Cluster errors and Truth information Algorithms

Mikael Berggren[1]

[1]DESY, Hamburg

ILD Software Workshop , DESY-HH, Feb 22-26, 2016

Cluster errors

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

The math of this has been worked out. For details see:

`http://agenda.linearcollider.org/event/6948/contribution/1/material/slides/0.pdf`

Classes and processors

This implemented into `Marlin` as:

- One `MarlinUtil` class : `WeightedPoints3D`.
- One `MarlinReco` processor : `AddClusterProperties` (in `Analysis`)

Classes and processors: `WeightedPoints3D`

`WeightedPoints3D` 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 $\Rightarrow$ 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 ...) $\Rightarrow$ 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 `WeightedPoints3D` 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, four shape-parameters are added to any pre-existing ones (`"npoints"`,`"sum_wgt"`, `"sum_wgt`$\hat{2}$`"` and `"sum_wgt`$\hat{4}$`"`).
- **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 `WeightedPoints3D` 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, four shape-parameters are added to any pre-existing ones (`"npoints",”sum_wgt”, "sum_wgt`$\hat{2}$`"` and `"sum_wgt`$\hat{4}$`"`).
- **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 `WeightedPoints3D` 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, four shape-parameters are added to any pre-existing ones (`"npoints","sum_wgt", "sum_wgt`$\hat{2}$`"` and `"sum_wgt`$\hat{4}$`"`).
- **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);
```

Questions

- Where should this be called ?
- Presently: This information is foreseen to be added by the PFO-builder *in MarlinPandora*.
- Is this really the right place for this kind of action ? Shouldn't MarlinPandora just do it's thing, and separate processor be responsible to build the PFOs, from Pandora or elsewhere ?
- In any case: The PFO-builder actually doesn't use this new code, but rather the older `ClusterShapes` (which does **not** calculate errors, but does a lot of other things ...).
- So, actually it is already the case that info is added to the PFOs down-stream from Pandora (also for charged PFOs, by Tino's processor)
- So would need to re-write `ClusterShapes` processor to use `WeightedPoints3D`, and similarly for charged PFOs.
- Question: Is `ClusterShapes` used anywhere else ?

Questions

- Where should this be called ?
- Presently: This information is foreseen to be added by the PFO-builder *in MarlinPandora*.
- Is this really the right place for this kind of action ? Shouldn't MarlinPandora just do it's thing, and separate processor be responsible to build the PFOs, from Pandora or elsewhere ?
- In any case: The PFO-builder actually doesn't use this new code, but rather the older `ClusterShapes` (which does **not** calculate errors, but does a lot of other things ...).
- So, actually it is already the case that info is added to the PFOs down-stream from Pandora (also for charged PFOs, by Tino's processor)
- So would need to re-write `ClusterShapes` processor to use `WeightedPoints3D`, and similarly for charged PFOs.
- Question: Is `ClusterShapes` used anywhere else ?

Questions

- Where should this be called ?
- Presently: This information is foreseen to be added by the PFO-builder *in MarlinPandora*.
- Is this really the right place for this kind of action ? Shouldn't MarlinPandora just do it's thing, and separate processor be responsible to build the PFOs, from Pandora or elsewhere ?
- In any case: The PFO-builder actually doesn't use this new code, but rather the older `ClusterShapes` (which does **not** calculate errors, but does a lot of other things ...).
- So, actually it is already the case that info is added to the PFOs down-stream from Pandora (also for charged PFOs, by Tino's processor)
- So would need to re-write `ClusterShapes` processor to use `WeightedPoints3D`, and similarly for charged PFOs.
- Question: Is `ClusterShapes` used anywhere else ?

Questions

- Where should this be called ?
- Presently: This information is foreseen to be added by the PFO-builder *in MarlinPandora*.
