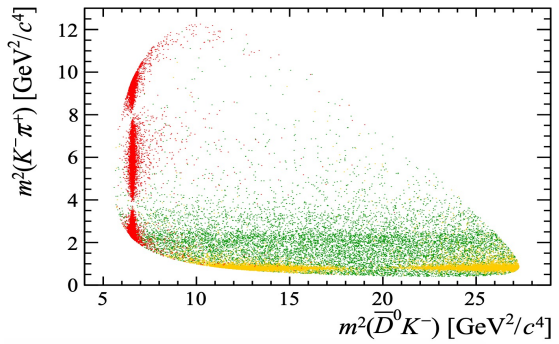
Amplitude and angular analysis

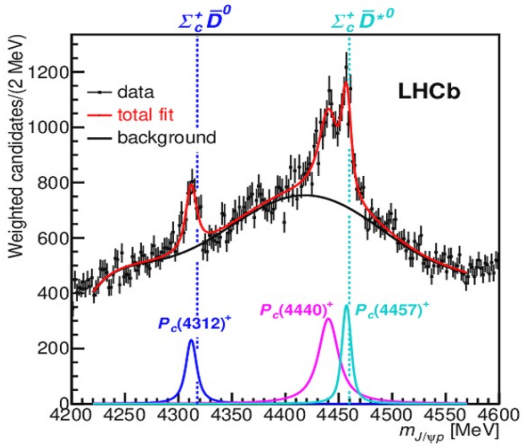
- (As evident from Tim's talk) Amplitude and angular analyses are important in studies of hadron spectroscopy, finding exotic states, CP violation (CPV), effects of BSM, etc.
- One can exploit the tools developed by a much broader Machine Learning (ML) community, to conduct such studies.

JCPC 2018 04 017



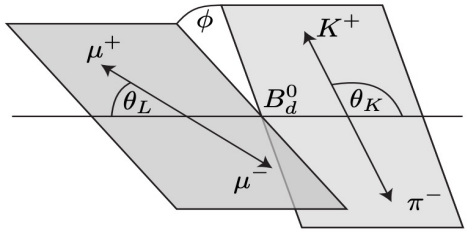
Spectroscopy study and CPV: Toy Dalits plot of $B_s^0 \to \overline{D}^0 K^- \pi^+$

Phys. Rev. Lett. 122.222001



Pentaquark discovery in $\Lambda_b \to J/\psi p K^-$

JHEP02(2016)104



Constrain BSM effects: Angular analysis of rare and semi-leptonic decays

ML software…

Lots of ML software's freely available:

- Scikit-learn
- PyTorch
- TensorFlow
- Keras
- Weka
- KNIME
- The list goes on…

Covers most of our basic needs:

- Written in C++, python and CUDA.
- API's available for several languages (see here), with python being the main one.
- A plethora of mathematical operations and functions for numerical and statistical analysis.
- **Flexibility in developing a model with compact and readable code.**
- **Clever optimisations of code.**
- **Can run on various heterogeneous computing architectures with small tweaks (multi-core CPUs, TPU, GPU, CPU/GPU farm).**

Tensor in TensorFlow

- Data in TF represented as an nD arrays with rows (number of events) and columns (observables to fit). **Easily scalable to multi-dimensions.**
- Bulk data can be **mapped** (e.g. probability at various points in phase space) or **reduced** (e.g observables integrated over phase space).

Flow in TensorFlow

Based on a **dataflow** paradigm that **speeds up computation** and **optimisation**. Here a program is modelled as directed flow of data between mathematical operations (***computational graph***), this allows for:

- **Evaluation of analytic gradients** (*automatic differentiation)* used by gradient based optimisers (e.g. Minuit).
- **Clever optimisations** (e.g. data caching, *common subgraph elimination* avoiding multiple computations of same object).

Hang on, there already HEP packages for amplitude analysis, why TF?

