Conversion from energy deposit to number of photoelectrons

Minho Kim
Argonne National Laboratory

BIC Simulation Meeting
March 16, 2026

Argonne NATIONAL LABORATORY

Conversion from edep to Npe

- The conversion is being added in PulseGeneration algorithm.
- It can be applied even when the conversion factor depends on certain fields.

PulseGenerationConfig.h

EICrecon / src / algorithms / digi / **PulseGenerationConfig.h**

```
// Parameters for converting energy deposit to number of photoelectrons
std::string readout{};
std::vector<std::string> edep_to_npe_fields{}; // What fields does the conversion factor depends on (e.g., layer, grid)?
std::string edep_to_npe_filename{}; // If the conversion factor is field-dependent, we use a lookup table.
double edep_to_npe{};  // A representative conversion factor. If the conversion factor is field-independent, this value is used.
```

bic_edepToNpe_layer.lut

epic-data / bic_edepToNpe_layer.lut

```
1  1.0000
2  0.9880
3  0.9717
4  0.9535
5  0.9317
6  0.9199
7  0.9057
8  0.8928
9  0.8806
10 0.8651
11 0.8511
12 0.8367
```

- If the conversion factor depends on sector, layer, and grid, the lookup table would have 4 rows.: sector, layer, grid, and factor.
  - In this case, the conversion factor would be edep_to_npe $*$ factor[sec][lay][grid].

PulseGeneration::init

```
std::string filename = fmt::format("calibrations/{}", m_cfg.edep_to_npe_filename);
std::ifstream infile(filename);
if (!infile) {
  throw std::runtime_error(
      fmt::format("Unable to open LUT file: {}", m_cfg.edep_to_npe_filename));
}
std::string line;
while (std::getline(infile, line)) {
  std::istringstream iss(line);
  std::vector<int> keys;
  for (std::size_t i = 0; i < m_cfg.edep_to_npe_fields.size(); i++) {
    int value;
    iss >> value;
    keys.push_back(value);
  }
  double factor;
  iss >> factor;
  m_edep_to_npe_factors[keys] = factor;
}
```

Open the lookup table.
Get the field values and the corresponding conversion factors.
Store them in std::map<std::vector<int>, double> m_edep_to_npe_factors.

PulseGeneration::process

```
if (m_edep_to_npe) {
  double npe = charge * m_edep_to_npe.value();

  if (!m_edep_to_npe_factors.empty()) {
    std::vector<int> field_values;
    for (std::size_t i = 0; i < m_cfg.edep_to_npe_fields.size(); ++i) {
      const int value = id_dec != nullptr && !m_cfg.edep_to_npe_fields[i].empty()
                            ? static_cast<int>(id_dec->get(hit.getCellID(), m_field_idxs[i]))
                            : -1;
      field_values.push_back(value);
    }
    auto it = m_edep_to_npe_factors.find(field_values);
    if (it != m_edep_to_npe_factors.end()) {
      npe *= it->second;
    }
  }

  std::poisson_distribution<> poisson(npe);
  charge = poisson(m_gen);
}
```

For a given energy deposit at a SiPM, get the field values (e.g., layer = 3, grid = 5).
Find the corresponding conversion factor.
Apply the Poisson smearing.