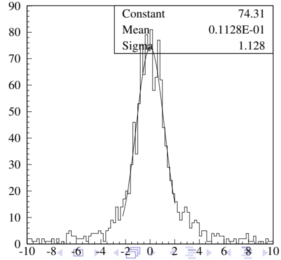




Checking Classes and processors: pulls PFO 4-mom

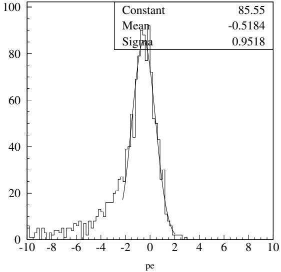
- Pulls PFO energy of γ:s.
- Pulls PFO p_x of γ:s.
- Pulls PFO energy of K_L^0:s.
- Pulls PFO p_x of K_L^0:s.
- **NB**: A number of tricks needed to figure out what the true value should be!

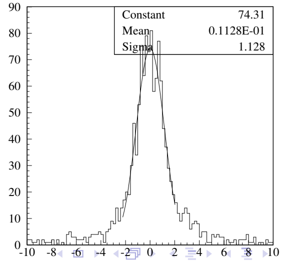




Checking Classes and processors: pulls PFO 4-mom

- Pulls PFO energy of γ:s.
- Pulls PFO p_x of γ:s.
- Pulls PFO energy of K_L^0:s.
- Pulls PFO p_x of K_L^0:s.
- **NB**: A number of tricks needed to figure out what the true value should be!





Conclusions

- Math of the general problem to determine errors on the direction of the mayor axis of weighted 3D point distributions worked out.
- Position and direction with covariances of clusters can now be evaluated using WEIGHEDPOINTS3D.
- These can be added to the clusters in the event with the MARLIN processor ADDCLUSTERPROPERTIES.
- ADDCLUSTERPROPERTIES also adds 4-momentum with correlation matrix the neutral PFOs.
- Outlook: LCNote and/or publication of the math to come.

Conclusions

- Math of the general problem to determine errors on the direction of the mayor axis of weighted 3D point distributions worked out.
- Position and direction with covariances of clusters can now be evaluated using WEIGHEDPOINTS3D.
- These can be added to the clusters in the event with the MARLIN processor ADDCLUSTERPROPERTIES.
- ADDCLUSTERPROPERTIES also adds 4-momentum with correlation matrix the neutral PFOs.
- Outlook: LCNote and/or publication of the math to come.

Conclusions

- Math of the general problem to determine errors on the direction of the mayor axis of weighted 3D point distributions worked out.
- Position and direction with covariances of clusters can now be evaluated using WEIGHEDPOINTS3D.
- These can be added to the clusters in the event with the MARLIN processor ADDCLUSTERPROPERTIES.
- ADDCLUSTERPROPERTIES also adds 4-momentum with correlation matrix the neutral PFOs.
- Outlook: LCNote and/or publication of the math to come.