Software and computing challenges

with focus on simulation of particle passage through matter

Anna Zaborowska

CERN

NORCC 30 June 2022

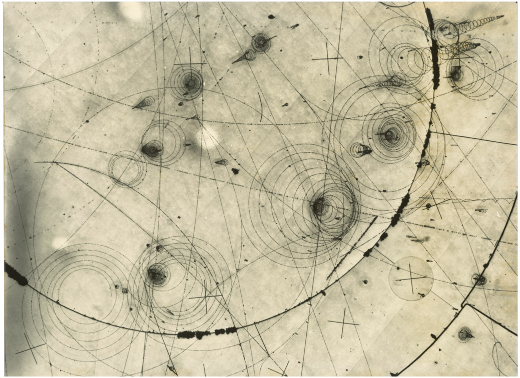
Beginnings of software for HEP

Bubble chamber

Donald Glaser
1960

wikipedia Nobel Prize

Scanning pictures from the 2 m Hydrogen Bubble Chamber,
Aug 1974

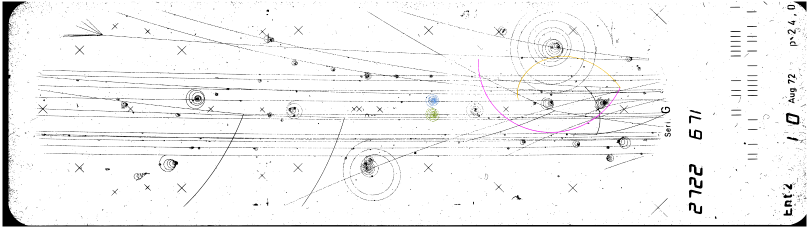


CERN-PHOTO-847-106

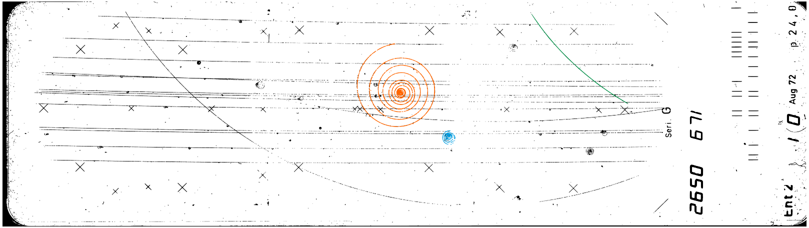
More details on bubble chambers

CERN-PHOTO-7408141-1

Beginnings of software for HEP



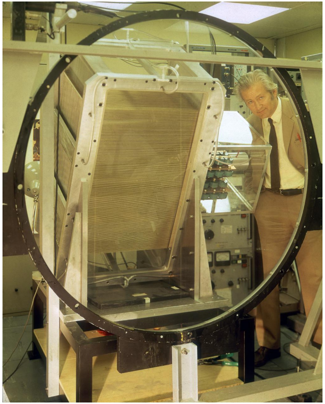
OPEN-PHO-EXP-1972-001-11



OPEN-PHO-EXP-1972-001-7

Beginnings of software for HEP

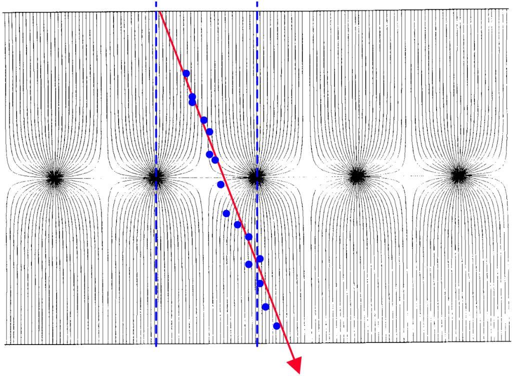
Multi-wire proportional chambers (MWPC)



