



Colibri

A NEW TOOL FOR FAST-FLYING PDF FITS

[10.1140/epjc/s10052-025-15185-8](https://doi.org/10.1140/epjc/s10052-025-15185-8)

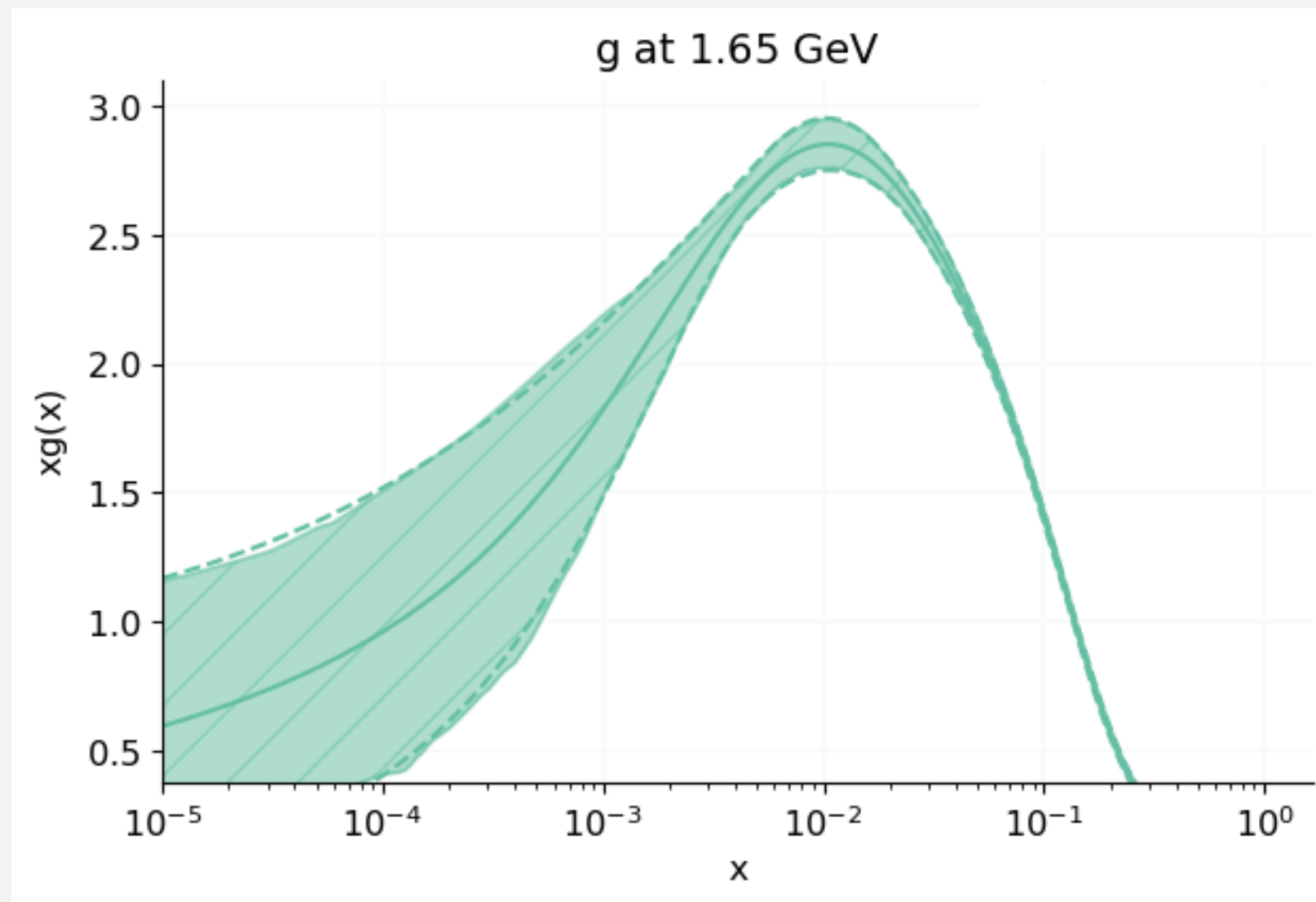


Valentina Schütze Sánchez
in collaboration with:
Mark N. Costantini
Luca Mantani
James Moore
Maria Ubiali

