UNIVERSIDADE DA CORUÑA

OPORTUNIUS RESEARCH PROFESSOR

# COMCHA @ UdC

Diego Martinez Santos, CITENI, UdC

Offline analysis tools:

## Basic idea

CPU

Matrix of data points

Binning (for numeric integral)

Pdf(t, Ω | {fit_params})
~~Pdf(t,| {fit_params})~~

(Minimizing I/O is very important here)

## Basic idea FCN

CPU

Values for {fit_parameters}

Pdf(t, Ω | {fit_params})
Pdf(t,| {fit_params})
Data

Binning for numeric integral

(Minimizing I/O is very important here)

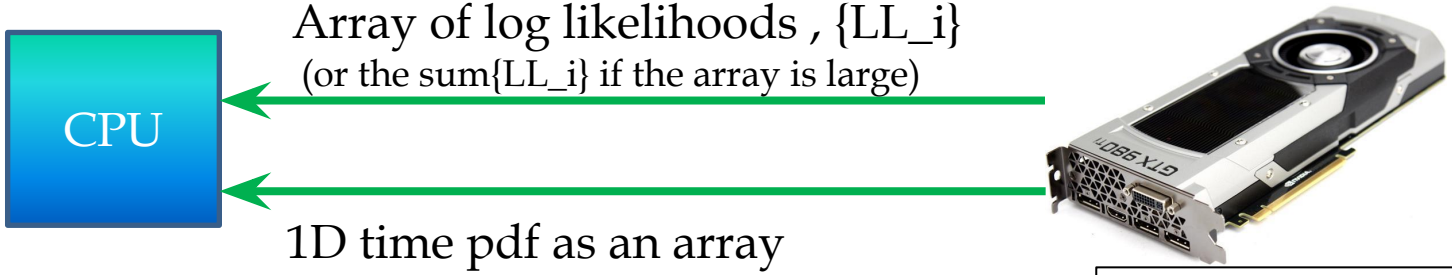
## Basic idea FCN

CPU

(Minimizing I/O is very important here)

- Likelihood for each data point
- 1D pdf for the time

## Basic idea FCN




(Minimizing I/O is very important here)

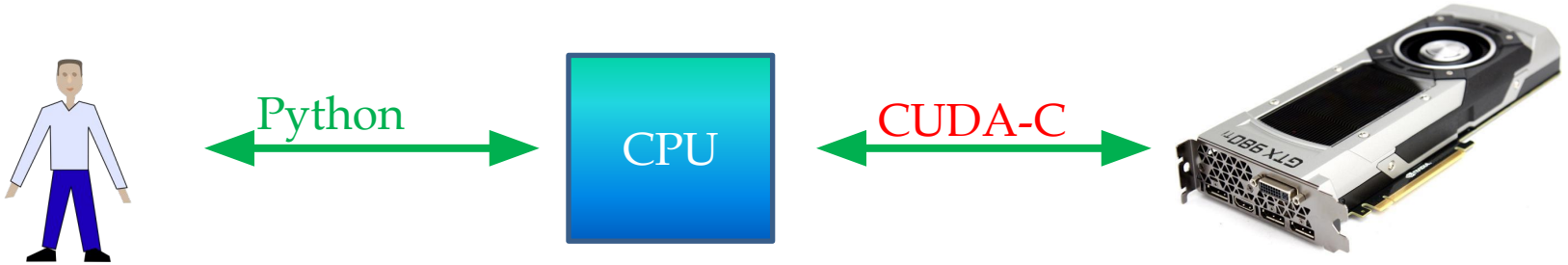
# Basic idea FCN

CPU


- Σ (LL_i)
  (if not done @GPU)
- Σ (pdf)  (normalization)