CERN 7304218

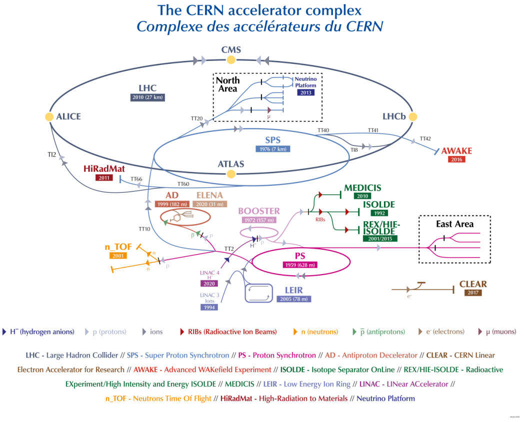
Georges Charpak
1992

wikipedia Nobel Prize

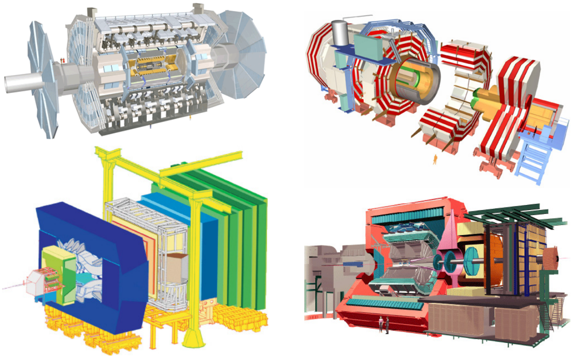


C. Joram

Beginnings of software for HEP

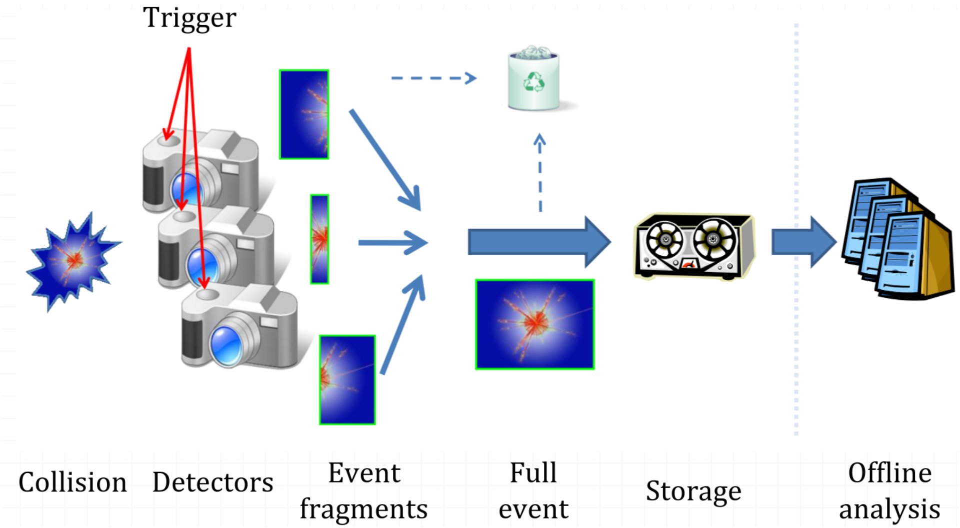


CERN-PHOTO-202206-116-1



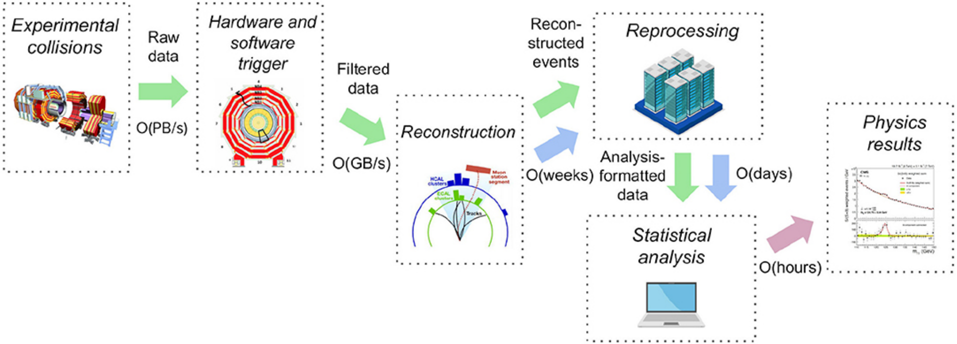
public-archive.web.cern.ch

Typical data processing at HEP experiments



W. Pokorski

Typical data processing at HEP experiments

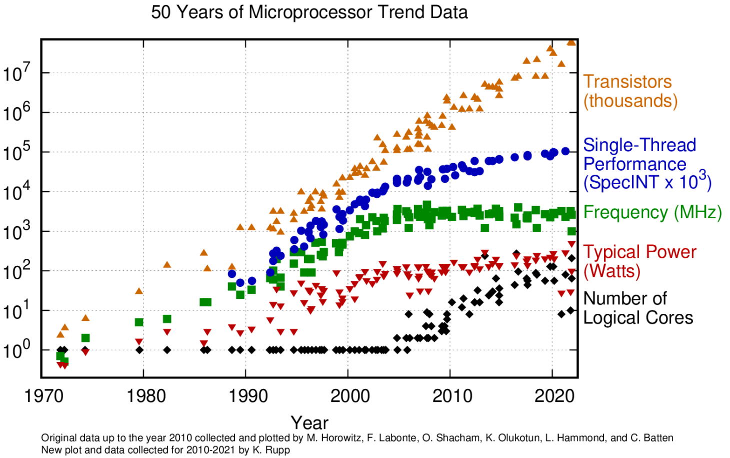


Front. Big Data, 07 May 2021 — doi.org/10.3389/fdata.2021.661501

HEP software

- Software
  - Estimated to be around 50 M lines of C++
  - Which would cost more than 500M $ to develop commercially
- Computing
  - LHC experiments use about 1 M CPU cores every hour of every day
  - we have around 1000 PB of data
  - with 100 PB of data transfers per year (10-100 Gb links)

[G. Stewart]



github.com/karlrupp/microprocessor-trend-data

Just a few software packages in use

GEANT4
A SIMULATION TOOLKIT

ROOT
Data Analysis Framework

FLUKA

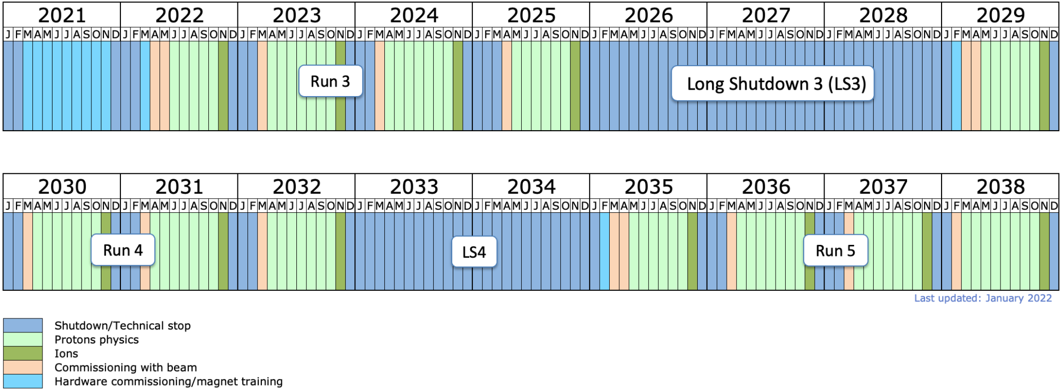
CLUE

DD4hep

acts

HL-LHC

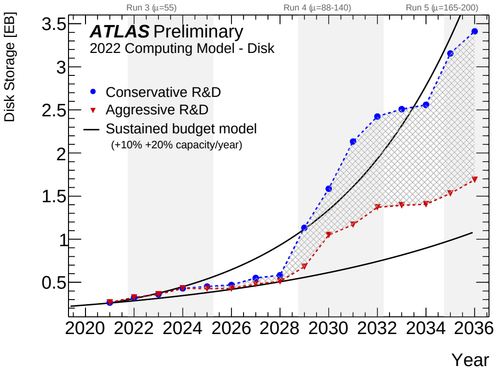
The High Luminosity Large Hadron Collider (HL-LHC) is an upgrade of the LHC which aims to achieve instantaneous luminosities a **factor of five larger** than the LHC nominal value, thereby enabling the experiments to **enlarge their data sample by one order of magnitude** compared with the LHC baseline programme.



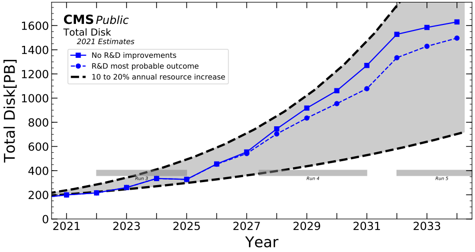
lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm

Computing challenges ahead

Disk storage



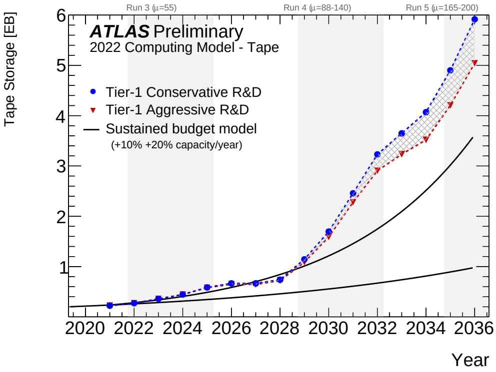
CERN-LHCC-2022-005



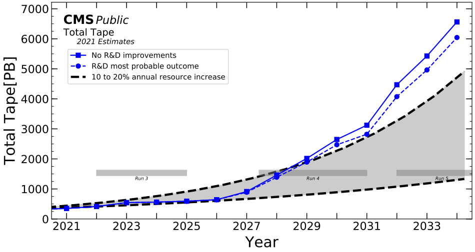
CMS public results

Computing challenges ahead

Tape storage



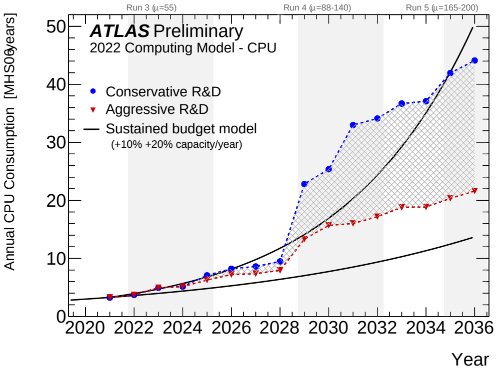
CERN-LHCC-2022-005



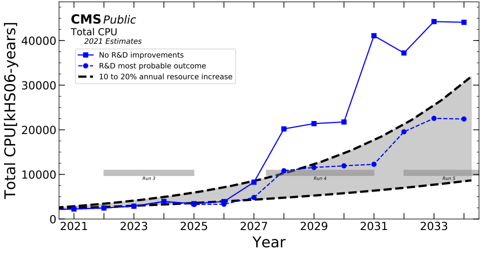
CMS public results

Computing challenges ahead

CPU time



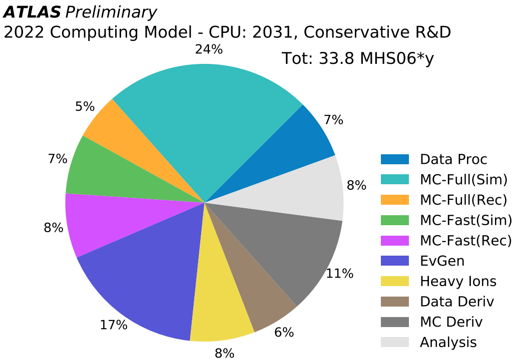
CERN-LHCC-2022-005

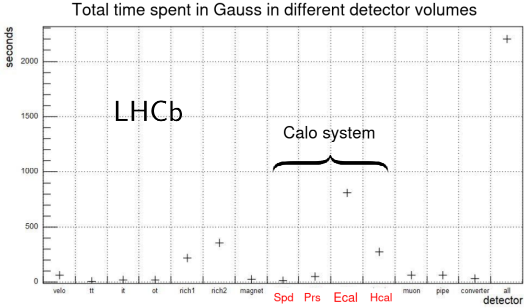


CMS public results

Computing challenges ahead

CPU time: simulation



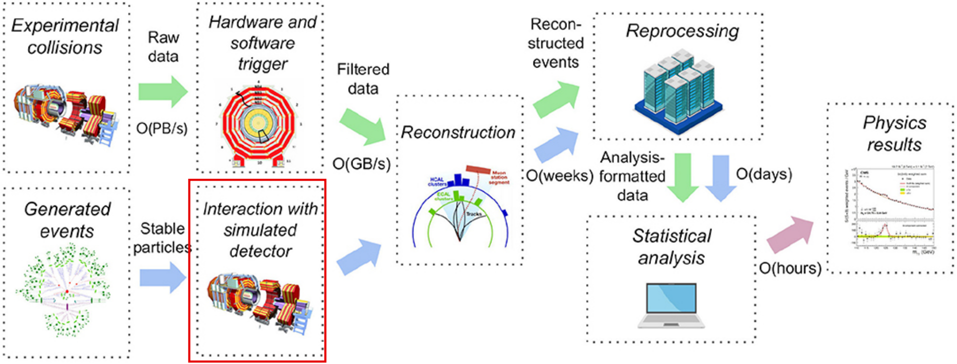


CPU time in calorimeter system: ~ **53**%

C. Rama, 2018

CERN-LHCC-2022-005

Simulation of particle passage through matter

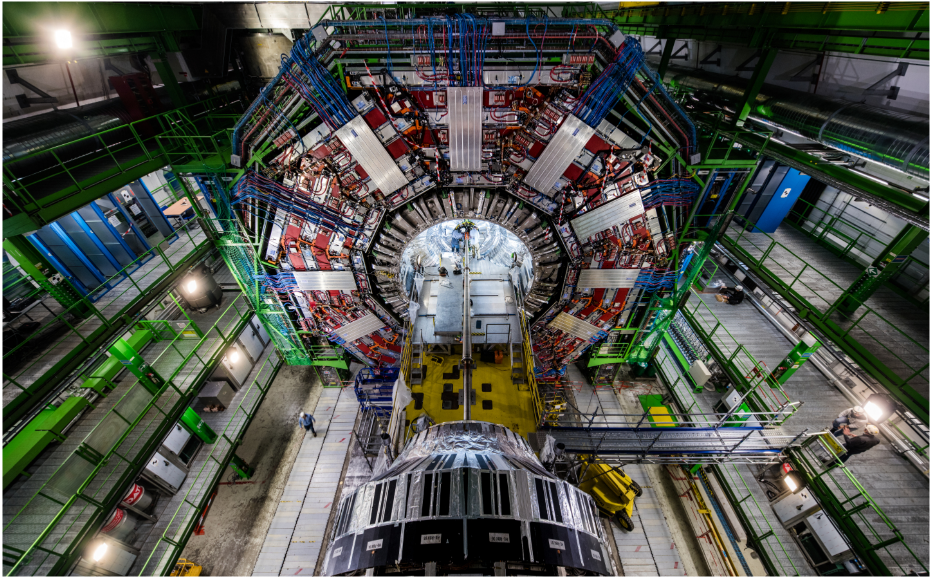


Front. Big Data, 07 May 2021 — doi.org/10.3389/fdata.2021.661501

Why do we need simulation?

- Before building the detector - to design it;
- To operate the detector - background simulation, ...;
- For data analysis - to understand known and new phenomena;

