# Updates 11/2

- Encountered some concerning discrepancies with energy, mass, and momentum of MC particles
- There may be a problem with our current particle gun setup for taus
- Weekly update on track-cluster matching

# Energy of MCPs – setting and retrieving

- In the lcio_particle_gun gen file, we set the mass and momentum of the tau
- This is where we sample from a flat pT distribution, mass is set at 1.78 GeV
- The MCParticleImpl class doesn't have a setEnergy function, so we cannot set energy manually at this juncture
- When we **retrieve** the energy, it is calculated from p and m:

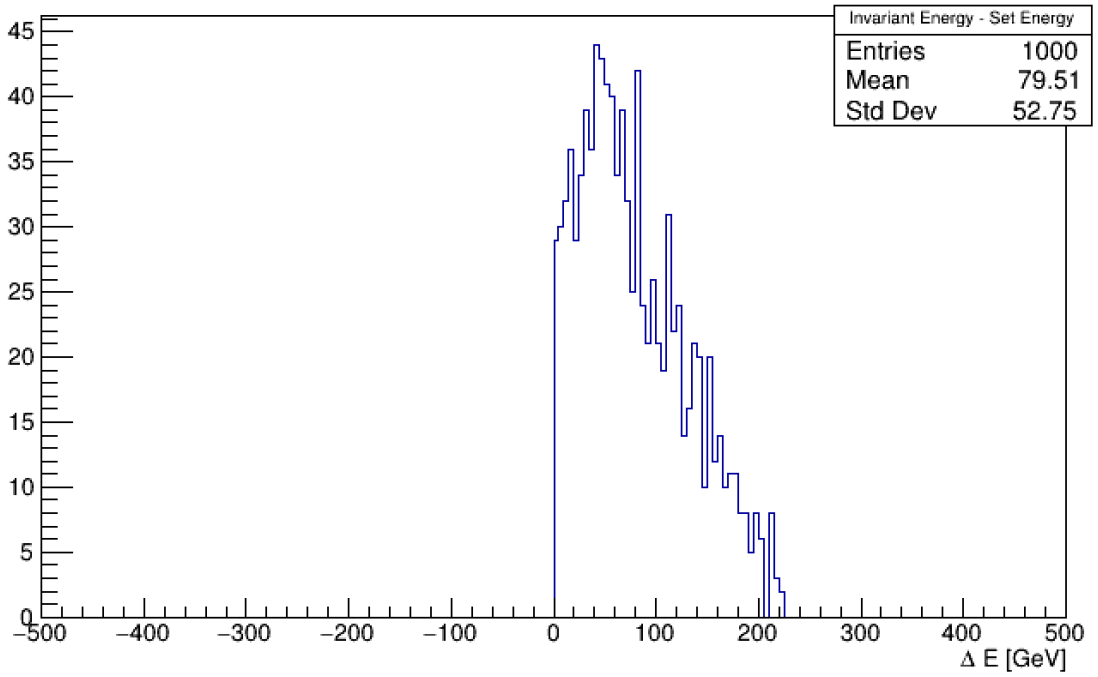
```
/** Returns the energy of the particle (at the vertex) in [GeV] computed from
 * the particle's momentum and mass.
 */
virtual double getEnergy() const ;
```

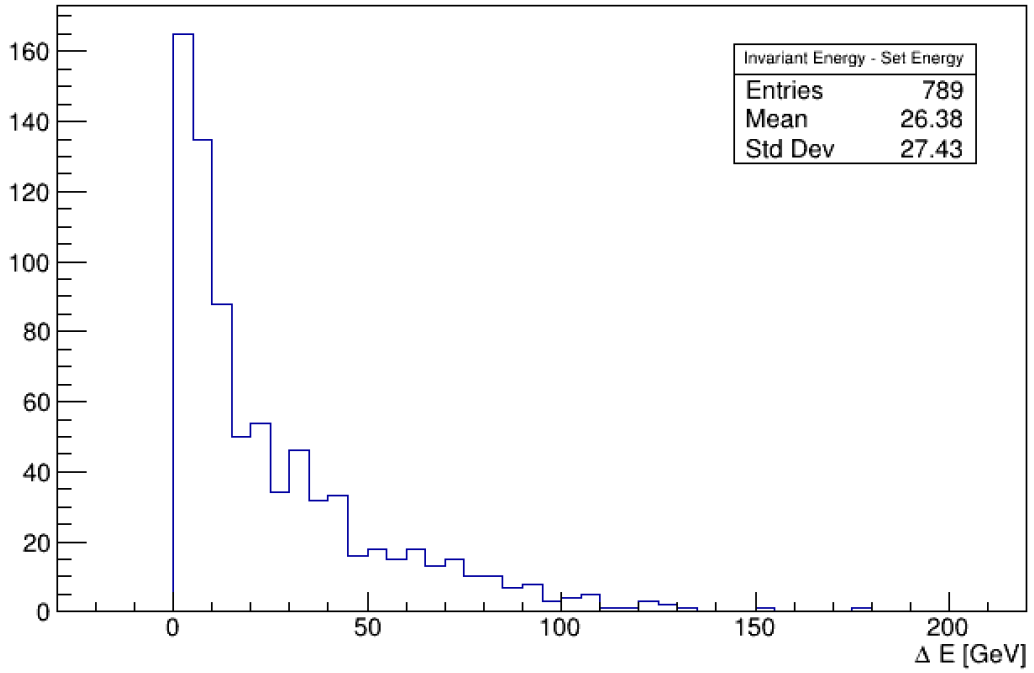
- Therefore, mcp.getEnergy() for an mcp in the sample should yield $E = \sqrt{p^2+m^2}$
- This is not the case

```
# ---------------- create MCParticle --------------------

        mcp = IMPL.MCParticleImpl()

        mcp.setGeneratorStatus(genstat)
        mcp.setMass(mass)
        mcp.setPDG(pdg)
        mcp.setMomentum(momentum)
        mcp.setCharge(charge)
        mcp.setVertex(vertex)
        mcp.setTime(time)

        if (decayLen < 1.e9):   # arbitrary ...
            mcp.setEndpoint(endpoint)

        print("  ", ipart, pdg, charge, pt, phi, theta)


# -------------------------------------------------------
```