Issues with existing fitting frameworks

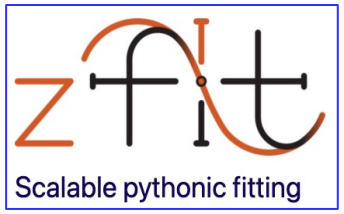
- Existing packages built and/or optimised to study specific set of decays.
- **Lack functionality and/or flexibility** to cover all cases encountered in amplitude/angular analyses.
- Significant alteration might be needed to accommodate outlying cases, e.g.:
  - Studying decays involving particles with non-zero spin (e.g. $\Xi_b^- \to pK^-K^-$).
  - Angular analysis with missing particles in final state (e.g. $\Lambda_b \to \Lambda_c^+ \mu^- \bar{\nu}_\mu$).
- TensorFlow provides **great flexibility** with model building without compramising too much on **speed of development** and **speed of computation**!

Fitting frameworks based on TensorFlow!

Packages where only TF is used as computational backend!

Packages with different computational backends: TensorFlow, Numpy, PyTorch, JAX

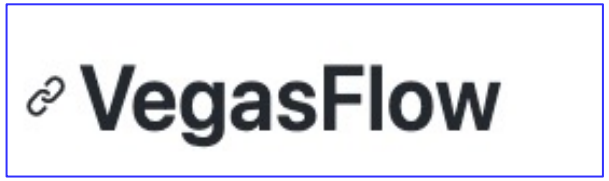
Roofit-like pkg [Webpage]



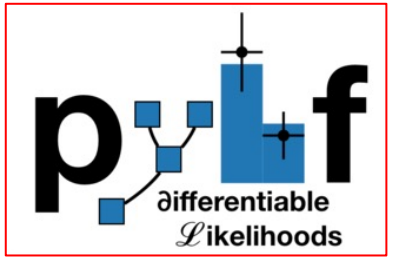
For amplitude analysis [Webpage]

For amplitude analysis [Webpage]

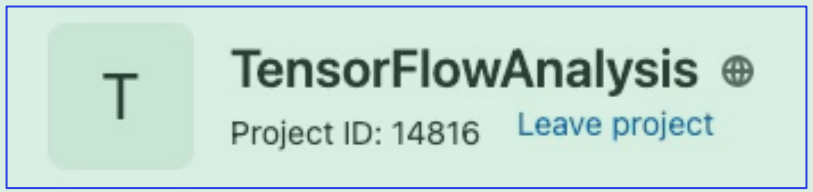
MC integration pkg [Webpage]



Template fitting [Webpage]



For amplitude analysis [gitlab]



Will talk only about TensorFlowAnalysis (TFA) with a ginue pig analysis ($\Xi_b^- \to pK^-K^-$)!