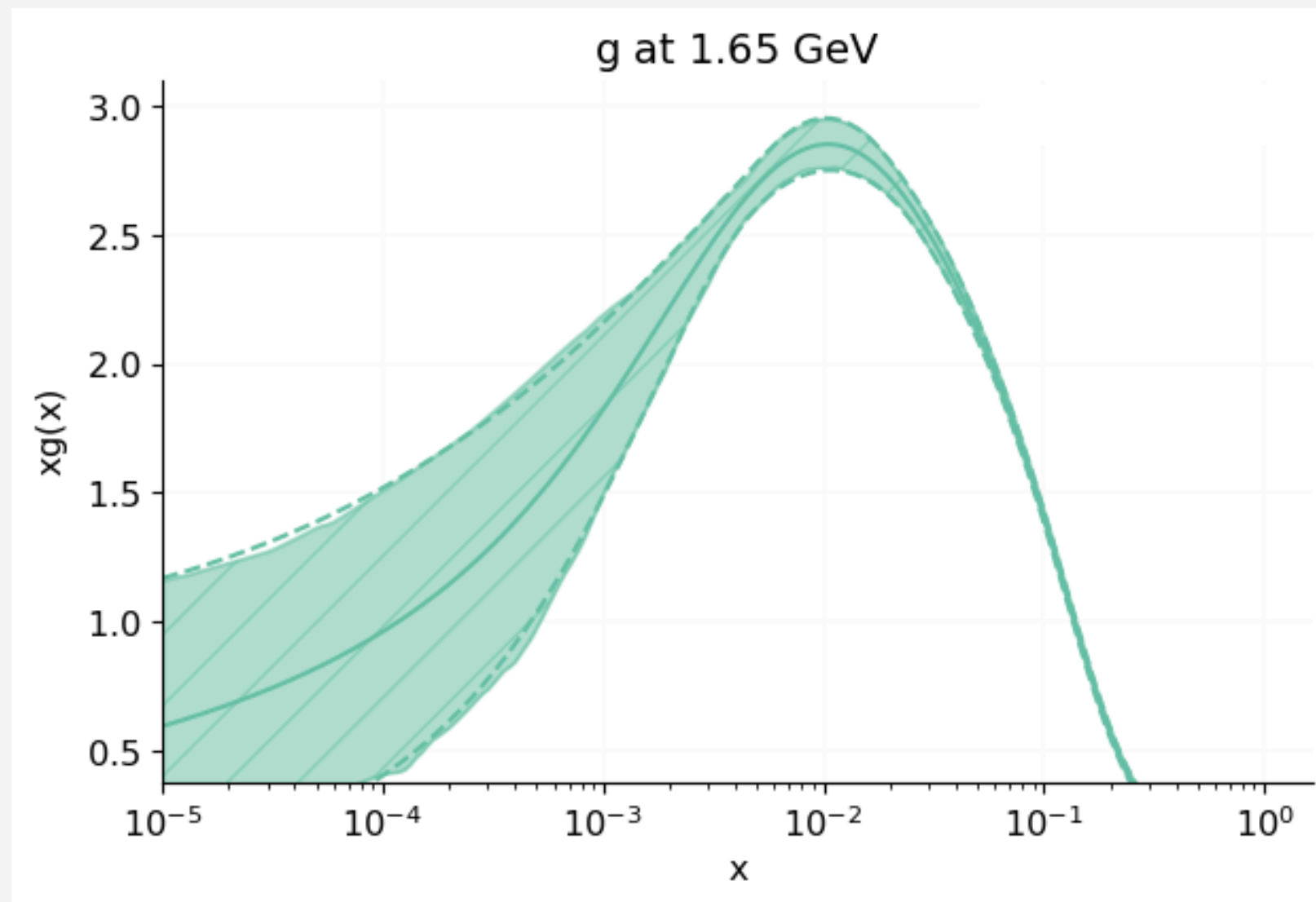


Modelling Parton Distribution Functions

Modelling Parton Distribution Functions

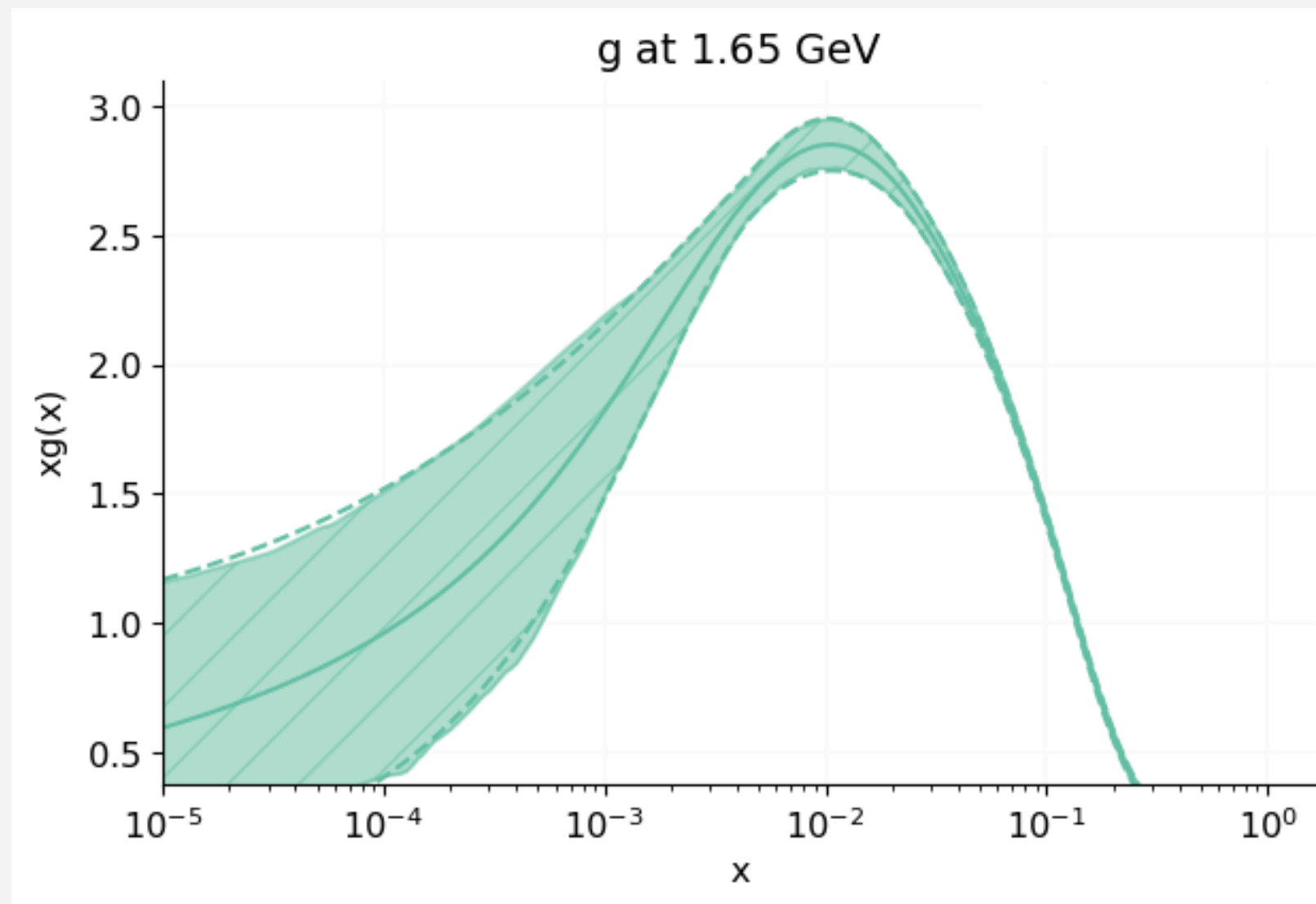


Modelling Parton Distribution Functions



$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$

