zfit scalable pythonic fitting

Jonas Eschle (Jonas.Eschle@cern.ch)
A. Puig, R. S. Coutinho, N. Serra
University of Zurich^UZH

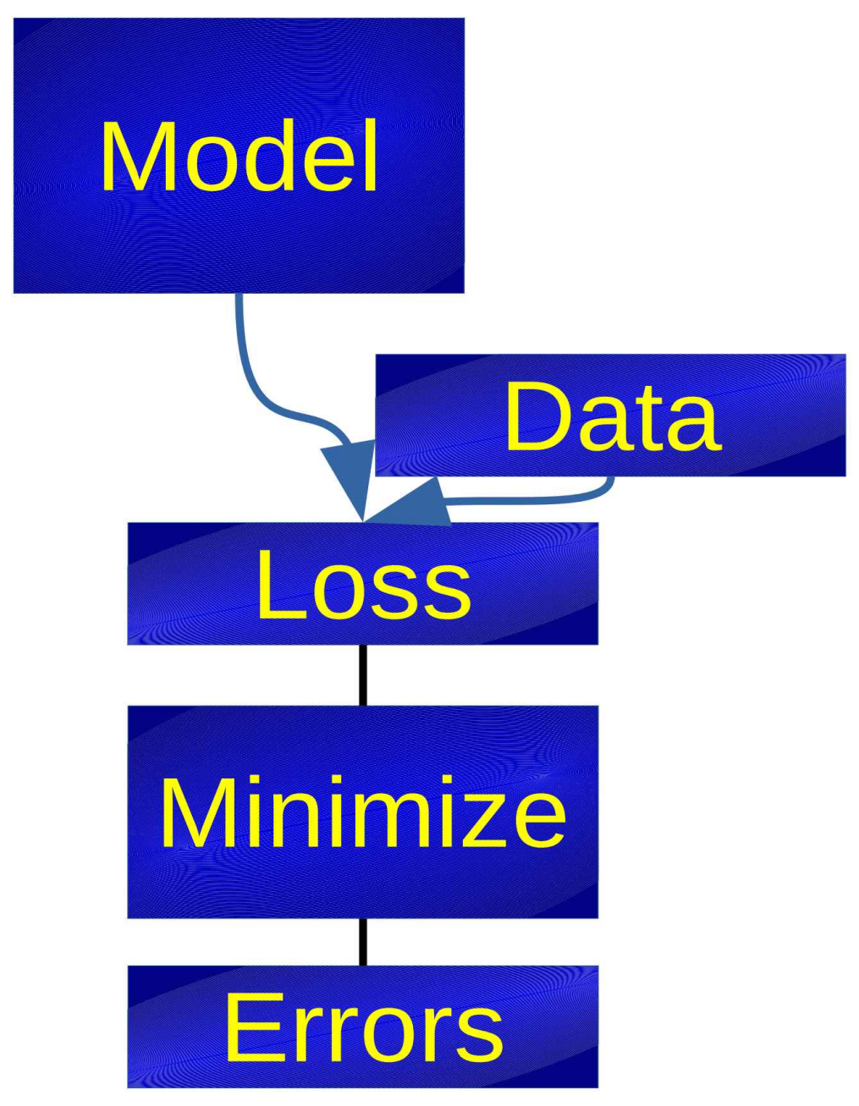
Introduction

zfit is a model fitting library in pure Python, well integrated into the scientific ecosystem.
It offers an alternative to ROOT/RooFit.

- Load ROOT files, use Minuit
- pip/conda install zfit (no ROOT needed)
- Highly performant (TensorFlow as backend)
- Coordinated effort for model fitting in Python for HEP

Scikit HEP affiliated

Workflow



```
obs = zfit.Space("x", limits=(-2, 3))

mu = zfit.Parameter("mu", 1.2, -4, 6)
sigma = zfit.Parameter("sigma", 1.3, 0.1, 10)
gauss = zfit.pdf.Gauss(mu=mu, sigma=sigma, obs=obs)

data = zfit.Data.from_numpy(obs=obs, array=normal_np)

nll = zfit.loss.UnbinnedNLL(model=gauss, data=data)

minimizer = zfit.minimize.Minuit()
result = minimizer.minimize(nll)

param_errors = result.error()
```

Model building

Custom model

```
class CustomPDF(zfit.pdf.ZPDF):
    _PARAMS = ['alpha']

    def _unnormalized_pdf(self, x):
        data = x.unstack_x()
        alpha = self.params['alpha']
        return tf.exp(alpha * data)

obs = zfit.Space("y", (-4, 4))
custom_pdf = CustomPDF(obs=obs, alpha=0.2)

integral = custom_pdf.integrate(limits=(-1, 2))
sample   = custom_pdf.sample(n=1000)
prob     = custom_pdf.pdf(sample)
```