HEP detector: CMS

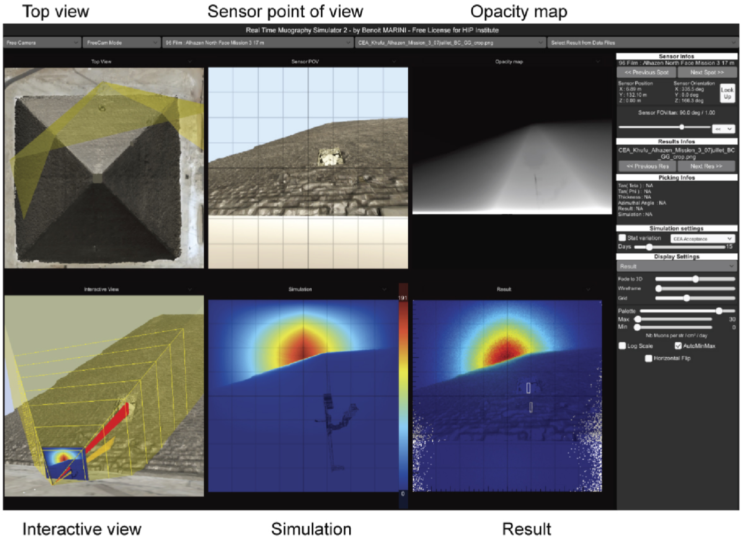


CERN-PHOTO-201703-062-52

Simulation beyond HEP

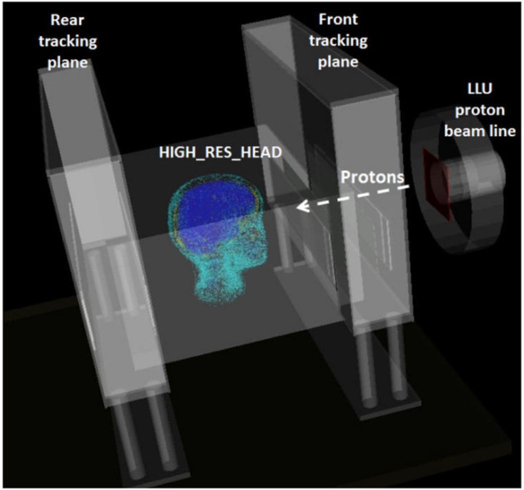
CMS-PHO-GEN-2017-009-6

Simulation beyond HEP



Nature volume 552, pages 386–390 (2017)

Simulation beyond HEP



Phys. Med. 33 (2017)

Simulation beyond HEP

XMM Newton X-Ray telescope

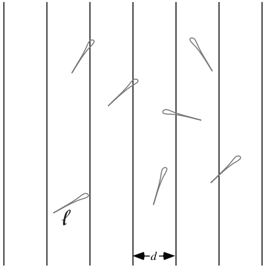
What is simulation?

A “virtual” experiment.

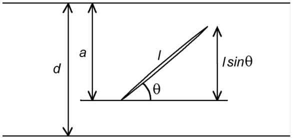
- Take known physics;
- Start from initial conditions (particles, materials, ...);
- Calculate final conditions;

Analytically? ... No!

Buffon's needle



Wolfram



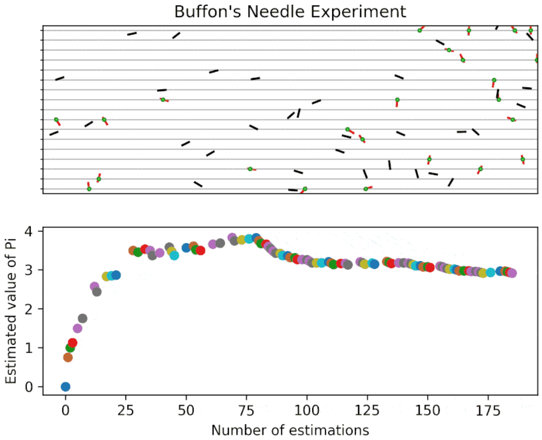
G. Holton, Value-at-Risk

$$a \leqslant l \sin\theta$$

- One of the oldest problems in the field of geometrical probability, first stated in 1777.
- Drop a needle on a lined sheet of paper and determine the probability of the needle crossing one of the lines.
- For $l < d$:

$$P = \int_{\theta=0}^{pi} \int_{a=0}^{l\sin\theta} \frac{1}{d\pi}\, da\, d\theta = \frac{2l}{d\pi}$$

Buffon’s needle

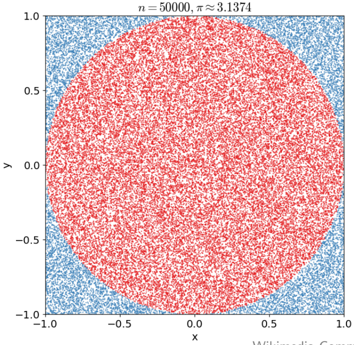


Probability P is directly related to the value of π.

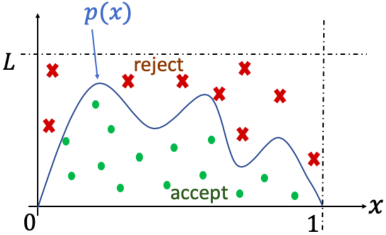
pi estimation

- Described by Laplace in 1886.
- Take circle of radius $r = 1$ inscribed in a square.
- Probability of random points falling inside the circle: $P = \frac{\pi}{4}$
- How to estimate π?
    - Draw uniformly N random points (x, y) from $(-1, 1)$ range.
    - Count the C points for which $x^2 + y^2 < 1$.
    - The ratio $\frac{C}{N}$ converges towards $\pi/4$.

- Can be easily extended to any distribution



Wikimedia Commons

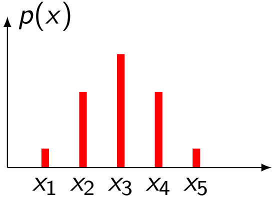


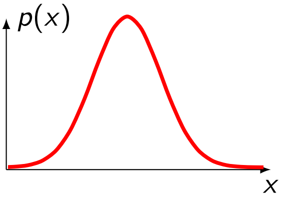
M. Loem, Towards Data Science

Random processes

Random (stochastic) processes are widely used as mathematical models of phenomena that appear to vary in a random manner.

- Result cannot be specified in advance of observing it.
- Probabilities are used to describe the process.
- Discrete processes:
  - throwing a dice ;
  - selecting a decay channel for an unstable particle;
  - described by probabilities of events;
- Continuous processes:
  - decay time of an unstable particle;
  - described by probability density function (PDF);





Simulation of stochastic processes

Simulation of natural (stochastic) laws = reproduction of probability distributions.

Generation of samples that follow probability distributions (f(x)) means throwing (many) random numbers.

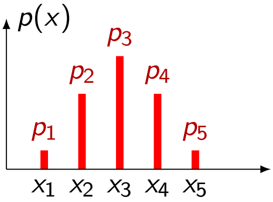
A sequence of random numbers is a set of numbers that have nothing to do with the other numbers in the sequence.

In computer programs we use pseudo-random numbers.
Linear congruential generator:

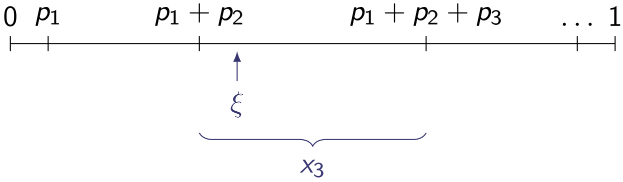
$$I_{n+1} = (aI_n + c) \mod m$$

e.g. $m = 2^{31}$, $a = 1103515245$, $c = 12345$ (for glibc (gcc) implementation)

Discrete sampling

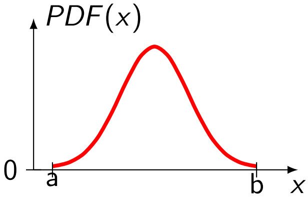


$$\sum_{i=1}^{5} p_i = 1$$

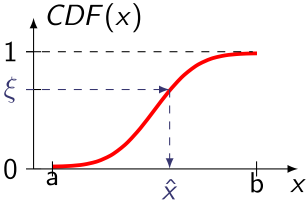


1. Split $[0, 1]$ into intervals corresponding to discrete probabilities p_i.
2. Generate a random number $\xi \in [0, 1]$.
3. Look in which interval it falls.