Modelling Parton Distribution Functions

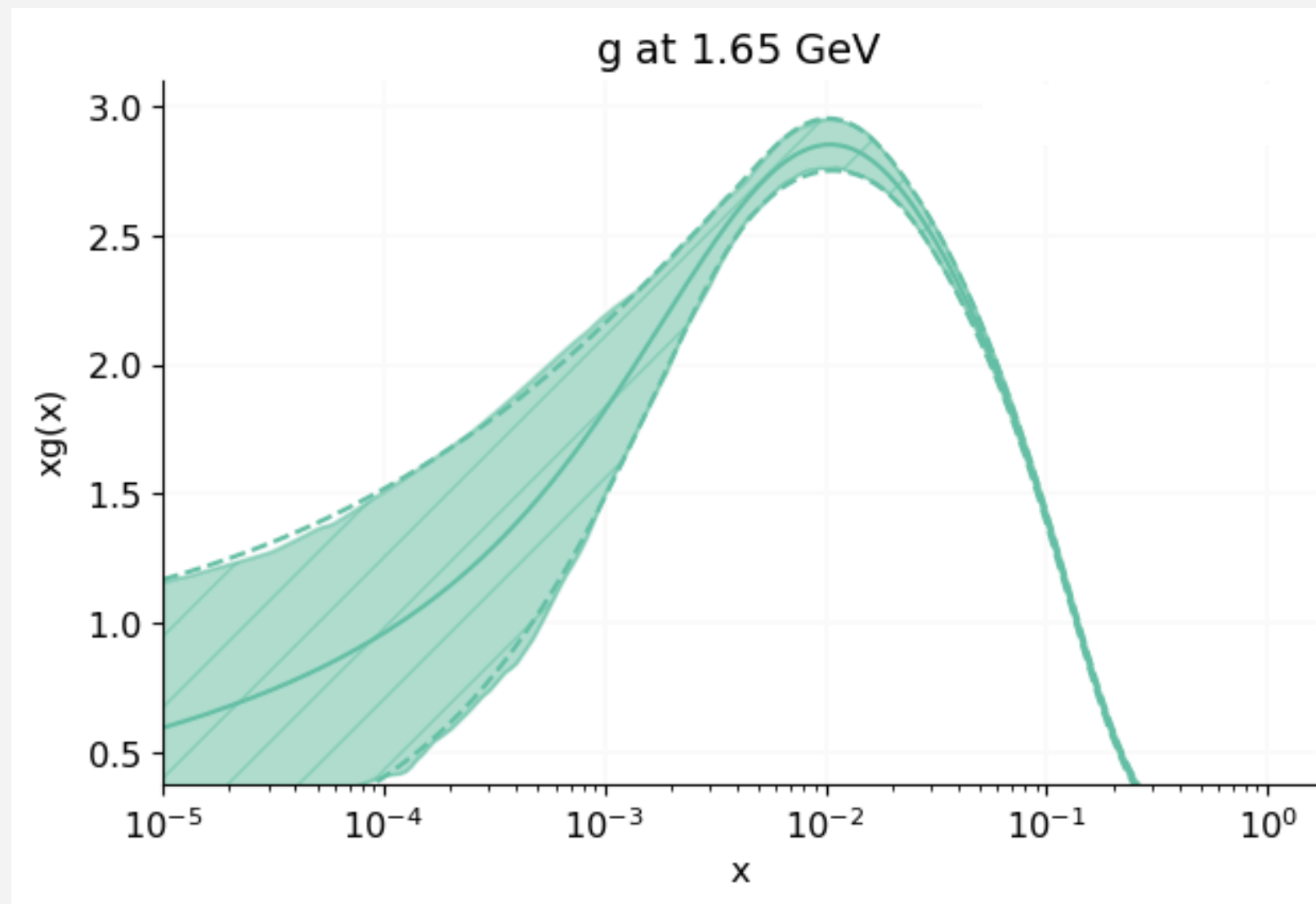


$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$

Polynomial parametrisations

$$f(x, Q_0) = Ax^{-\alpha}(1-x)^{\beta} (1 + \varepsilon\sqrt{x} + \gamma x + \dots)$$