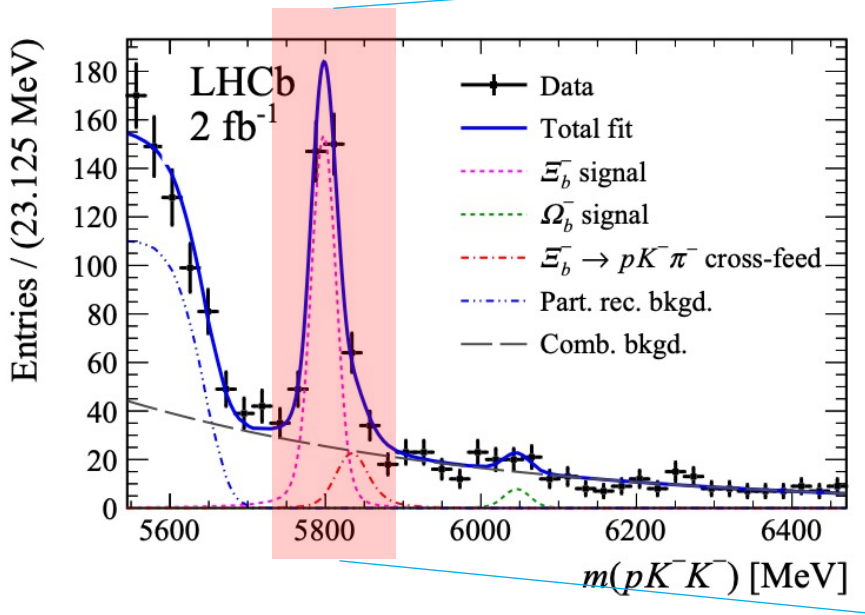
Guinea pig analysis

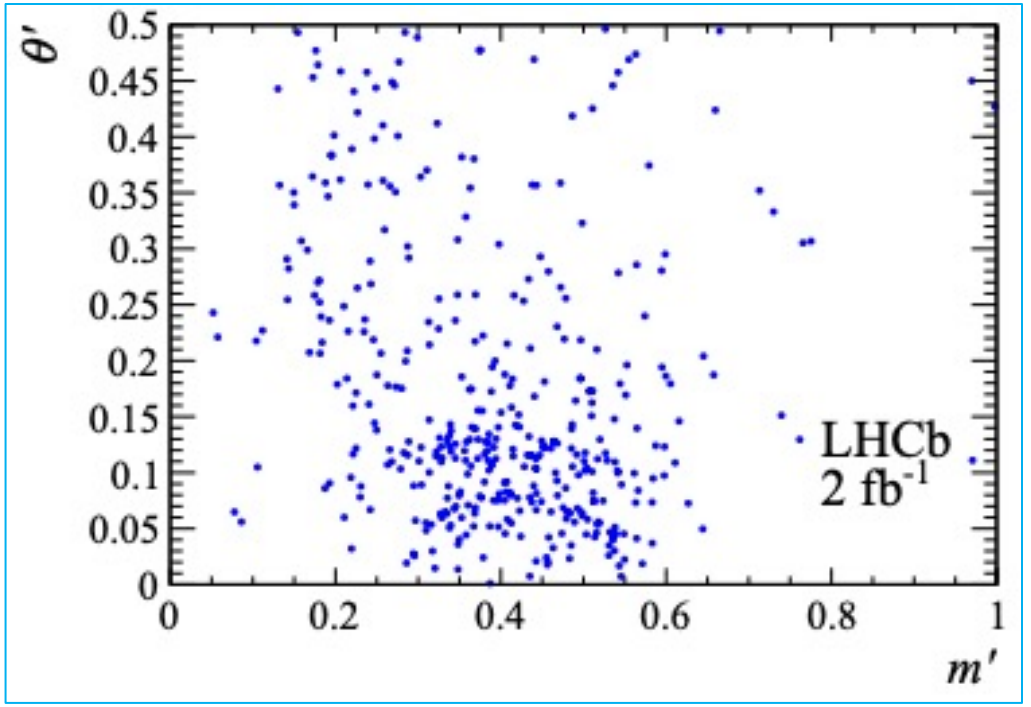
Phys.Rev.D 104 (2021) 5, 052010

Decays of $\Xi_b^- \to pK^-K^-$ can in principle exhibit large CP violating effects, so we can probe this via an amplitude analysis.

We select a region with ~500 decays and examine their phase space.

We need to first model this phase space through an amplitude analysis using TFA package!





