



argon_box.py

Dan Dwyer (LBNL)

DUNE Near Detector Workshop / CERN

Nov. 6, 2017



Overview

argon_box.py

What it is:

- A plain-vanilla Geant4 simulation program
- A tool for very quick studies of particle tracking and energy deposition in a large box of liquid argon ($200\text{ m} \times 200\text{ m} \times 200\text{ m}$).
- A single-file python script which can be run against any standard Geant4 (+ G4py) installation.
- Inputs: Initial state particles (e.g. particle gun, HEPEVT data)
- Outputs: Particle tracking steps, energy depositions
- My personal rough simulation tool; not originally intended for public use.

What it is not:

- A full detector simulation, with detailed geometry (see: DUNE GGD)
- A complete experiment analysis framework (see: LArSoft)
- A reliable physics tool (my motto: beware of any result from simulation...)



Example Operation

Simple Particle Gun:

```
$ python argon_box.py --nevents=100 --source='e-' --energy=2.0  
--output='electron_2GeV_sim.root'
```

HEPEVT input data:

```
$ python argon_box.py --nevents=100 --source='beamnu.hepevt'  
--output='example_beamnu.root'
```

Other Useful Options:

--seed=N

Change random seed

--physlist='QGSP_BERT'

Set G4 Physics list (FTFP_BERT, QGSP_BERT, QGSP_BERT_HP)

--enable_edepsim:

Enable primitive simulation of 3-D energy depositions

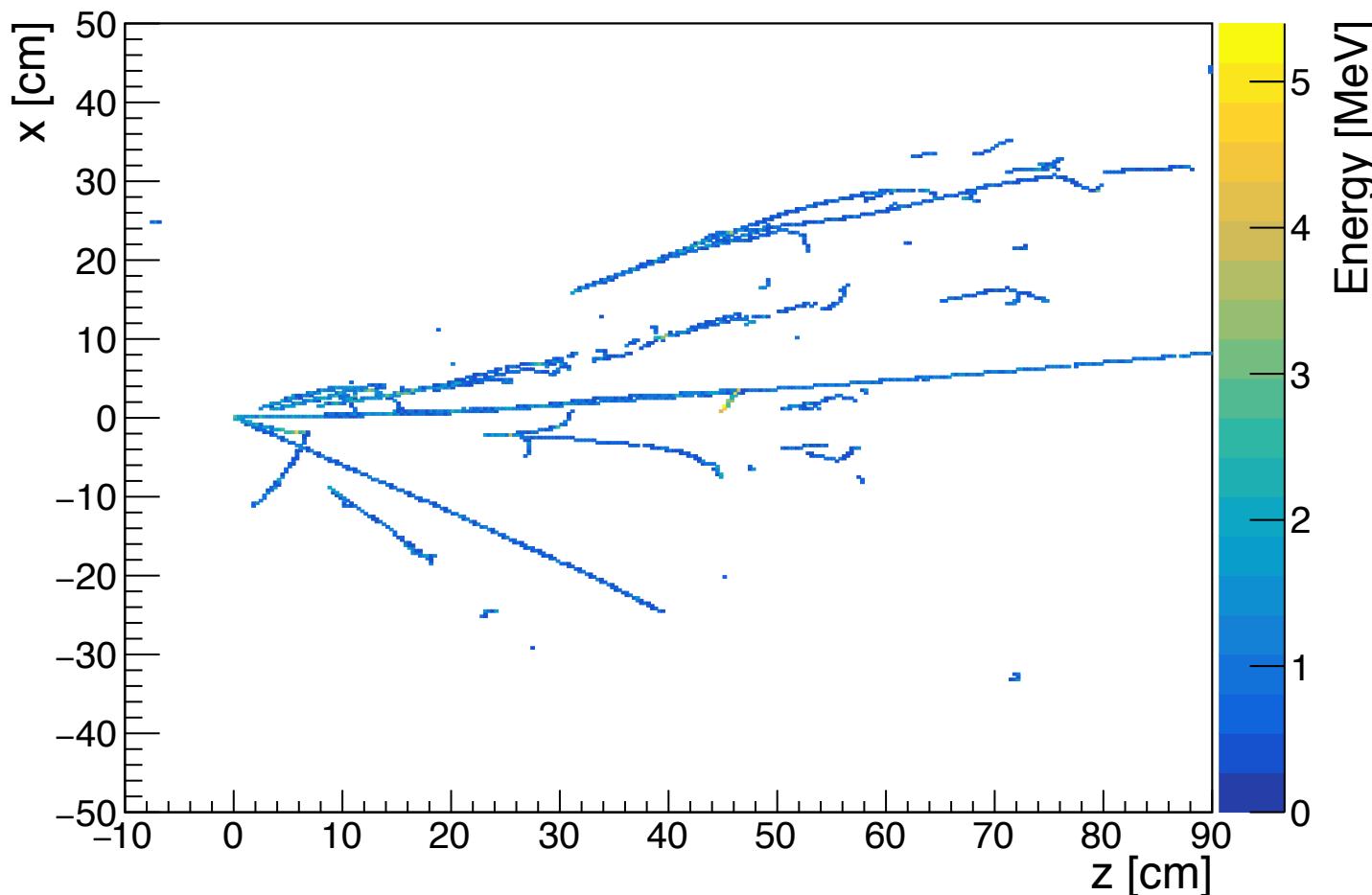
--edep_step:

Maximum energy deposition step size, in mm

Example Output

Standard ROOT output file:

```
$ root -l  
root [0] TFile f("example_beamnu.root");  
root [1] TTree* argon = (TTree*)f.Get("argon");  
root [2] TH2F h2("h2","",300,-10,90,300,-50,50);  
root [3] argon->Draw("xq:zq>>h2","dq*(ev==1)","colz");
```

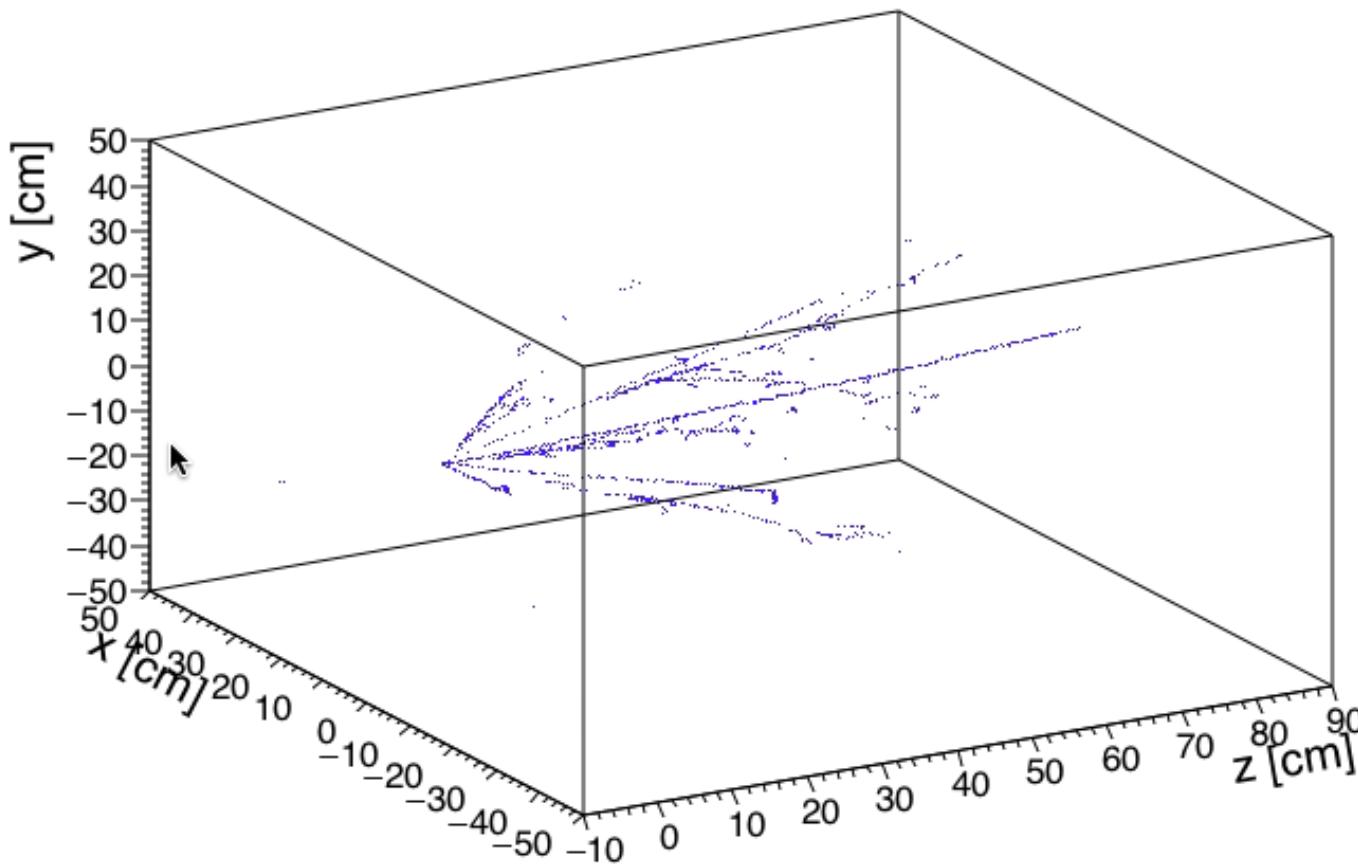




Example Output

Standard ROOT output file:

```
root [4] TH3F h3("h3","",300,-10,90,300,-50,50,300,-50,50);  
root [5] argon->Draw("xq:yq:zq>>h3","dq*(ev==1)", "");
```