Modelling Parton Distribution Functions



$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$

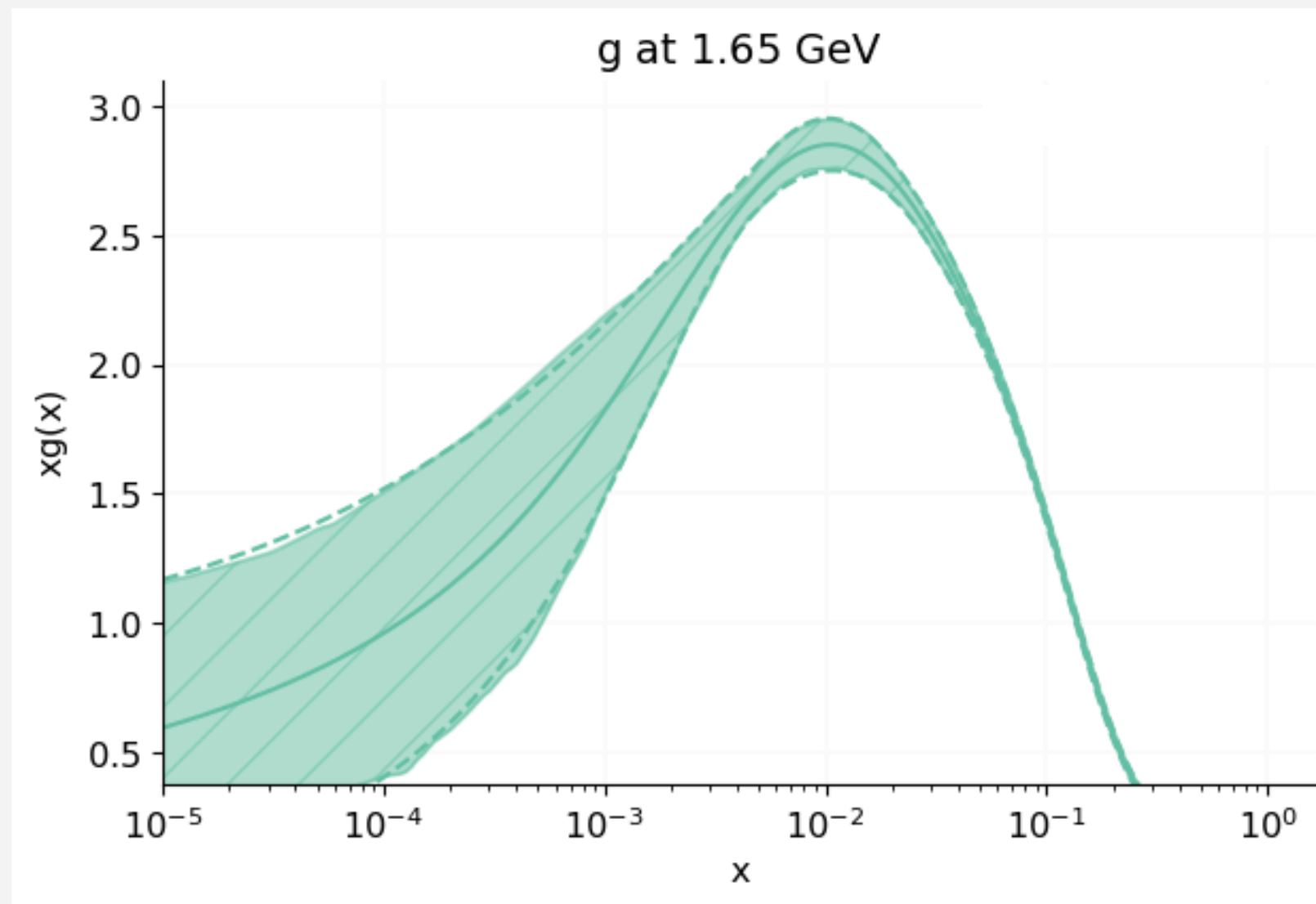
Polynomial parametrisations

$$f(x, Q_0) = Ax^{-\alpha}(1-x)^{\beta} (1 + \varepsilon\sqrt{x} + \gamma x + \dots)$$

Neural Network

$$f(x, Q_0) = x^{-\alpha}(1-x)^{\beta} NN(x)$$

Modelling Parton Distribution Functions



$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$

Polynomial parametrisations

$$f(x, Q_0) = Ax^{-\alpha}(1-x)^{\beta} (1 + \varepsilon\sqrt{x} + \gamma x + \dots)$$

Neural Network

$$f(x, Q_0) = x^{-\alpha}(1-x)^{\beta} NN(x)$$

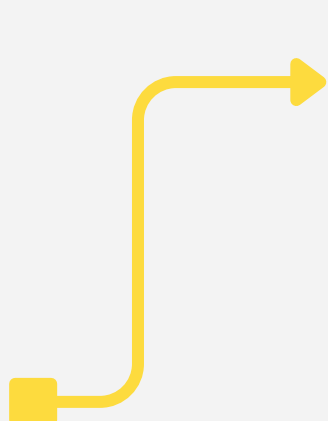
... and more!