# $E_{invariant}$-$E_{MCP}$ (taus on left, pions on right)

$E_{invariant}$ is the energy we **should** get according to definition on previons slide → $E_{invariant}$ =sqrt($p_{MCP}^{2+}m_{mcp}^{2}$)
$E_{MCP}$ is what we get from mcp->getEnergy()

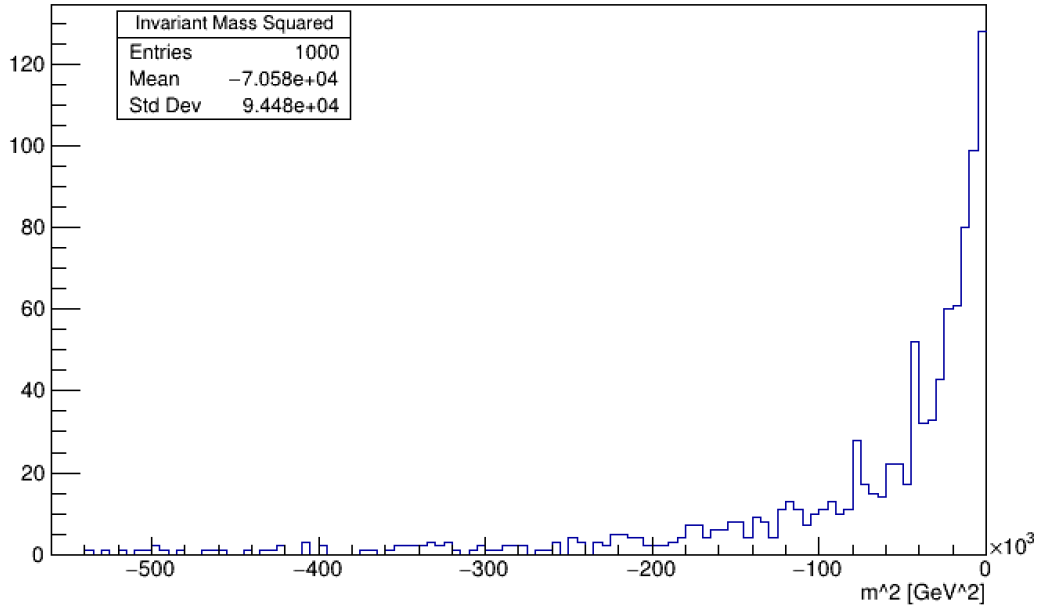


# Problems with invariant mass

- The energy and momentum that are set for the particle therefore do not correspond to the mass that has been set (e.g. 1.78 GeV for taus)
- Calculated $m^2=E^2-p^2$:
  - We have **non-real** invariant masses:
- No reason we should have off-shell taus
- The question: why is mcp->getEnergy()

returning a value not equal to sqrt($m^2+p^2$)?



# Particle gun steering – another place E could be set

- In the simulation steering file, the particle gun energy is explicitly set:

```
144     SIM.gun.energy = GUN_ENERGY*GeV
```

- This is in https://github.com/madbaron/SteeringMacros/blob/master/Sim/sim_steer_particleGun_CONDOR.py which is set up for Condor, I don't know what has been entered in the submission file for GUN_ENERGY, but perhaps we are force-setting a value of E here which doesn't correspond to the invariant definition?
- Would appreciate any insights people may have!

# Back to track-cluster matching

- We have confirmed over the past week that none of the cuts that can be input to the TrackClusterAssociationAlgorithm are wholly responsible for the failing in matching
  - Turned all of them off individually, relaxing parallel distance brought about a 20% increase in efficiency but we are peaking around 40%
- However, track-cluster matching still appears to be the main roadblock to reconstruction efficiency for the following reasons:
  - Turning off the requirement for a track to be matched to a cluster sends our efficiency up to the 80-100% range
  - The majority of events have both a track and cluster, truth-pdg=±211, within geometrical range, and somehow the track is being ignored and cluster is being reco'd as a neutron or photon
  - Most events default to neutrals because a cluster w/o a matched track is sent automatically to neutral reco Pandora algorithms
- Sergo, Fede, and I are going over the track-cluster distance calculations with a fine tooth comb to make sure we understand exactly where they are taking the track state, but it would appear that none of the parameters we can vary externally are responsible for the problem
- Perhaps something deeper in the black box of Pandora…