FCN  Minuit

## Basic idea. Programing language(s)

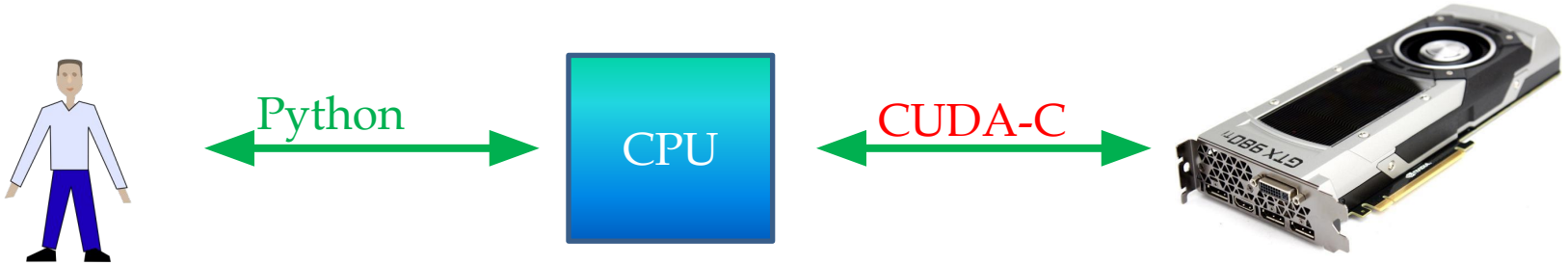



Organic CPU

I've converged towards :

- Pdf's implemented as `__device__ void` Cuda-C functions (stay in the GPU)
- Likelihood evaluation/toy generation as __global__
- Fit framework/configuration/OO/etc in Python. (pyCUDA + python analysis software + custom classes for parameter bkk, etc…)

## Basic idea. Programing language(s)



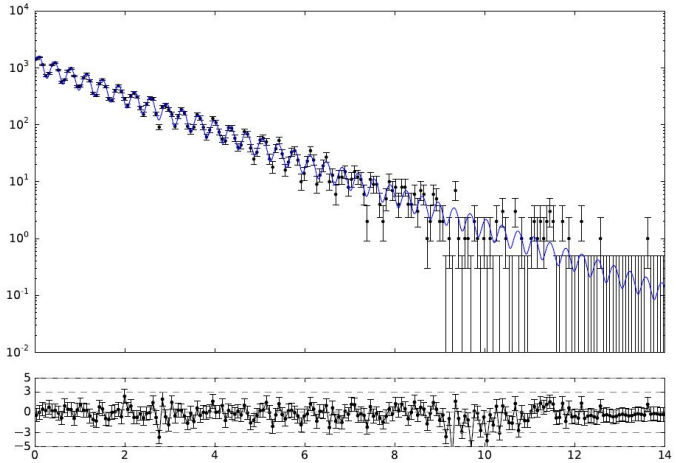

- Global/external constraints also at the python level (no point on parallelizing them):

```
FCN +=  (( np – mean_np)/sigma_np) **2
```

# Observed timings J/ψ KK

All parameters free, numeric integral for the time (even if not necessary). Call Migrad and Hesse (not Minos, otherwise the CPU-only fit takes ages)

- 100K events:
  - 1 KK mass bin (□ 2 categories, 10 free params): < 4 sec. (771 calls)
    ~same in CPU (C++) : 520 sec (□ > 100x gain !) (530 calls)



| | Name | Value | Para Err |
|---|---|---|---|
| 0 | fL = | 0.5043 | 0.001729 |
| 1 | fpe = | 0.2461 | 0.001706 |
| 2 | phis = | 0.7986 | 0.006126 |
| 3 | dpa = | 1.565 | 0.009001 |
| 4 | dpe = | -1.567 | 0.01302 |
| 5 | G = | 0.6595 | 0.001602 |
| 6 | DG = | 0.08441 | 0.00541 |
| 7 | DM = | 17.7 | 0.002613 |

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| fL | 0 | 1.00 | -0.45 | -0.24 | 0.00 | 0.00 | -0.09 | 0.32 | 0.00 |
| fpe | 1 | -0.45 | 1.00 | 0.33 | 0.00 | -0.00 | 0.14 | -0.33 | -0.00 |
| phis | 2 | -0.24 | 0.33 | 1.00 | 0.00 | -0.00 | 0.13 | -0.33 | 0.00 |
| dpa | 3 | 0.00 | 0.00 | 0.00 | 1.00 | 0.51 | 0.00 | -0.00 | 0.05 |
| dpe | 4 | 0.00 | -0.00 | -0.00 | 0.51 | 1.00 | -0.00 | 0.00 | 0.30 |
| G | 5 | -0.09 | 0.14 | 0.13 | 0.00 | -0.00 | 1.00 | -0.19 | -0.00 |
| DG | 6 | 0.32 | -0.33 | -0.33 | -0.00 | 0.00 | -0.19 | 1.00 | 0.00 |
| DM | 7 | 0.00 | -0.00 | 0.00 | 0.05 | 0.30 | -0.00 | 0.00 | 1.00 |