Uncertainty treatment

$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$

Uncertainty treatment

$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$



**Monte Carlo
replica method**

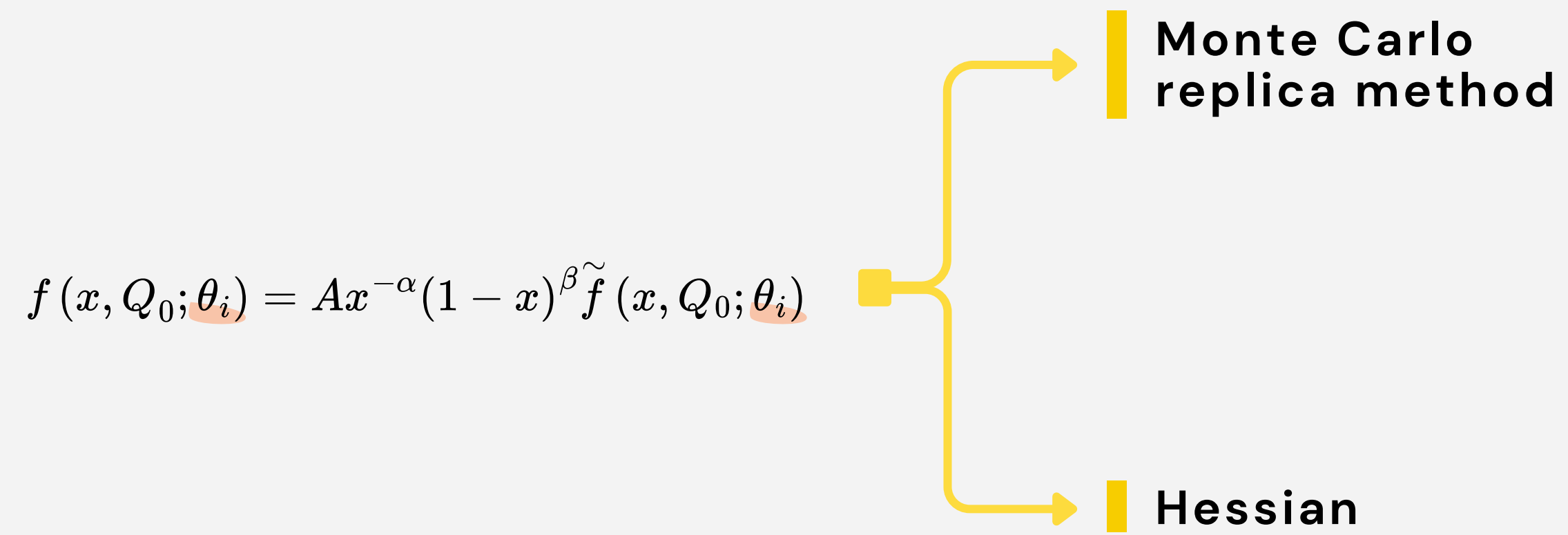
Uncertainty treatment

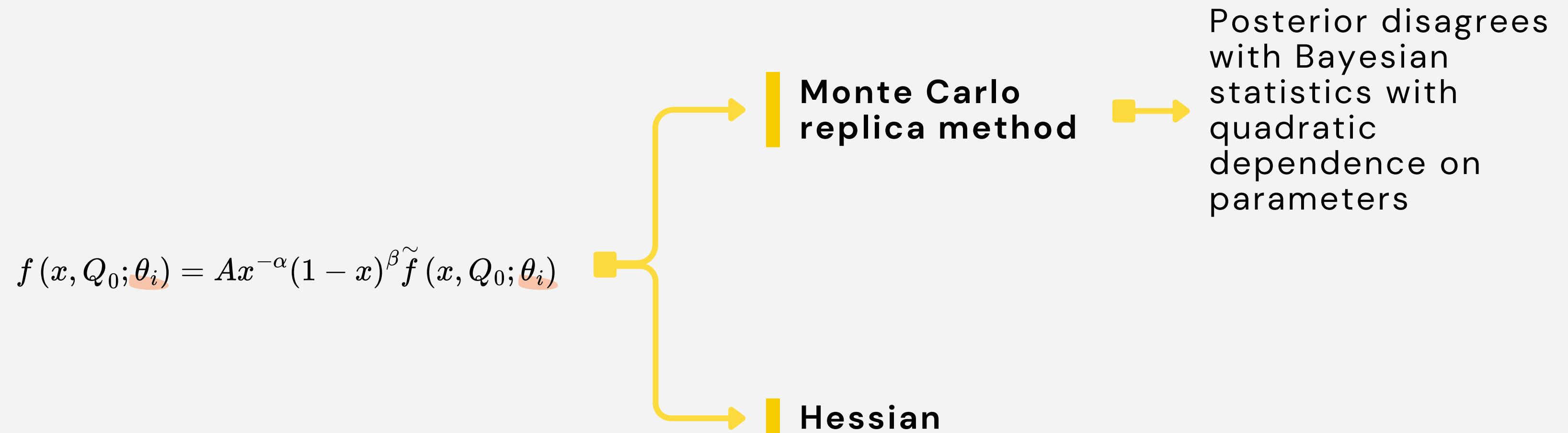
$$f(x, Q_0; \theta_i) = Ax^{-\alpha}(1-x)^{\beta} \tilde{f}(x, Q_0; \theta_i)$$