Output File Contents

```
# General Info
ev      # Event number

# Ancestor particle information (e.g. initial neutrino)
pida    # PID of particle
xa      # Interaction position of particle [cm,ns]
ya
za
ta
pxa    # Momentum of particle [GeV/c]
pya
pza
ekina   # Kinetic energy of particle [GeV]
ma      # Rest mass of particle [GeV/c^2]

# Primary particle information (e.g. leptons, hadrons from neutrino interaction)
ni
pidi[ni] # PID of particle
xi[ni]   # Initial position of particle [cm,ns]
yi[ni]
zi[ni]
ti[ni]
pxi[ni]  # Initial momentum of particle [GeV/c]
pyi[ni]
pzi[ni]
ekini[ni] # Initial kinetic energy of particle [GeV]
mi[ni]   # Rest mass of particle [GeV/c^2]

# Geant particle track step information
nstep
tid[nstep] # Track ID of particle
pid[nstep] # PID of particle
parid[nstep] # Parent track ID
ekin[nstep] # Kinetic energy of particle during step [GeV]
edep[nstep] # Energy deposit of particle during step [GeV]
xs[nstep]  # Start of track step [cm]
ys[nstep]
zs[nstep]
xe[nstep]  # End of track step [cm]
ye[nstep]
ze[nstep]

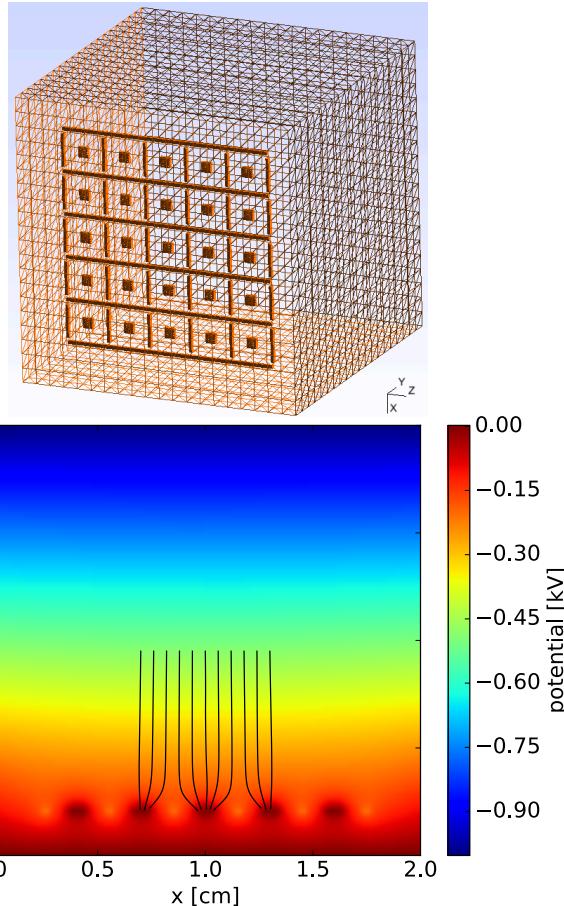
# Primitive energy deposition simulation (linearly along each track step)
nq
tidq[nq] # Track ID of particle
pidq[nq] # PID of particle
sidq[nq] # ID of track step
dq[nq]   # Energy deposit [MeV]
xq[nq]   # Position of energy deposit [cm]
yq[nq]
zq[nq]
```