- Is this really the right place for this kind of action ? Shouldn't MarlinPandora just do it's thing, and separate processor be responsible to build the PFOs, from Pandora or elsewhere ?
- In any case: The PFO-builder actually doesn't use this new code, but rather the older `ClusterShapes` (which does **not** calculate errors, but does a lot of other things ...).
- So, actually it is already the case that info is added to the PFOs down-stream from Pandora (also for charged PFOs, by Tino's processor)
- So would need to re-write `ClusterShapes` processor to use `WeightedPoints3D`, and similarly for charged PFOs.
- Question: Is `ClusterShapes` used anywhere else ?

Physics ⇒ Whizard ⇒ Parton shower ⇒ hadronisation ⇒ decays ⇒ Geant ⇒ MarlinReco ⇒ Pandora ⇒ Jet clustering ⇒YOU

The `TrueJet` processor tries to connect YOU with the Physics using the true information about the event.

- The connection from Geant to You is done by the `RecoMCTruthLinker` processor, linking PFOs (and jets) to MCParticles.
- `TrueJet` takes care of the rest: How does the MCParticles connect to the hard event.

Physics ⇒ .... ⇒ Geant ⇒ MarlinReco ⇒ Pandora ⇒ Jet clustering ⇒YOU

From ReconstructedParticle to MCParticle and back :

`RecoMCTruthLinker`

- There is a gap between MCParticles an ReconstructedParticles:
  - From Geant, the MCParticle creating every SimHit is known.
  - In digitisation and further reconstruction, SimHits are input, but the connection to the true particle creating the hit is not carried on.
  - `RecoMCTruthLinker` takes care of re-establishing this link.
- Input is the relations SimTrackerHit/SimCaloHit ↔ MCParticle, TrackerHit/CaloHit ↔ Track/Cluster and Track/Cluster ↔ ReconstructedParticle.
- Output is navigators between ReconstructedParticle/Track/Cluster and MCParticle.

Physics ⇒ .... ⇒ Geant ⇒ MarlinReco ⇒ Pandora ⇒ Jet clustering ⇒YOU

From ReconstructedParticle to MCParticle and back :

`RecoMCTruthLinker`

- There is a gap between MCParticles an ReconstructedParticles:
  - From Geant, the MCParticle creating every SimHit is known.
  - In digitisation and further reconstruction, SimHits are input, but the connection to the true particle creating the hit is not carried on.
  - `RecoMCTruthLinker` takes care of re-establishing this link.
- Input is the relations SimTrackerHit/SimCaloHit ↔ MCParticle, TrackerHit/CaloHit ↔ Track/Cluster and Track/Cluster ↔ ReconstructedParticle.
- Output is navigators between ReconstructedParticle/Track/Cluster and MCParticle.

Physics ⇒ Whizard ⇒ Parton shower ⇒ hadronisation ⇒ decays ⇒ Geant ⇒ ... ⇒YOU

From MCParticles to Physics and back: `TrueJet`

- To link further back, `TrueJet` joins hadrons from the final colour singlets to di-jets.
- The di-jet is split into two jets, connected to the final quarks.
- It follows the decay-chain of the primary hadrons, and assigns each of them to the jet of it's parent.
- The process continues from generated to simulated particles.
- Then the final quark is followed back through the parton-shower.
- Ultimately, the initial colour singlet is found.
- Rather complicated: use the helper-class `TrueJet_Parser`.

The initial colour singlet is the closest one gets to the initial physics (W,Z,h,...). For details: `http://agenda.linearcollider.org/event/6787/session/4/contribution/6/material/slides/0.pdf`

RecoMCTruthLinker: Updates

The "new" (wrt. DBD) `RecoMCTruthLinker`:

- RecoMCTruthLink is supplemented with MCTruthRecoLink to make it bi-directional in weight.
- Optionally, the weight can be redefined to contain weights to and from both clusters and tracks. (This feature is in the DBD version, but not used).
- For neutrals, it links all true particles that contributes.
- ClusterMCTruthLink is supplemented with MCTruthClusterLink to make it bi-directional in weight.
- TrackMCTruthLink and MCTruthTrackLink is bi-directional in the weight definition.

RecoMCTruthLinker: Updates

The "new" (wrt. DBD) `RecoMCTruthLinker`:

- RecoMCTruthLink is supplemented with MCTruthRecoLink to make it bi-directional in weight.