Composition

```
frac = zfit.Parameter('fraction', 0.5, 0, 1)
sum_pdf = zfit.pdf.SumPDF([gauss, exponential], frac)
```

Product 2-D
(dims defined by observables)

```
#['y', 'x'] <- obs 'y'   *    'x'
product_2d =  custom_pdf  * sum_pdf
```

Parameter

Build arbitrary compositions

```
param1 = zfit.Parameter("param1", 1, 0, 5)
param2 = zfit.Parameter("param2", 2, 1, 5)

tensor1 = 5 + param1 * 42 ** tf.sqrt(param2)
param_comp = zfit.ComposedParameter('comp', tensor1)
```

Data

- Multiple formats supported
- Full capability of Pandas DataFrames

```
data_raw = zfit.Data.from_root(...)
df = data_raw.to_pandas()
# preprocess in pandas
data = zfit.Data.from_pandas(df)
```

Loss

Simultaneous

```
nll1 = zfit.loss.UnbinnedNLL(model=gauss1, data=data1)
nll2 = zfit.loss.UnbinnedNLL(model=gauss2, data=data2)
nll_simultaneous = nll1 + nll2
```

Constraints

```
constr = zfit.constraint.GaussianConstraint(...)
mu_penalty = tf.square(mu - 1.3)
nll.add_constraints([constr, mu_penalty])
```

Minimization

- Wraps minimizer libraries
- Minuit, Scipy, ...

```
minimizer = zfit.minimize.Adam(...)
result = minimizer.minimize(loss)
```

- Convenient BaseClass available

Fit result

- Access results

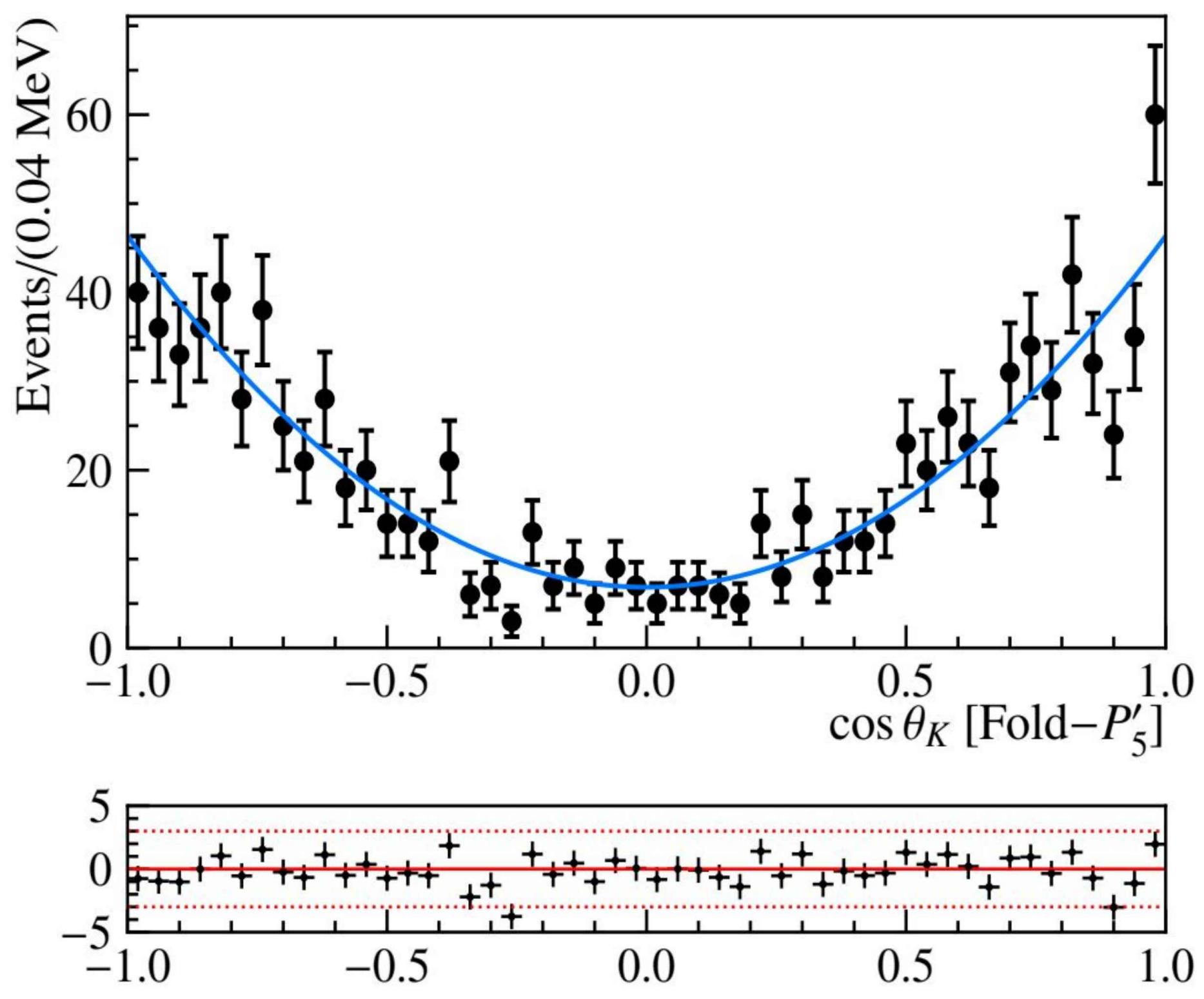
```
successful = result.converged
mu_result = result.params[mu]
```