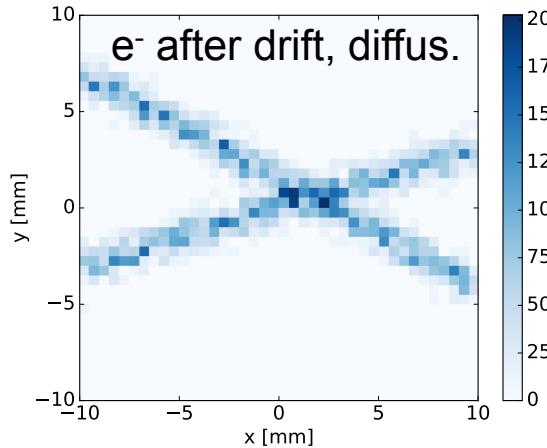
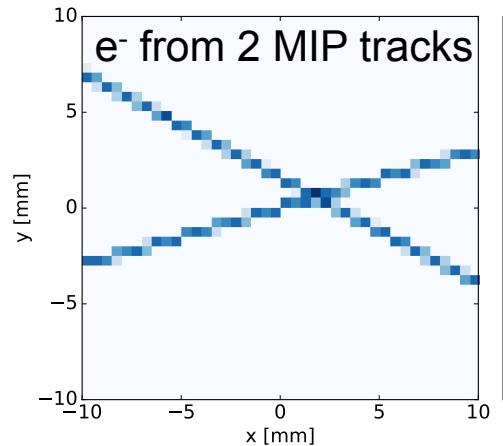
My Motivation: IC Design

Developed a set of tools to assess TPC readout design

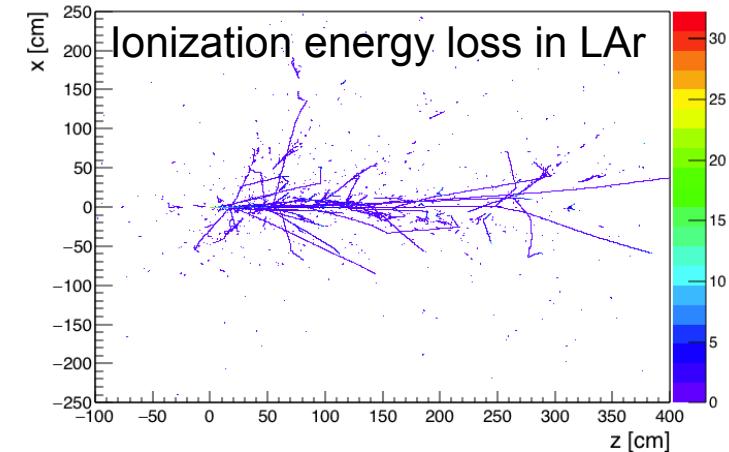
Sensor model in 3D



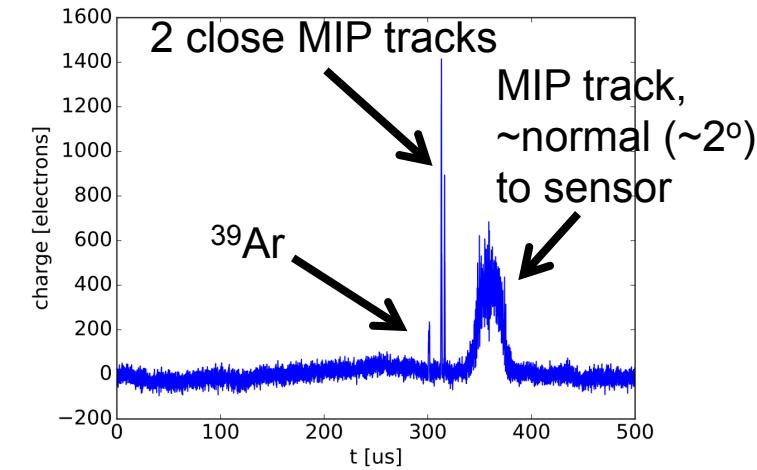
Ionization, recomb.,
drift loss, diffusion



Fast primitive Geant4 sim.



Realistic signals (with noise)



Signals have been input to IC modeling program
(Spice, Cadence) to assess LArPix design, performance.



Installation

Step 1: Install Geant4

See:

<http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/>

(...or use an existing installation at your cluster of choice.)

Step 2: Install G4py (Geant4 Python Interface)

See:

<http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/BackupVersions/V9.3/html/apas08.html>

Or maybe try:

<http://www.christopherpoole.net/compiling-and-installing-the-geant4-python-bindings.html>

(I wish the Geant4 community had made this easier.)

Step 3: Download argon_box.py script

\$ git clone https://github.com/dadwyer/argon_box.git

(...or just ask someone to email you the argon_box.py file.)



Summary

Particle Simulation in LAr

If you need detailed detector geometry, framework access, recon algs, etc:

→ See [DUNEgDD](#), [LArSoft](#)

But if you have simple questions on particle tracking and energy deposition

→ [argon_box.py](#) may be a quick and useful tool

Next Steps:

Plan to incorporate simple models for recombination, drift, diffusion

→ To assist electronics design efforts.

Chris Marshall also adding a variety of features to aid quick studies.

Warning: Don't trust any simulation without first making some basic cross-checks...