- Optionally, the weight can be redefined to contain weights to and from both clusters and tracks. (This feature is in the DBD version, but not used).
- For neutrals, it links all true particles that contributes.
- ClusterMCTruthLink is supplemented with MCTruthClusterLink to make it bi-directional in weight.
- TrackMCTruthLink and MCTruthTrackLink is bi-directional in the weight definition.

RecoMCTruthLinker: Updates

The "new" (wrt. DBD) `RecoMCTruthLinker`:

- RecoMCTruthLink is supplemented with MCTruthRecoLink to make it bi-directional in weight.
- Optionally, the weight can be redefined to contain weights to and from both clusters and tracks. (This feature is in the DBD version, but not used).
- For neutrals, it links all true particles that contributes.
- ClusterMCTruthLink is supplemented with MCTruthClusterLink to make it bi-directional in weight.
- TrackMCTruthLink and MCTruthTrackLink is bi-directional in the weight definition.

RecoMCTruthLinker: Updates

The "new" (wrt. DBD) `RecoMCTruthLinker`:

- RecoMCTruthLink is supplemented with MCTruthRecoLink to make it bi-directional in weight.
- Optionally, the weight can be redefined to contain weights to and from both clusters and tracks. (This feature is in the DBD version, but not used).
- For neutrals, it links all true particles that contributes.
- ClusterMCTruthLink is supplemented with MCTruthClusterLink to make it bi-directional in weight.
- TrackMCTruthLink and MCTruthTrackLink is bi-directional in the weight definition.

RecoMCTruthLinker: Updates

The “new” (wrt. DBD) `RecoMCTruthLinker`:

- RecoMCTruthLink is supplemented with MCTruthRecoLink to make it bi-directional in weight.
- Optionall … ghts to and from both … BD version, but not u …
- For neutrals, it links all true particles that contributes.
- ClusterMCTruthLink is supplemented with MCTruthClusterLink to make it bi-directional in weight.
- TrackMCTruthLink and MCTruthTrackLink is bi-directional in the weight definition.

> The aim is that the linking should be complete in both directions, ie. not just right in 98 % of the cases, but always !

RecoMCTruthLinker: Linking clusters

- The idea: Cluster $\leftrightarrow$ All particles hitting the calorimeters, and that contribute with at least one calo-hit to the cluster.
- The weights are
  - In one direction: E_{calo}(from MCP *in this cluster*)/E_{calo}(from MCP)
  - In the other: E_{calo}(In cluster *from this MCP*)/E_{calo}(In cluster)
- The part "all particles hitting the calorimeter" is tricky:
  - Sometimes, a calo-hit comes from an MCParticle created inside the calorimeter: Need to back-track.
  - Back-scatters: Do they end up in the same cluster they came from ?
  - The dogma is that one can figure out that a particle started in the tracker by knowing that it's mother ended there.
  - Not! Non-destructive interactions: In Geant (but never in the generator) a particle might create new particles without disappearing. In this case, one must
    1. Detect that this thing happened (and work around a bug in Mokka)
    2. Use a set of tricks to figure out/guess if it happened in the tracker.

RecoMCTruthLinker: Linking clusters

- The idea: Cluster $\leftrightarrow$ All particles hitting the calorimeters, and that contribute with at least one calo-hit to the cluster.
- The weights are
  - In one direction: E_{calo}(from MCP *in this cluster*)/E_{calo}(from MCP)
  - In the other: E_{calo}(In cluster *from this MCP*)/E_{calo}(In cluster)
- The part "all particles hitting the calorimeter" is tricky:
  - Sometimes, a calo-hit comes from an MCParticle created inside the calorimeter: Need to back-track.
  - Back-scatters: Do they end up in the same cluster they came from ?
  - The dogma is that one can figure out that a particle started in the tracker by knowing that it's mother ended there.
  - Not! Non-destructive interactions: In Geant (but never in the generator) a particle might create new particles without disappearing. In this case, one must
    1. Detect that this thing happened (and work around a bug in MOKKA)
    2. Use a set of tricks to figure out/guess if it happened in the tracker.