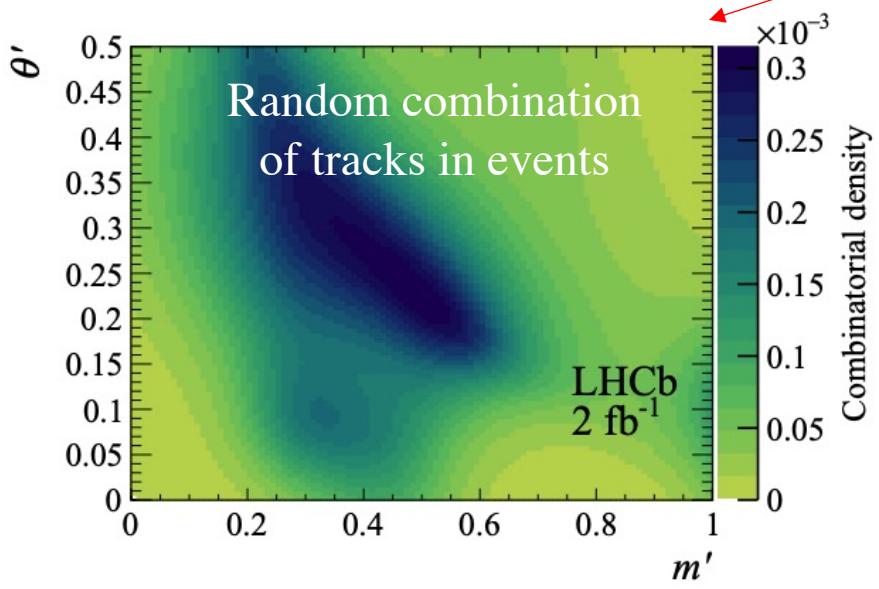
Modelling the phase space

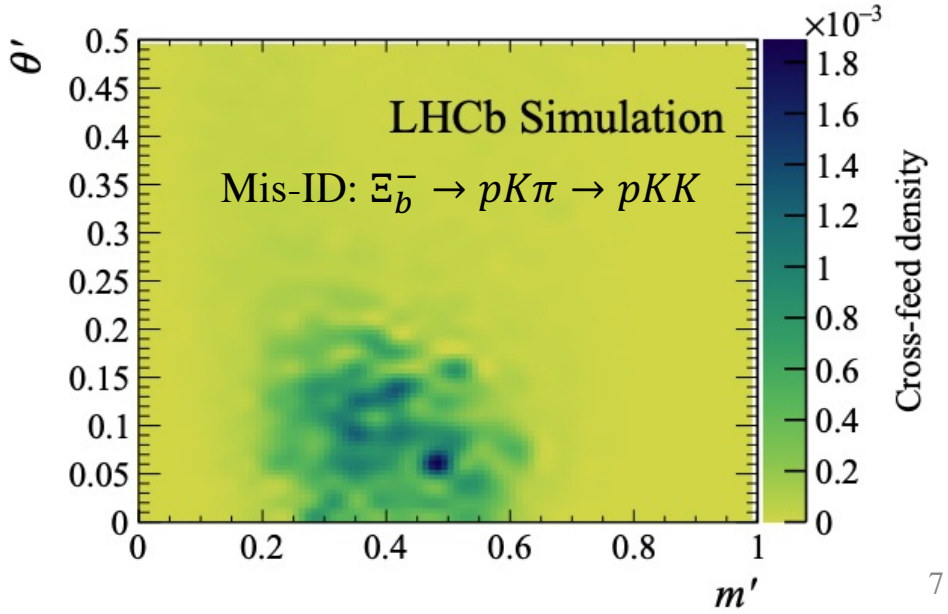
Phys.Rev.D 104 (2021) 5, 052010

$$\log \mathcal{L} = \sum_i \log \left(\int d\psi' \mathcal{G}(\psi', \psi_i) \varepsilon(\psi') \left(\frac{f_{sig}}{N_{sig}} \mathcal{P}_{sig}(\psi') + \frac{(1 - f_{sig})}{N_{bkg}} \mathcal{P}_{bkg}(\psi') \right) \right)$$

Non-physical bkgs: Modelled using artificial neural networks with TFA (see JINST 16 P06016 and talk).

Physical bkgs: Explicit model built using TFA.