- Calculate uncertainties

```
hesse_error = result.hesse()
minos_error = result.error()
```

Toy study angular fit 3-D

- Custom PDF (implemented from [1])
- Plot: projection of 1 observable

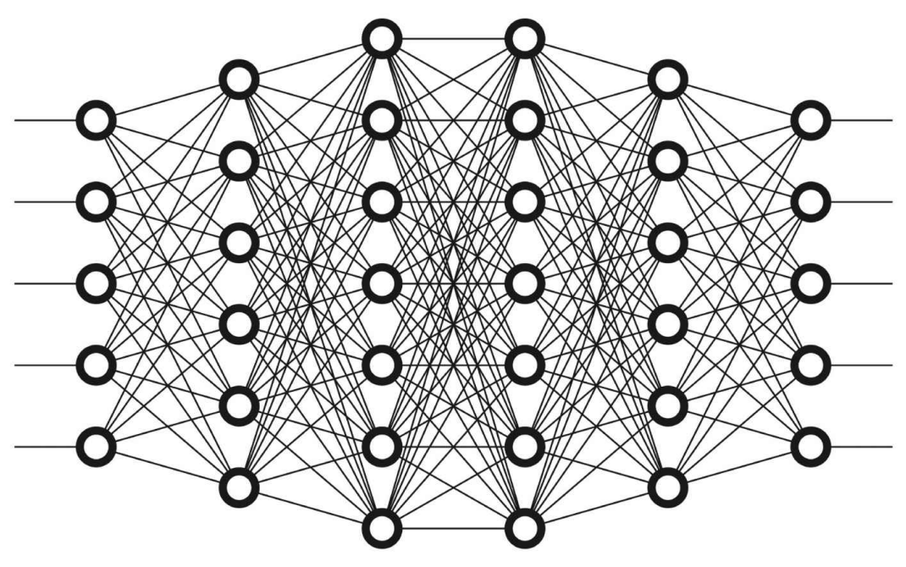


Conclusion

zfit provides the possibility of model fitting in pure Python for HEP analyses.
With its well defined API, workflow and modularity, it is simply extendable and allows to build libraries on top, e.g. for advanced statistical analysis.

Earning the fruits of Machine Learning

The success of Deep Learning in recent years led to the appearance of several libraries such as TensorFlow.



TensorFlow

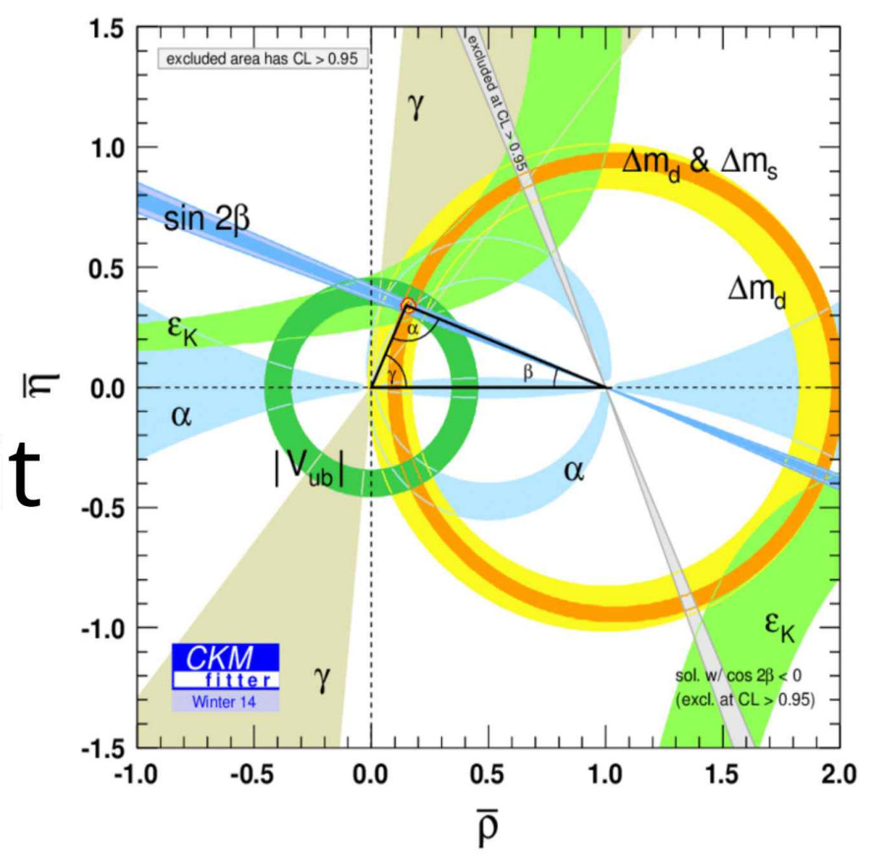
They offer Numpy style syntax while delivering state-of-the-art performance.