Continuous sampling - direct method



$$\int_a^b p(x)x = 1$$



$$P(x) = \int_a^x p(x)\mathrm{d}x$$

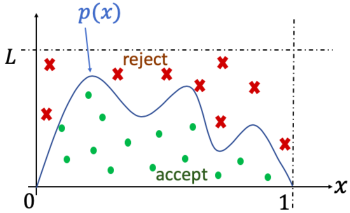
1. For probability density function $p(x)$ find the cumulative density function $P(x)$.

2. Generate a random number $\xi \in [0, 1]$.

3. Find $x = \hat{x}$ for which $P(x) = \xi$.

$$\hat{x} = P^{-1}(\xi)$$

Continuous sampling - accept–reject method



M. Loem, Towards Data Science

1. Generate two random numbers $\xi_x \in [0,1]$ and $\xi_y \in [0,1]$.
2. Scale them if necessary: $x_i = \xi_x,\ y_i = L\xi_y$.
3. If $y_i > p(x_i) \Rightarrow$ reject x_i,
   If $y_i \leqslant p(x_i) \Rightarrow$ accept x_i.
4. Fraction of accepted points is equal to fraction of area below curve $p(x)$.

Simple example: Particle decay in flight

- Spontaneous process of an unstable particle.
- Decay time is a random value with probability density function

$$f(t) = \frac{1}{\tau} \exp\left(-\frac{t}{\tau}\right), t \geqslant 0$$

  τ is the mean life of particle

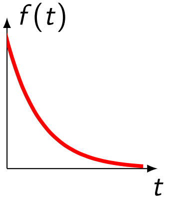
- Probability that particle decays before time T is given by CDF

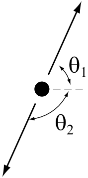
$$F(t) = 1 - \exp\left(-\frac{t}{\tau}\right)$$

  which can be used to sample directly.

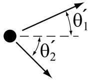
$$\xi_1 \in [0, 1] \to \hat{t} = \tau \ln(1 - \xi_1)$$

- Random angles are chosen for decay products, e.g. θ_1 and θ_2 (in rest frame).
- Decay products are boosted to lab frame.





JabberWok, Wikipedia



JabberWok, Wikipedia

Monte Carlo methods

- Monte Carlo name coined by Ulam and Metropolis in 1949 (Manhattan project).
- Recognition of newly invented computer power to application of statistical sampling to solve .
- Metropolis (1948): First actual Monte Carlo calculations using a computer (ENIAC).
- Berger (1963): First complete coupled electron-photon transport code that became known as ETRAN.
- Exponential growth since the 1980's with the availability of computers.

Monte Carlo codes

Non-exhaustive list of Monte Carlo codes

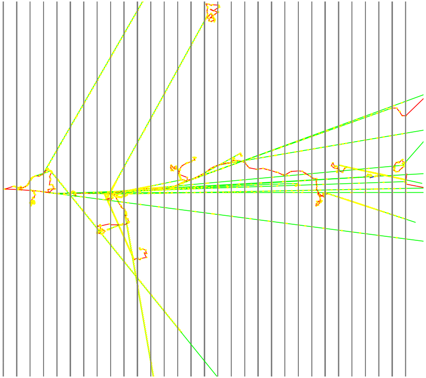
- EM physics
  - ETRAN (Berger & Seltzer; NIST)
  - EGS4 (Nelson, Hirayama, Rogers; SLAC)
  - EGS5 (Hirayama et al.; KEK/SLAC)
  - EGSnrc (Kawrakow & Rogers; NRCC)
  - Penelope (Salvat et al.; U. Barcelona)
- Hadronic physics / general purpose
  - Fluka (Ferrari et al., CERN/INFN)
  - **Geant4** (Geant4 Collaboration)
  - MARS (James & Mokhov; FNAL)
  - MCNPX / MCNP5 (LANL)
  - PHITS (Niita et al.; JAEA)

Geant4

Geant4 is a toolkit used to simulate particle passage through matter.

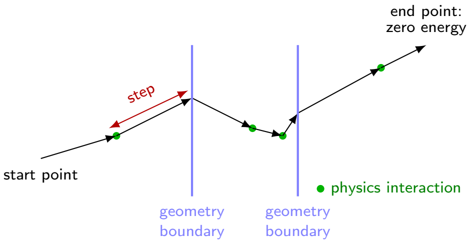
- Non-deterministic:
  - no equations to be solved,
  - use of random numbers to reproduce distributions.
- General code:
  - allows to describe different geometries (shapes, materials),
  - contains distributions describing various physics processes.
- Finds application in many areas:
  - high energy physics,
  - astrophysics,
  - medical physics,
  - industry.
- Toolkit:
  - no `main` program, set of tools that allows to build user applications.

GEANT4 A SIMULATION TOOLKIT



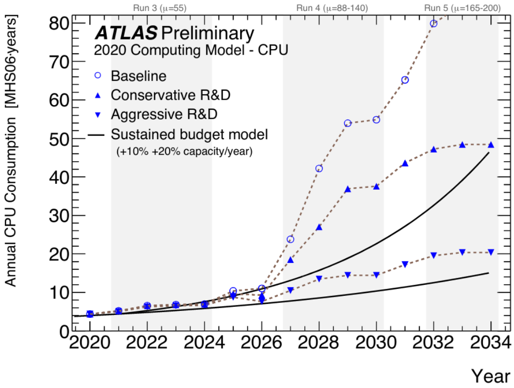
How does simulation work?

- Track one particle at a time.
- Consider particle passage in steps.
- For each step:
  - Determine **step length** and **interaction** (if any)from cross-sections (probabilities) of physics processes, and geometrical boundaries.
  - **Deposit** energy.
  - If physics process creates **new particles**, add them to the list.
  - **Move** particle to new position, taking into account electromagnetic field.
  - If particle has energy $E > 0$ and is still within detector ("world"), **repeat**. Otherwise, take new particle.