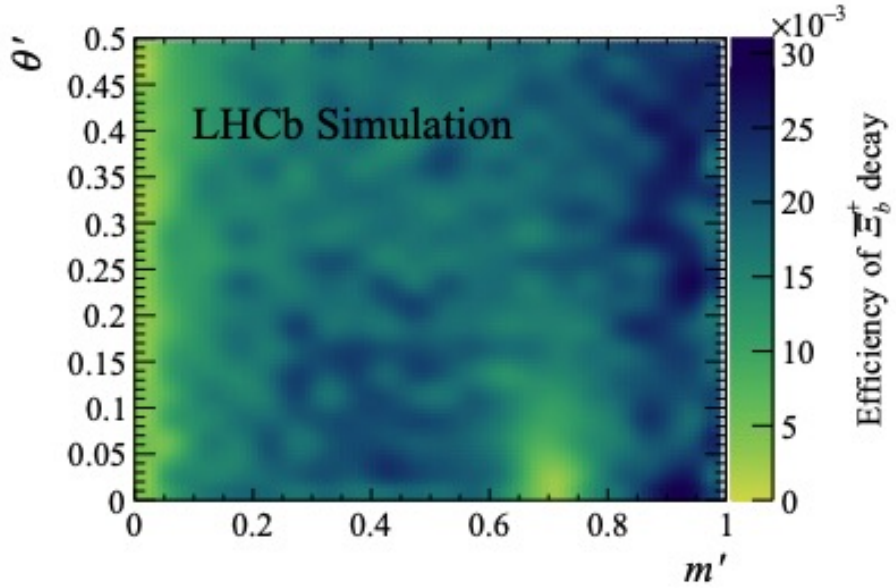
Modelling the phase space

Phys.Rev.D 104 (2021) 5, 052010

$$\log \mathcal{L} = \sum_i \log \left(\int d\psi' \, \mathcal{G}(\psi', \psi_i) \varepsilon(\psi') \left(\frac{f_{sig}}{N_{sig}} \mathcal{P}_{sig}(\psi') + \frac{(1 - f_{sig})}{N_{bkg}} \mathcal{P}_{bkg}(\psi') \right) \right)$$

Resolution effects convolved into PDF, using TF's fast foureir transfrom (FFT) algorithms. In the ginue pig analysis, we had very good resolution, so this was ignored.

Signal selection efficiency parametrised **using interpolation techniques** with TFA.



Modelling the phase space

Phys.Rev.D 104 (2021) 5, 052010

$$\log \mathcal{L} = \sum_i \log \left(\int d\psi' \, \mathcal{G}(\psi', \psi_i) \varepsilon(\psi') \left(\frac{f_{sig}}{N_{sig}} \mathcal{P}_{sig}(\psi') + \frac{(1 - f_{sig})}{N_{bkg}} \mathcal{P}_{bkg}(\psi') \right) \right)$$

- For signal model, **various required components supplied by TFA**:
  - Modelling of multi-dimensional phase space.
  - Functions to calculate relevant kinematic observables with four-vectors.
  - Modelling for intermediate components (resonant and nonresonant).
  - Monte-Carlo integration for normalisation of functions.
  - Generation of toys for fit validation.
- TFA can be interfaced with other packages for **optimisation with various minimisers** (e.g. iminuit, Ipyopt, NLopt, SciPy, TF).

Modelling the phase space

Phys.Rev.D 104 (2021) 5, 052010

To establish a signal model with data various approaches need to be taken.

Complex model (~150+ parameters)