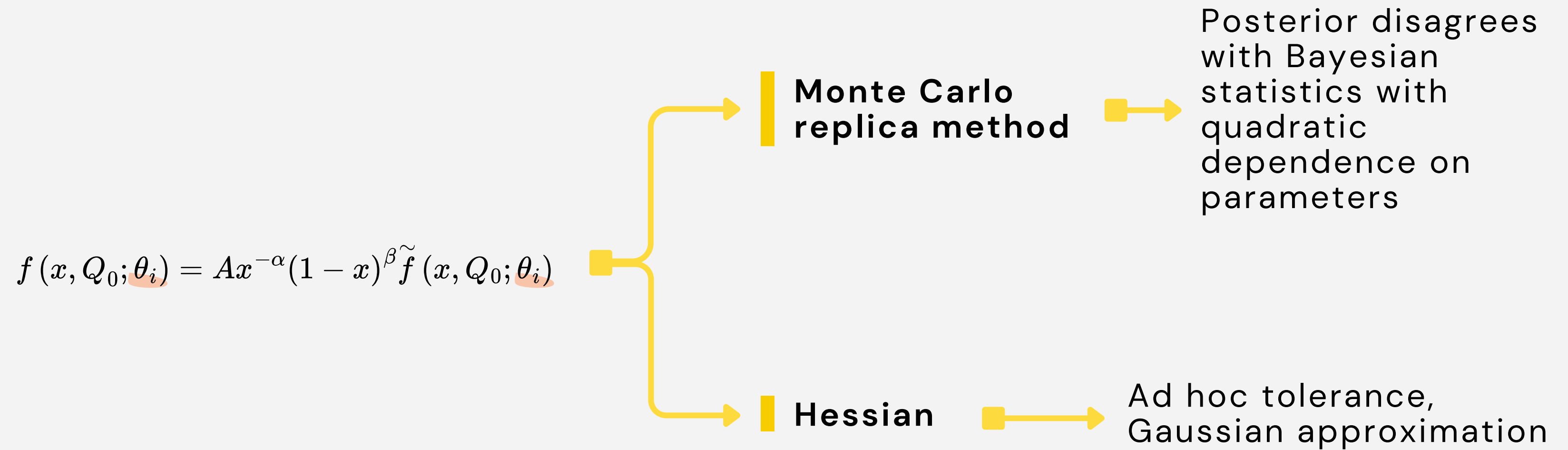
Monte Carlo
replica method

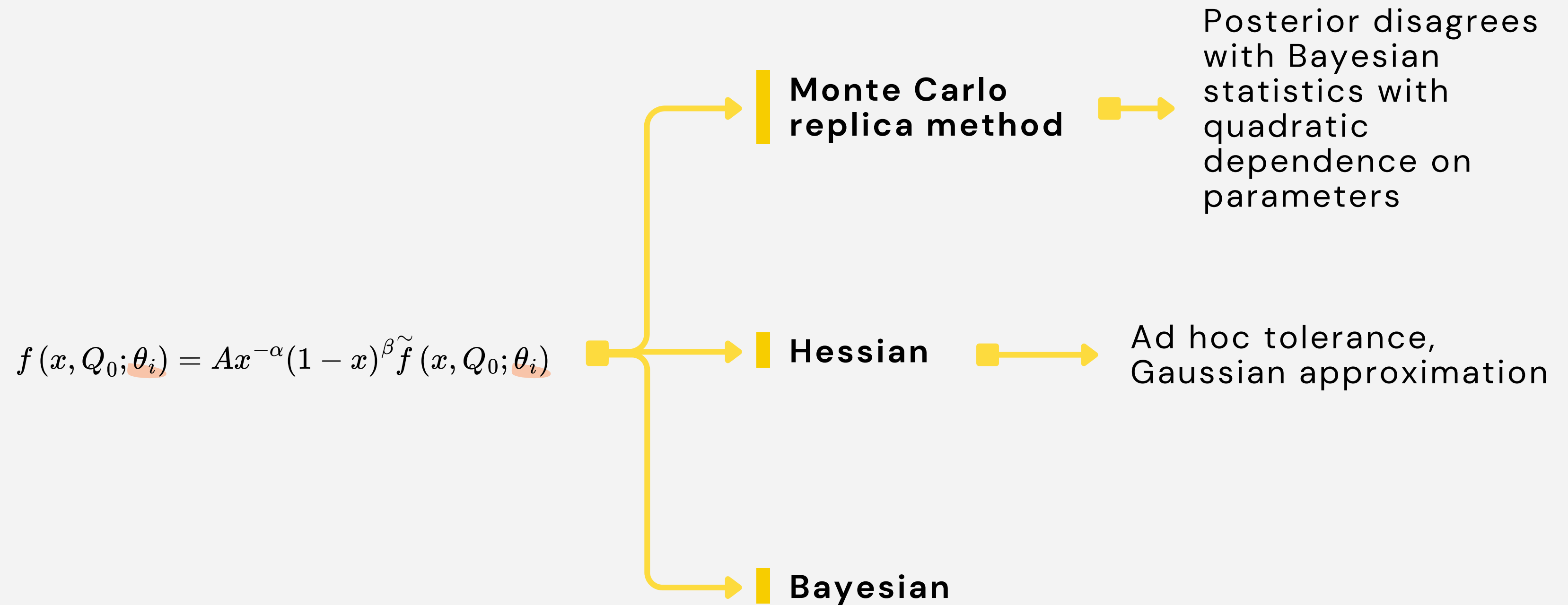
Hessian

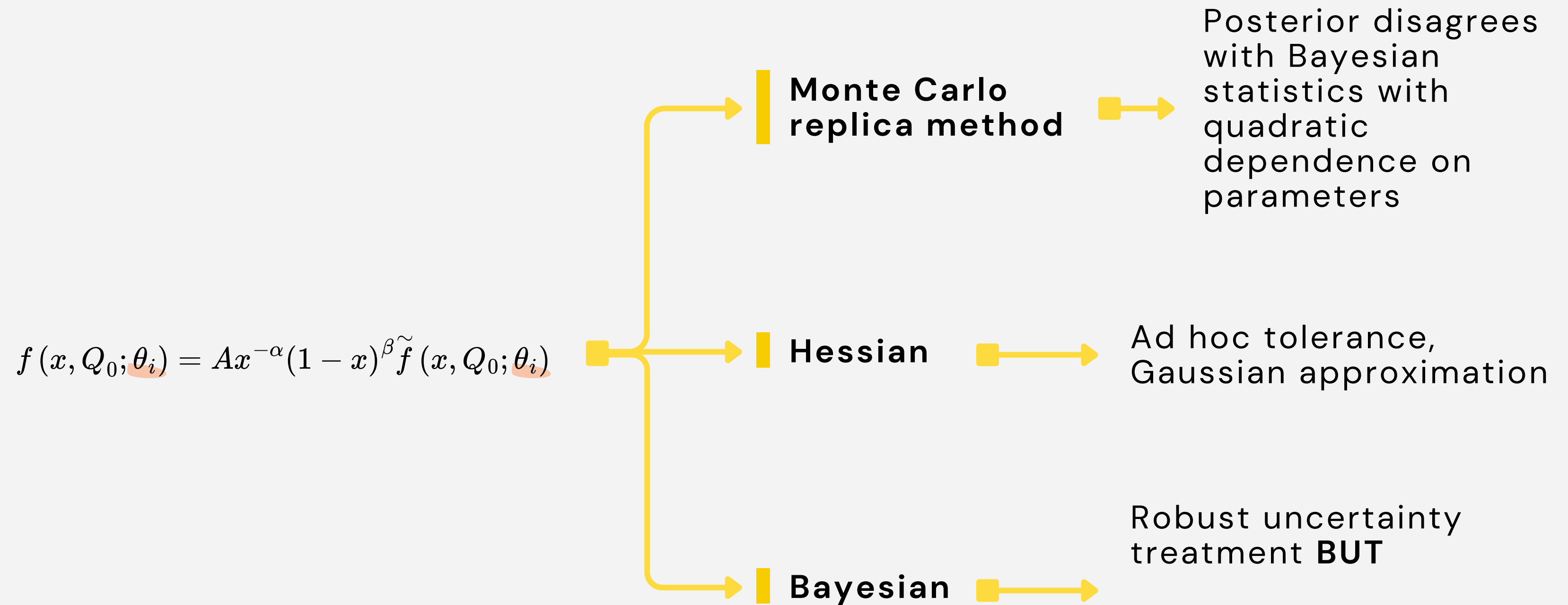
How can we systematically compare
different PDF models and benchmark
inference methods within the same
code framework?