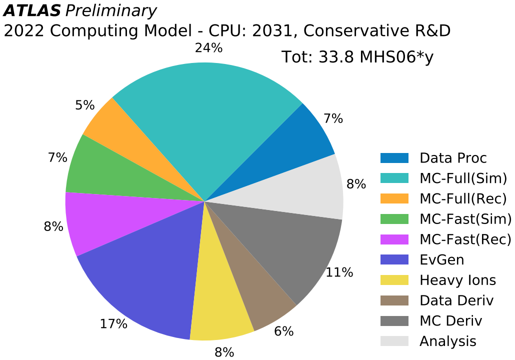


Computing challenges ahead

CPU time: simulation

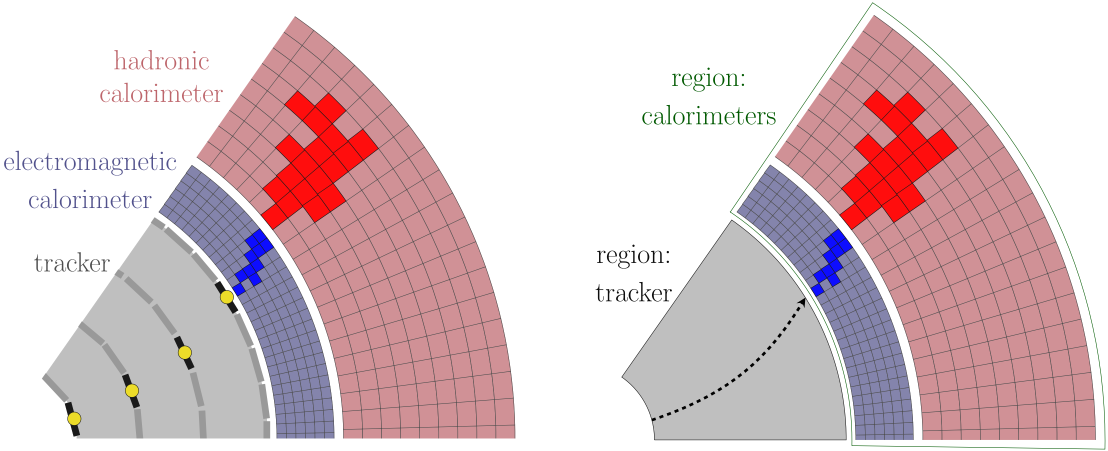


CERN-LHCC-2022-005



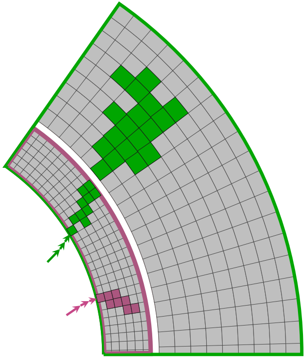
CERN-LHCC-2022-005

How we simulate particles?

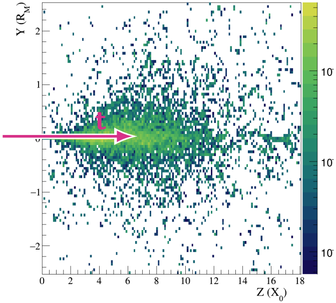


Fast simulation is a shortcut to the standard tracking and detailed simulation.

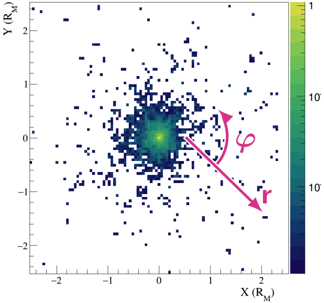
„Classical" shower parameterisation



longitudinal profile

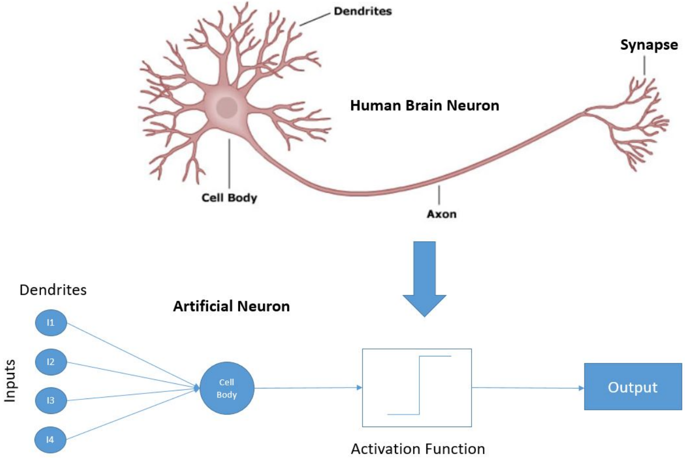


lateral profile



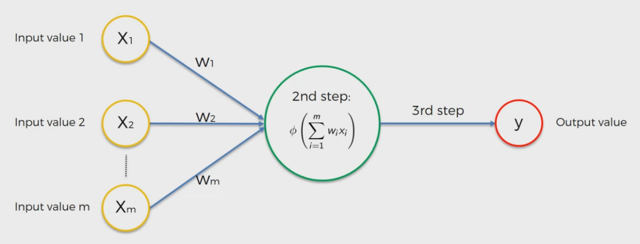
$$f(t,r,\varphi)=f(t)f(r)f(\varphi)$$

Neural networks

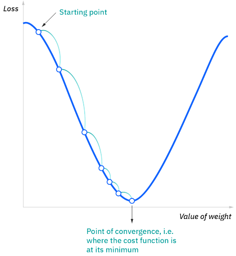


source

Model training



source



source

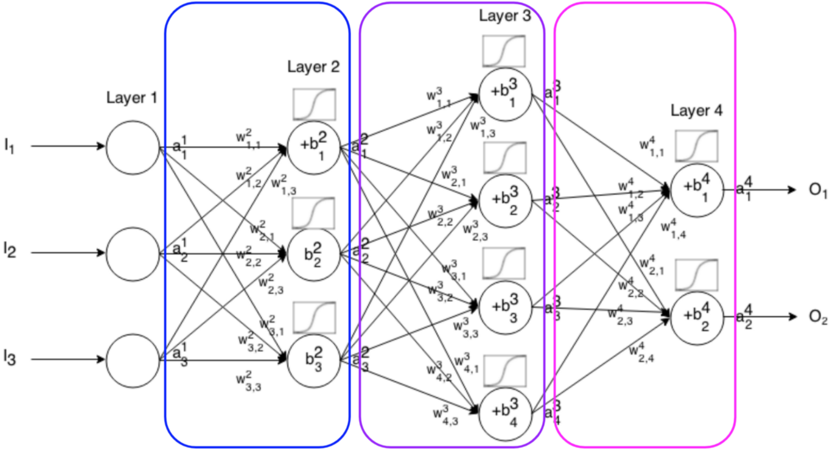
1st step: Pass independent input variables and assign weights

2nd step: Apply activation function to the sum of inputs and weights. Neurons learn which signal to pass.

3rd step: Generate output and calculate cost function to provide feedback to assignment of weights.