Top-down approach

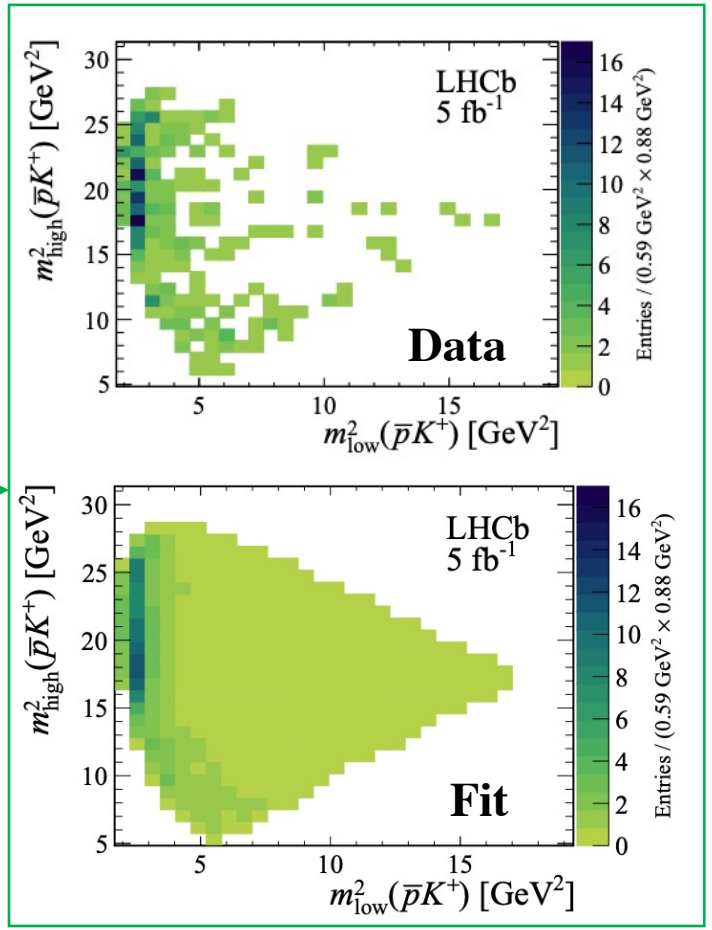
Nominal model (~50 parameters)

Bottom-up approach

Simple model (~8 parameters)

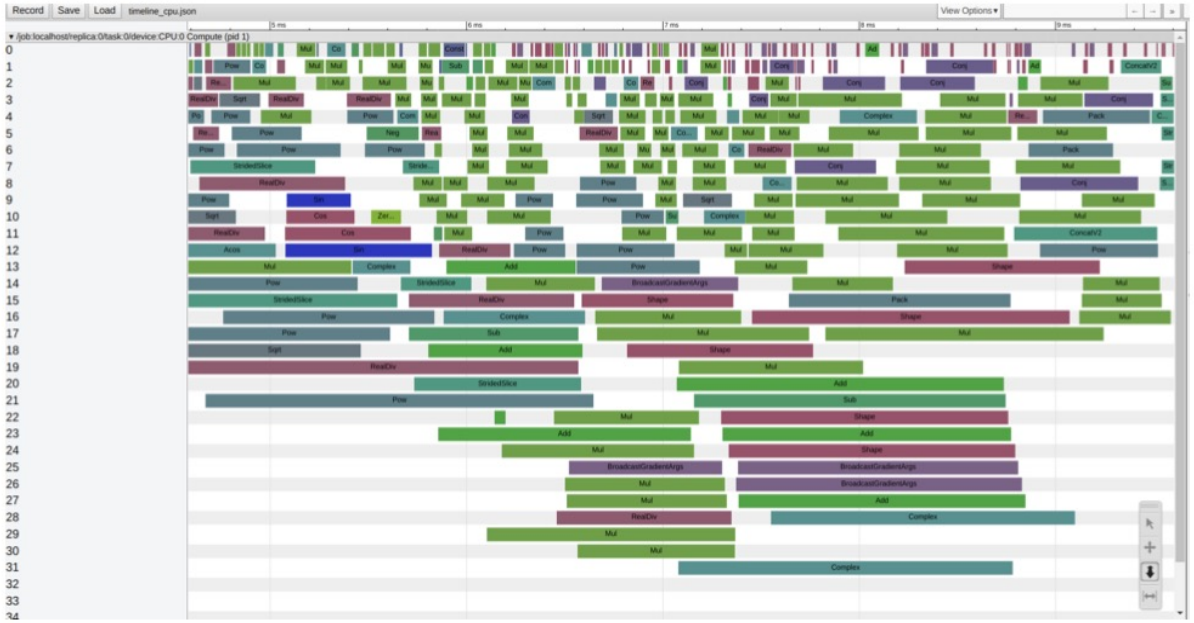
Establishing a model (and its assocated systematics) is the **most computing intesive task of the analysis**!