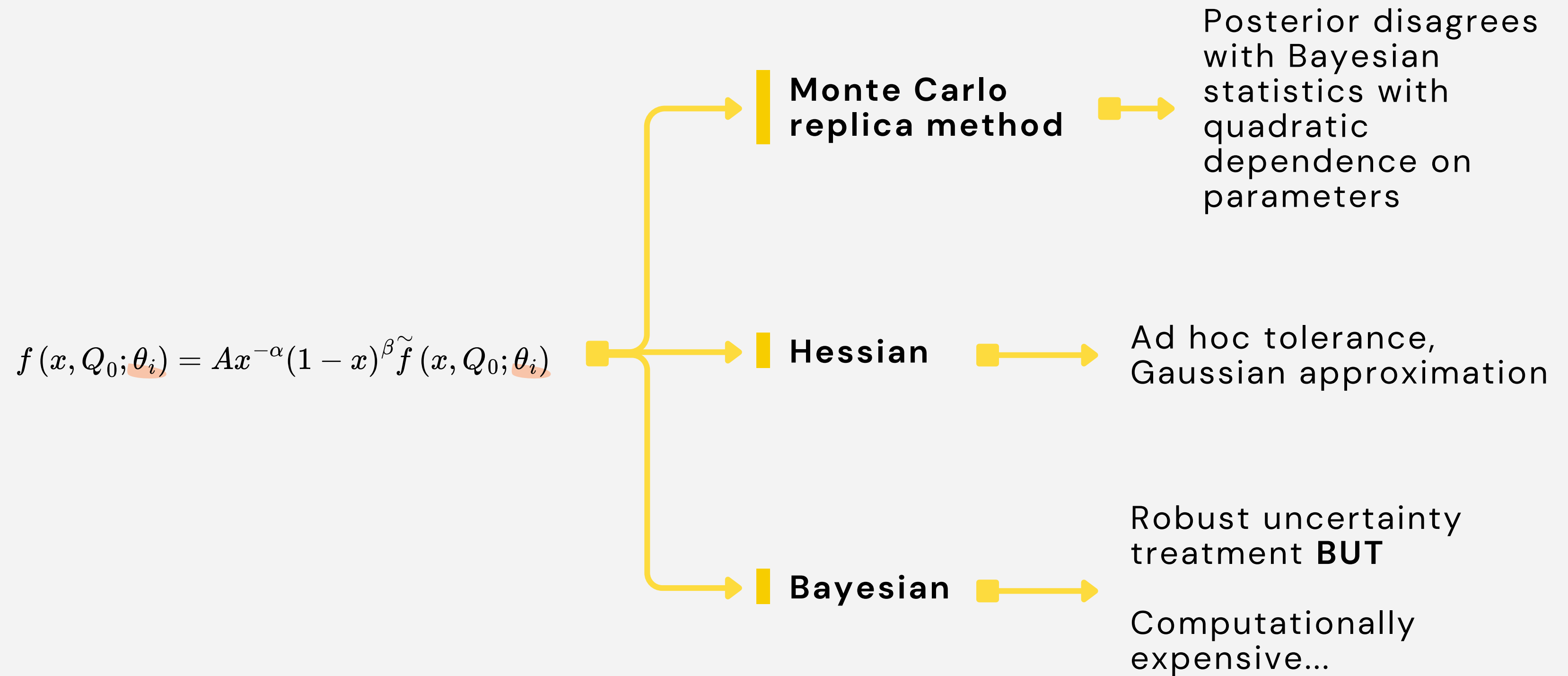












Motivation

Develop a Bayesian PDF fitting tool
that allows for benchmarking fitting
methodologies and systematic model
comparison

Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.