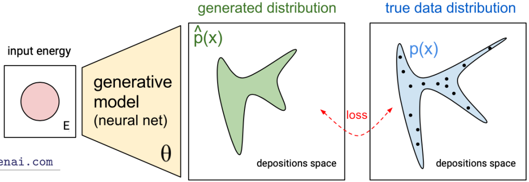
Goal: Train network to provide best prediction.

Neural networks



source

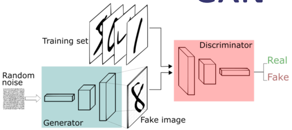
Fast simulation with generative models



openai.com

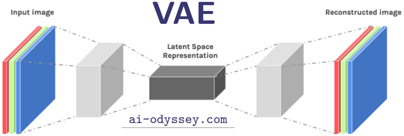
- Recent work mostly on ML-aided fast simulation;
- Generative networks;
- 'Learning' data in the training process, then generation of new samples;

GAN



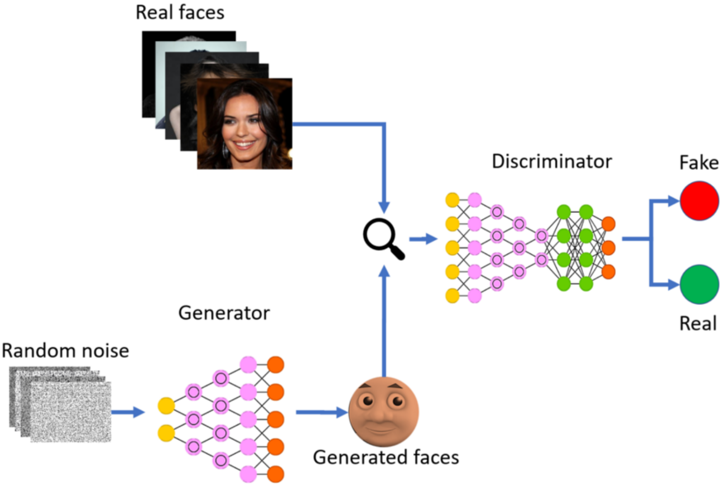
freecodecamp.org, T. Silva

VAE



ai-odyssey.com

Generative adversarial network (GAN)

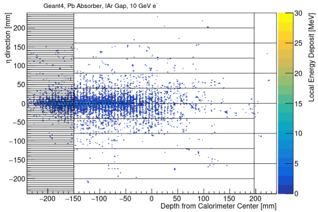


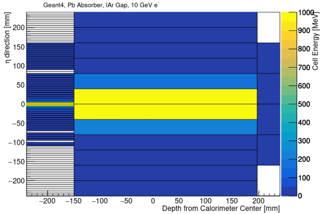
kaggle.com

CaloGAN

M. Paganini et al. (2017)

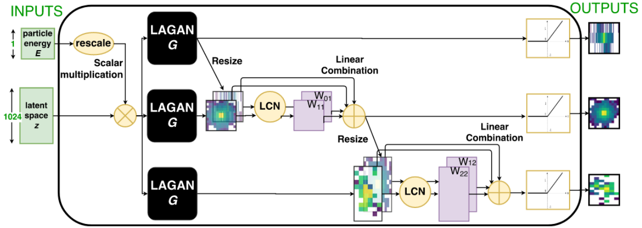
- ATLAS inspired detector geometry
  - Three layers; different depth and cell size
  - No accordion shape, predefined window
- Trained to generate energies in cells
  - Layer0: 3x96
  - Layer1: 12x12
  - Layer2: 12x6
- Particle incident perpendicular to the centre of calorimeter
- Uniform energies between 1 and 100 GeV for three particle types
  - Separate model per particle ($e^{\pm}$, γ, $\pi^{\pm}$)

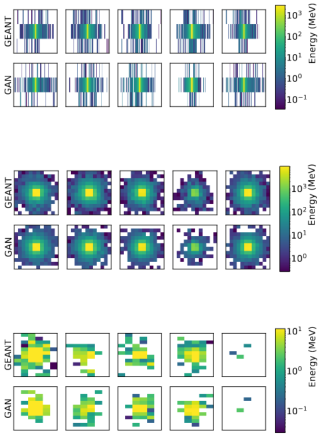




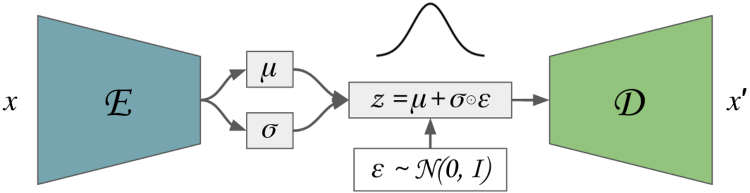
CaloGAN

M. Paganini et al. (2017)





Variational Auoencoder (VAE)

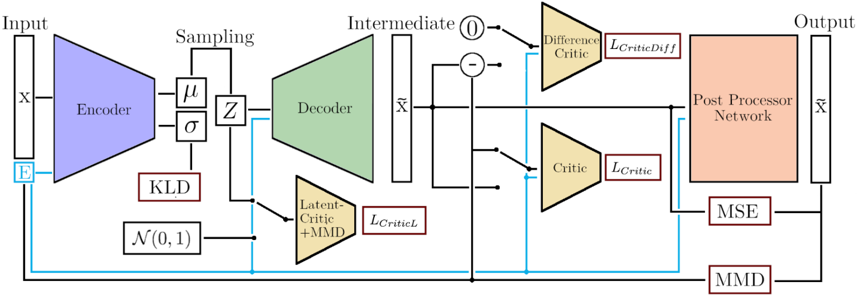


B. Dillon

BIBAE

E. Buhmann et al. (2005)

E. Buhmann (2021)

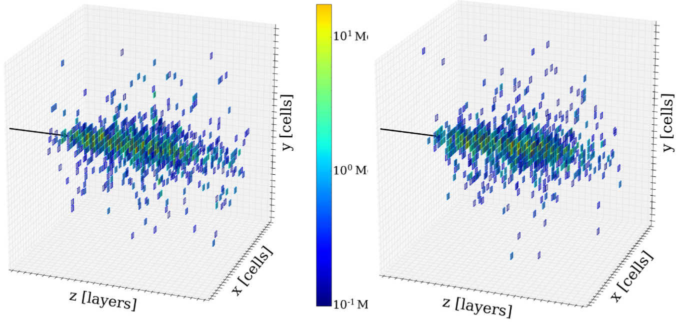


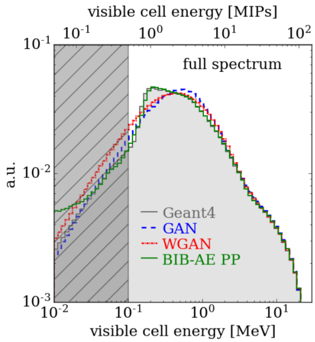
- Dataset from electromagnetic calorimeter for ILD
- 30 active silicon layers with 20 tungsten absorbers 2.1 mm thick followed by 10 absorbers 4.2 mm thick
- 5×5 mm^2 cell sizes and a rectangular grid of $30 \times 30 \times 30$ cells