zfit uses this as the computing backend.

Projects

CKM fitter

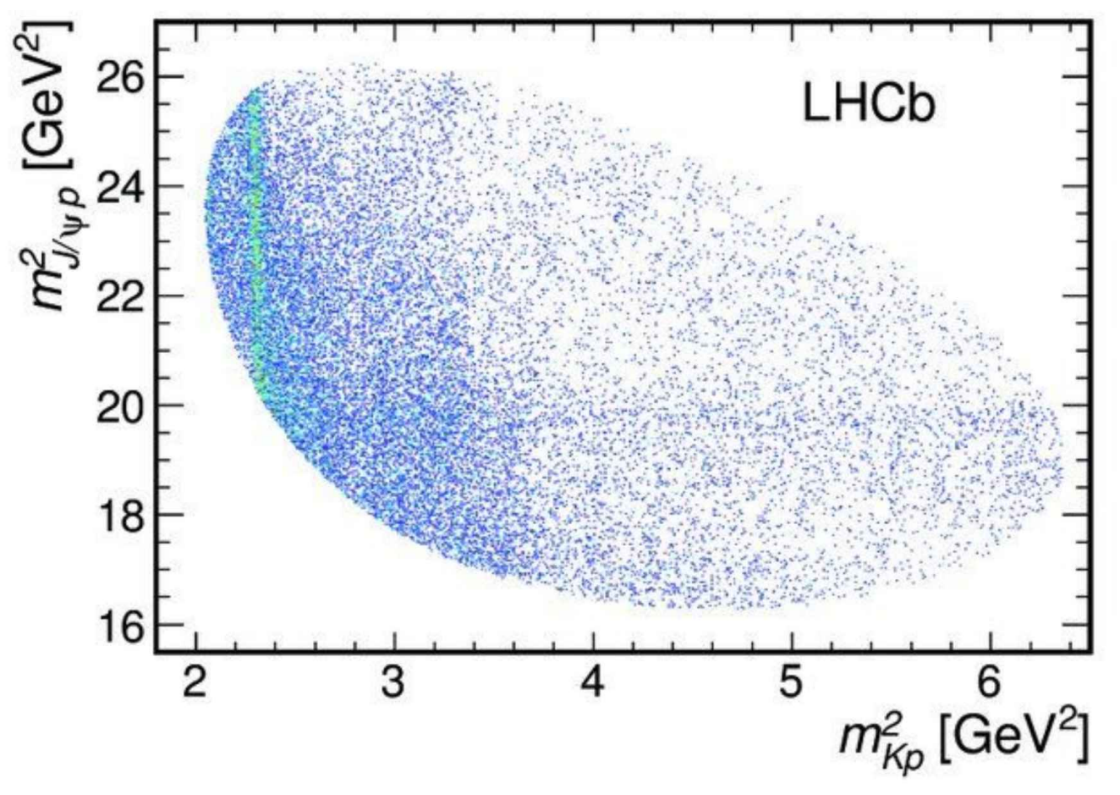
Determining the values of the CKM matrix from a global fit



Dalitz analysis

Amplitude analysis of three body decays:
spectroscopy

CP violation



Many more...

Do you like Python? Machine Learning? Statistics? Want to learn how to work on a large software project? Just contact us!

[1] The LHCb collaboration, Aaij, R., Abellán Beteta, C. et al. J. High Energ. Phys. (2016) 2016: 104. https://doi.org/10.1007/JHEP02(2016)104

https://www.researchgate.net/publication/319122026/figure/fig20/AS:631639814193241@1527606075987/Dalitz-plot-distribution-for-L-0-b-J-pspK-decays-as-observed-by-LHCb-47_W640.jpg