# Observed timings J/ψ KK

All parameters free, numeric integral for the time (even if not necessary). Call Migrad and Hesse (not Minos, otherwise the CPU-only fit takes ages)

- 100K events:
  - 1 KK mass bin (□ 2 categories, 10 free params): < 4 sec. (771 calls)
    ~same in CPU (C++) : 520 sec (□ > 100x gain !) (530 calls)

- **10 M events**:
  - 1 KK mass bin (□ 2 categories, 10 free params): ~< 1 minute (612 calls)
  - **6 KK mass bins** (□ 12 categories, 20 free params): **~ 3.5 min** (2882 calls)

## Summary

Repository: git clone https://gitlab.cern.ch/bsm-fleet/Ipanema.git (to be migrated to python3) .

Documentation: https://arxiv.org/abs/1706.01420

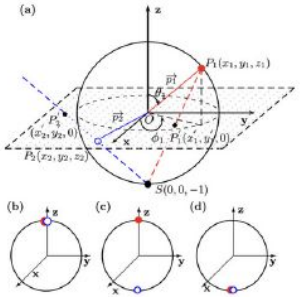
- Examples of how to combine python packages for HEP analysis in CUDA
  - Maximun likelihood fits, CLs limits …
  - toyMC generation
- Bookkeeping classes to handle parameters, plots, and interfaces to
  - Minuit
  - Sampling ( via `multinest` package, others to be added)
  - The custom classes we created are fairly simple, you can inherit/modify/rewrite at your will

# Quantum Computing@UdC-LHCb

## Qutrits for physics at the LHC

Miranda Carou[1] Veronika Chobanova[2] Miriam Lucio Martinez[3]

https://arxiv.org/abs/2510.14001



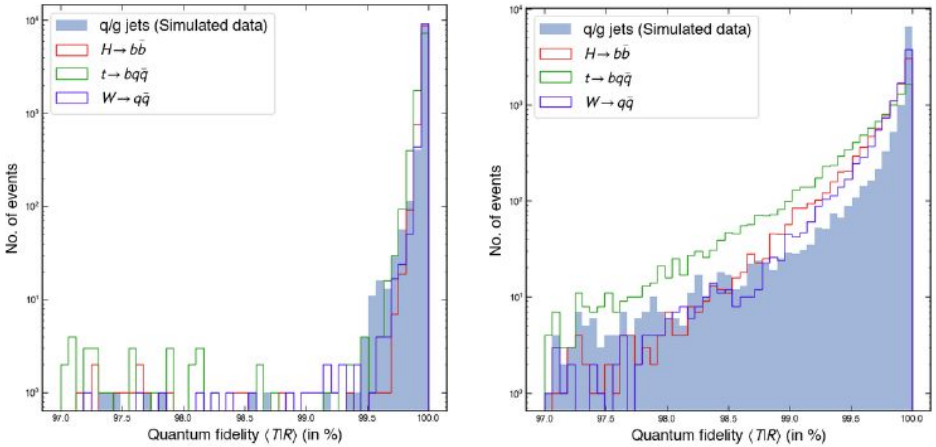
**Majorana qutrit representation.**

https://arxiv.org/abs/1703.06102

Advantages
- Higher storage capacity ✓
- Richer set of operations ✓
- Real hardware posibilities ✓

## Results

Qubits VS Qutrits




| Model | $W \to q\bar{q}$ | $H \to b\bar{b}$ | $t \to bq\bar{q}$ | Train data |
|---|---|---|---|---|
| 4-Qb Our Ref. | 0.7343 | 0.7755 | 0.8460 | Simulated |
| 4-Qt ($\varphi$, $\vartheta$, $\varrho_0$, $\varrho_z$) | 0.7258 | 0.7677 | 0.8366 | Simulated |

Table 2: AUC Scores for the different models with flat distribution in jet $p_T$, in the range [500, 1000] GeV. 'Qb' = Qubit Model, 'Qt' = Qutrit Model

## Other activities

RTA@LHCb

- Optimizations in luminosity counters
- GNN for calorimeters at trigger level
- Kaltman Filter in GPU

Reconstruction in GPU for HyperK

Data compression algorithms for KOTO-II