RecoMCTruthLinker: Linking clusters

- The idea: Cluster $\leftrightarrow$ All particles hitting the calorimeters, and that contribute with at least one calo-hit to the cluster.
- The weights are
  - In one direction: E_{calo}(from MCP *in this cluster*)/E_{calo}(from MCP)
  - In the other: E_{calo}(In cluster *from this MCP*)/E_{calo}(In cluster)
- The part "all particles hitting the calorimeter" is tricky:
  - Sometimes, a calo-hit comes from an MCParticle created inside the calorimeter: Need to back-track.
  - Back-scatters: Do they end up in the same cluster they came from ?
  - The dogma is that one can figure out that a particle started in the tracker by knowing that it's mother ended there.
  - Not! Non-destructive interactions: In Geant (but never in the generator) a particle might create new particles without disappearing. In this case, one must
    1. Detect that this thing happened (and work around a bug in MOKKA)
    2. Use a set of tricks to figure out/guess if it happened in the tracker.

RecoMCTruthLinker: Linking clusters

- The idea: Cluster $\leftrightarrow$ All particles hitting the calorimeters, and that contribute with at least one calo-hit to the cluster.
- The weights are
  - In one direction: E_{calo}(from MCP *in this cluster*)/E_{calo}(from MCP)
  - In the other: E_{calo}(In cluster *from this MCP*)/E_{calo}(In cluster)
- The part "all particles hitting the calorimeter" is tricky:
  - Sometimes, a calo-hit comes from an MCParticle created inside the calorimeter: Need to back-track.
  - Back-scatters: Do they end up in the same cluster they came from ?
  - The dogma is that one can figure out that a particle started in the tracker by knowing that it's mother ended there.
  - Not! Non-destructive interactions: In Geant (but never in the generator) a particle might create new particles without disappearing. In this case, one must
    1. Detect that this thing happened (and work around a bug in MOKKA)
    2. Use a set of tricks to figure out/guess if it happened in the tracker.

RecoMCTruthLinker: Linking clusters

- The idea: Cluster $\leftrightarrow$ All particles hitting the calorimeters, and that contribute with at least one calo-hit to the cluster.
- The weights are
  - In one direction: E_{calo}(from MCP *in this cluster*)/E_{calo}(from MCP)
  - In the other: E_{calo}(In cluster *from this MCP*)/E_{calo}(In cluster)
- The part "all particles hitting the calorimeter" is tricky:
  - Sometimes, a calo-hit comes from an MCParticle created inside the calorimeter: Need to back-track.
  - Back-scatters: Do they end up in the same cluster they came from ?
  - The dogma is that one can figure out that a particle started in the tracker by knowing that it's mother ended there.
  - Not! Non-destructive interactions: In Geant (but never in the generator) a particle might create new particles without disappearing. In this case, one must
    1. Detect that this thing happened (and work around a bug in MOKKA)
    2. Use a set of tricks to figure out/guess if it happened in the tracker.

RecoMCTruthLinker: Linking clusters

- The idea: Cluster $\leftrightarrow$ All particles hitting the calorimeters, and that contribute with at least one calo-hit to the cluster.
- The weights are
  - In one direction: E_{calo}(from MCP *in this cluster*)/E_{calo}(from MCP)
  - In the other: E_{calo}(In cluster *from this MCP*)/E_{calo}(In cluster)
- The part "all particles hitting the calorimeter" is tricky:
  - Sometimes, a calo-hit comes from an MCParticle created inside the calorimeter: Need to back-track.
  - Back-scatters: Do they end up in the same cluster they came from ?
  - The dogma is that one can figure out that a particle started in the tracker by knowing that it's mother ended there.
  - Not! Non-destructive interactions: In Geant (but never in the generator) a particle might create new particles without disappearing. In this case, one must
    1. Detect that this thing happened (and work around a bug in MOKKA)
    2. Use a set of tricks to figure out/guess if it happened in the tracker.