Where in the optimisation is time mostly spent?



Profiling with TF

Profiling feature allows to identify bottlenecks in execution speed.



Breakup of operations by CPU core (32-core Xeon).

Slide borrowed from Anton Poluektov [talk]

- The culprits are usually numerical integral evaluation at each minimisation step.
- Analytical integrals rarely available with these complex models.
- In the guinea pig analysis, **with complex model (150+ parameters) and with 500 events, single fit takes 4-5 hours.**
- If some parameters of the model (masses and widths) are fixed. One can factor out integrals and cache them. **This allows for massive speed gains i.e. single fit now took 15 mins!**

Some benchmarks with other decays

Benchmark runs (fit time only), compare 2 machines.
CPU1: Intel Core i5-3570 (4 cores @ 3.4GHz, 16Gb RAM)
GPU1: NVidia GeForce 750Ti (640 CUDA cores @ 1020MHz, 2Gb VRAM, 88Gb/s, 40 Gflops DP)
CPU2: Intel Xeon E5-2620 (32 cores @ 2.1GHz, 64Gb RAM)
GPU2: NVidia Quadro p5000 (2560 cores @ 1600MHz, 16Gb VRAM, 320Gb/s BW, 280 Gflops DP)
GPU3: NVidia K20X (2688 cores @ 732MHz, 6Gb VRAM, 250Gb/s BW, 1300 Gflops DP)

	Iterations	Time, sec				
		CPU1	GPU1	CPU2	GPU2	GPU3
$D^0 \to K^0_S\pi^+\pi^-$, 100k events, 500×500 norm.						
Numerical grad.	2731	488	250	113	59	82
Analytic grad.	297	68	36	18	12	19
$D^0 \to K^0_S\pi^+\pi^-$, 1M events, 1000×1000 norm.						
Numerical grad.	2571	3393	1351	937	306	378
Analytic grad.	1149	1587	633	440	148	180
$\Lambda^0_b \to D^0 p\pi^-$, 10k events, 400×400 norm.						
Numerical grad.	9283	434	280	162	157	278
Analytic grad.	425	33	23	18	21	32
$\Lambda^0_b \to D^0 p\pi^-$, 100k events, 800×800 norm.						
Numerical grad.	6179	910	632	435	266	364
Analytic grad.	390	133	62	126	32	45

$D^0 \to K^0_S\pi^+\pi^-$ amplitude: isobar model, 18 resonances, 36 free parameters
$\Lambda^0_b \to D^0 p\pi^-$ amplitude: 3 resonances, 4 nonres amplitudes, 28 free parameters

Slide borrowed from Anton Poluektov [talk]

Summary

- TFA package has many advantageous:
  - **Large flexibility** with quick model development.
  - Inherent optimisations and analytic gradients allow for **gain in evaluation speed**.
  - Easily **scalable to multidimensions**.
  - Easy to run on **various computing architechures without expert knowledge**.
  - **Wider support** from TF community.
  - Added value in training of young researchers seeking job outside of academia.
- Disadvantageous include:
  - Graph building impacts performance for large number of quick and simple fits.
  - Large datasets use large memory (> few Gb of RAM).
  - Less efficient than code developed with CUDA, etc.
- For future, allow for switching of computing backends to numpy, numba, JAX (like in TensorWaves package, even integrate with it).
- **Note TFA (based on TF v1) has been upgraded and split into two: AmpliTF and TFA2 [Demo scripts][Installation instructions and guide].**