BIBAE

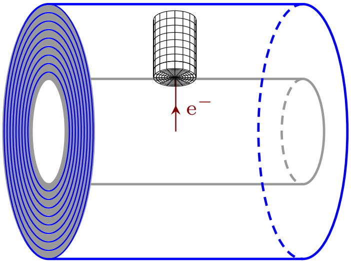
E. Buhmann et al. (2005)

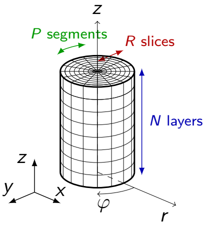
E. Buhmann (2021)

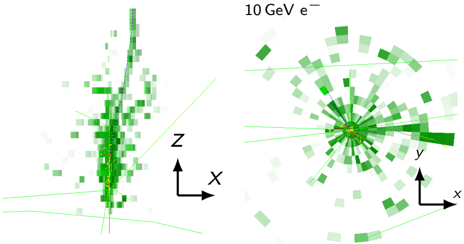




Geant4 Par04 example

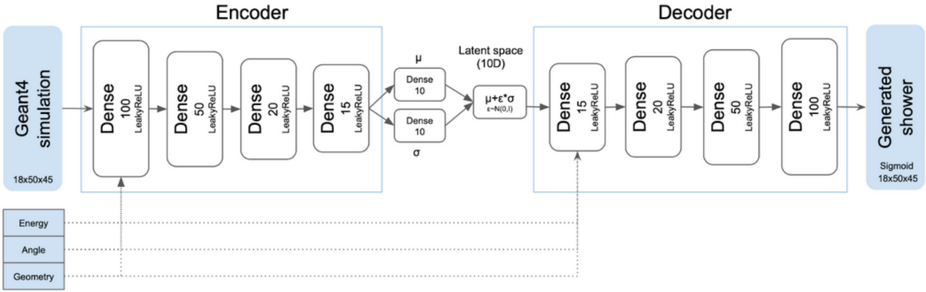




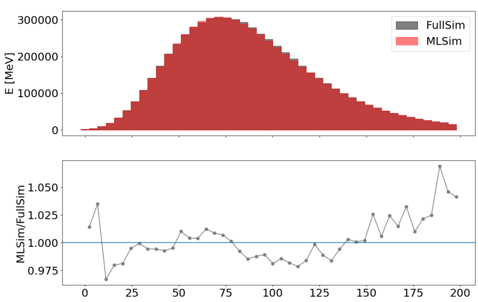


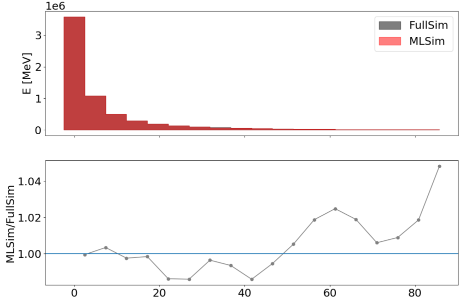
- `examples/extended/parameterisations/Par04` available in Geant4 11.0 release (Dec 2021)
- Energy scored independently on detector readout in cylindrical mesh around the incident particle achieving highly granular shower outputs
- Cell dimensions expressed in units of X_0 and R_M: $\Delta r \approx 0.75 X_0$, $\Delta z \approx 0.25 X_0$
- EM showers simulated for different materials (in current study: Si-W, scintillator-Pb, used to pre-train neural network (variational autoencoder)
- Full sim used to produce training and validation data dataset is available on Zenodo

VAE within Geant4 Par04 example



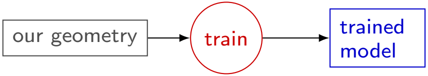
Physics performance studies



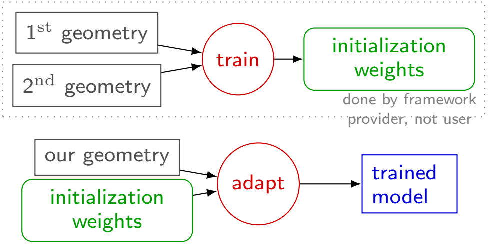


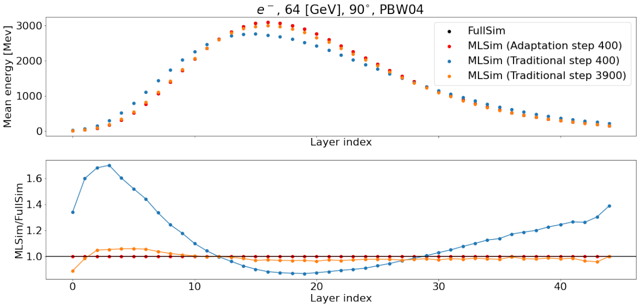
Meta-learning

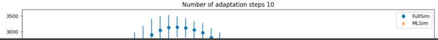
‘Traditional‘ ML training



Meta-learning







Fast Calorimeter Simulation Challenge 2022

View on GitHub

calochallenge.github.io/homepage/

Welcome to the home of the first-ever Fast Calorimeter Simulation Challenge!

The purpose of this challenge is to spur the development and benchmarking of fast and high-fidelity calorimeter shower generation using deep learning methods. Currently, generating calorimeter showers of interacting particles (electrons, photons, pions, ...) using GEANT4 is a major computational bottleneck at the LHC, and it is forecast to overwhelm the computing budget of the LHC experiments in the near future. Therefore there is an urgent need to develop GEANT4 emulators that are both fast (computationally lightweight) and accurate. The LHC collaborations have been developing fast simulation methods for some time, and the hope of this challenge is to directly compare new deep learning approaches on common benchmarks. It is expected that participants will make use of cutting-edge techniques in generative modeling with deep learning, e.g. GANs, VAEs and normalizing flows.

This challenge is modeled after two previous, highly successful data challenges in HEP – the top tagging community challenge and the LHC Olympics 2020 anomaly detection challenge.

Timeline

The challenge will conclude approximately 1 month before the next ML4Jets conference (currently tentatively scheduled for the week of December 5, 2022). Results of the challenge will be presented at ML4Jets, and the challenge will culminate in a community paper documenting the various approaches and their outcomes.

Please do not hesitate to ask questions: we will use the ML4Jets slack channel to discuss technical questions related to this challenge. You are also encouraged to sign up for the Google groups mailing list for infrequent announcements and communications.

Good luck!

Michele Faucci Gianelli, Gregor Kasieczka, Claudius Krause, Ben Nachman, Dalila Salamani, David Shih and Anna Zaborowska

Thank you!

anna.zaborowska@cern.ch