RecoMCTruthLinker and WeightedPoints3D at work: the `TrueCluster` processor

(This is not yet committed ....)

- `TrueCluster` creates Cluster-objects containing all hits a given MCParticle created.
- It also creates cluster-parts (also Cluster-objects) of the hits of a given MCParticle that ended up in a given PandoraCluster.
- ... and navigators MCP $\leftrightarrow$ true cluster, true cluster $\leftrightarrow$ cluster-part, and cluster-part $\leftrightarrow$ Pandora cluster.
- Useful for:
  - FastSim parametrisation on PFA.
  - PFA performance studies and confusion parametrisations.
  - Checking eg. π^0 finding algorithms in an ideal world.
  - Factoring out PFA in jet-finding.
  - ...

RecoMCTruthLinker and WeightedPoints3D at work: the `TrueCluster` processor

(This is not yet committed ....)

- `TrueCluster` creates Cluster-objects containing all hits a given MCParticle created.
- It also creates cluster-parts (also Cluster-objects) of the hits of a given MCParticle that ended up in a given PandoraCluster.
- ... and navigators MCP $\leftrightarrow$ true cluster, true cluster $\leftrightarrow$ cluster-part, and cluster-part $\leftrightarrow$ Pandora cluster.
- Useful for:
  - FastSim parametrisation on PFA.
  - PFA performance studies and confusion parametrisations.
  - Checking eg. π^0 finding algorithms in an ideal world.
  - Factoring out PFA in jet-finding.
  - ...

Summary and Outlook

- `WeightedPoints3D` and `AddClusterProperties` adds the missing data-members in Cluster and ReconstructedParticle classes.
- `RecoMCTruthLinker` (+ `TrueCluster`) is useful for studying particle-flow in detail.
- It is a pre-requisite for `TrueJet` to be maximally useful.
- `TrueJet` will be useful for disentangling effects of jet clustering from particle flow, from combinatorics, for detector effects.
- It is also useful for testing and developing overlay-removal, jet-clustering and secondary vertex methods.

Outlook:

- Future of `ClusterShapes`?
- Right place to build PFOs?
- Get `RecoMCTruthLinker` working in the new world of DD4Hep

Summary and Outlook

- `WeightedPoints3D` and `AddClusterProperties` adds the missing data-members in Cluster and ReconstructedParticle classes.
- `RecoMCTruthLinker` (+ `TrueCluster`) is useful for studying particle-flow in detail.
- It is a pre-requisite for `TrueJet` to be maximally useful.
- `TrueJet` will be useful for disentangling effects of jet clustering from particle flow, from combinatorics, for detector effects.
- It is also useful for testing and developing overlay-removal, jet-clustering and secondary vertex methods.

Outlook:

- Future of `ClusterShapes`?
- Right place to build PFOs?
- Get `RecoMCTruthLinker` working in the new world of DD4Hep

Summary and Outlook

- `WeightedPoints3D` and `AddClusterProperties` adds the missing data-members in Cluster and ReconstructedParticle classes.
- `RecoMCTruthLinker` (+ `TrueCluster`) is useful for studying particle-flow in detail.
- It is a pre-requisite for `TrueJet` to be maximally useful.
- `TrueJet` will be useful for disentangling effects of jet clustering from particle flow, from combinatorics, for detector effects.
- It is also useful for testing and developing overlay-removal, jet-clustering and secondary vertex methods.

Outlook:

- Future of `ClusterShapes`?
- Right place to build PFOs?
- Get `RecoMCTruthLinker` working in the new world of DD4Hep

Summary and Outlook

- `WeightedPoints3D` and `AddClusterProperties` adds the missing data-members in Cluster and ReconstructedParticle classes.
- `RecoMCTruthLinker` (+ `TrueCluster`) is useful for studying particle-flow in detail.
- It is a pre-requisite for `TrueJet` to be maximally useful.
- `TrueJet` will be useful for disentangling effects of jet clustering from particle flow, from combinatorics, for detector effects.
- It is also useful for testing and developing overlay-removal, jet-clustering and secondary vertex methods.

Outlook:

- Future of `ClusterShapes`?
- Right place to build PFOs?
- Get `RecoMCTruthLinker` working in the new world of DD4Hep
  = correct spelling of SimHit collections in steering =DONE