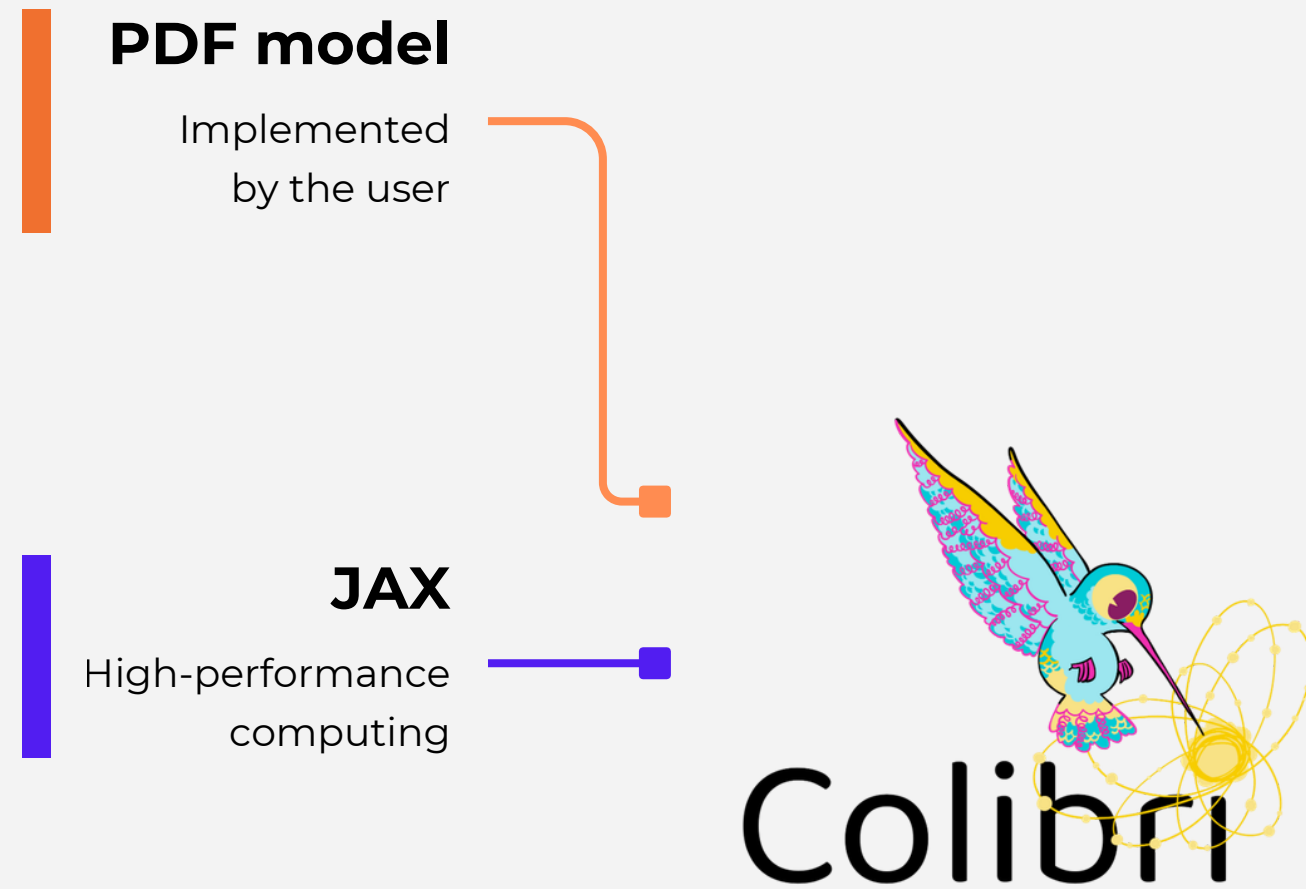
Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.

PDF model

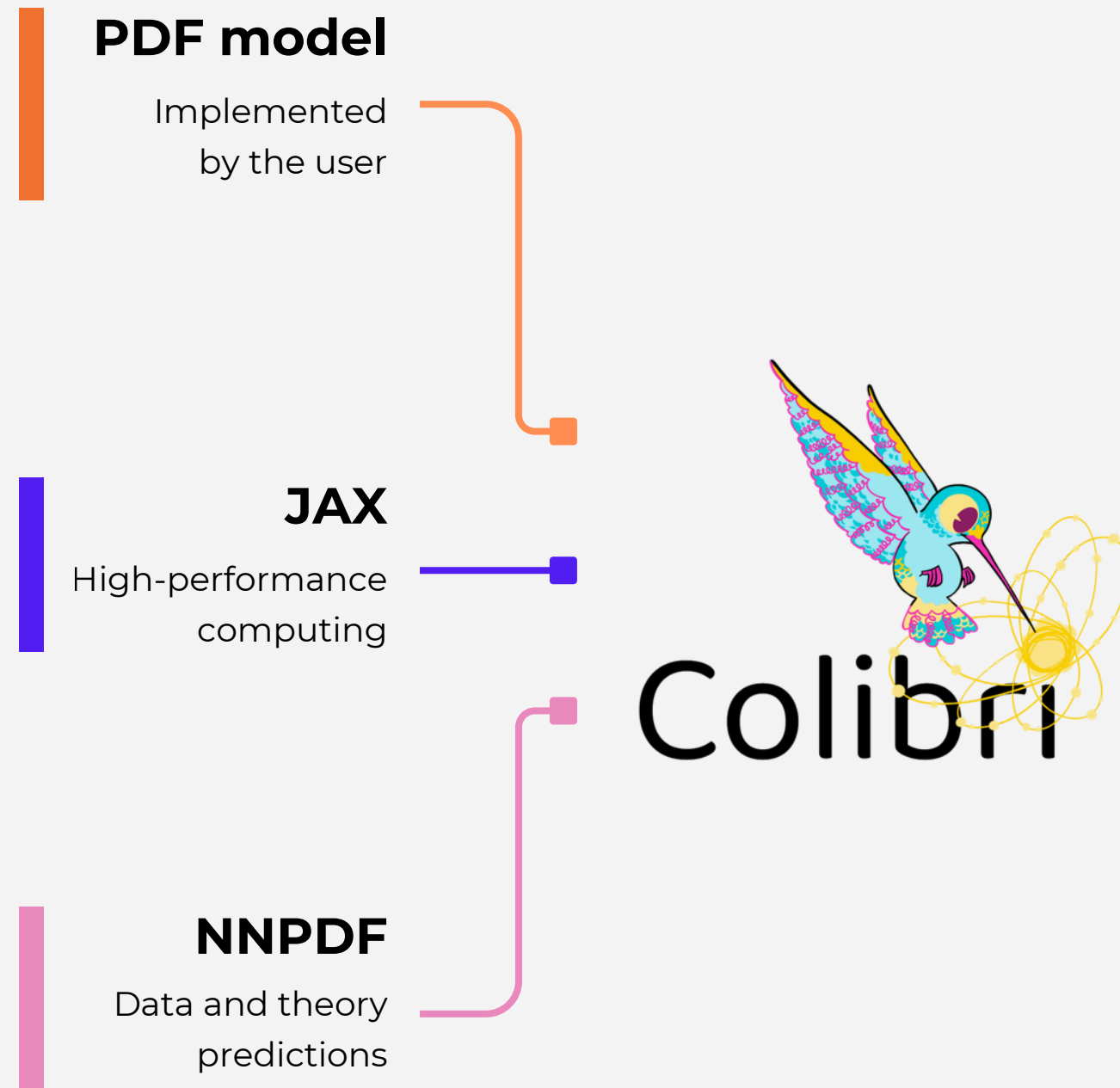
Implemented
by the user



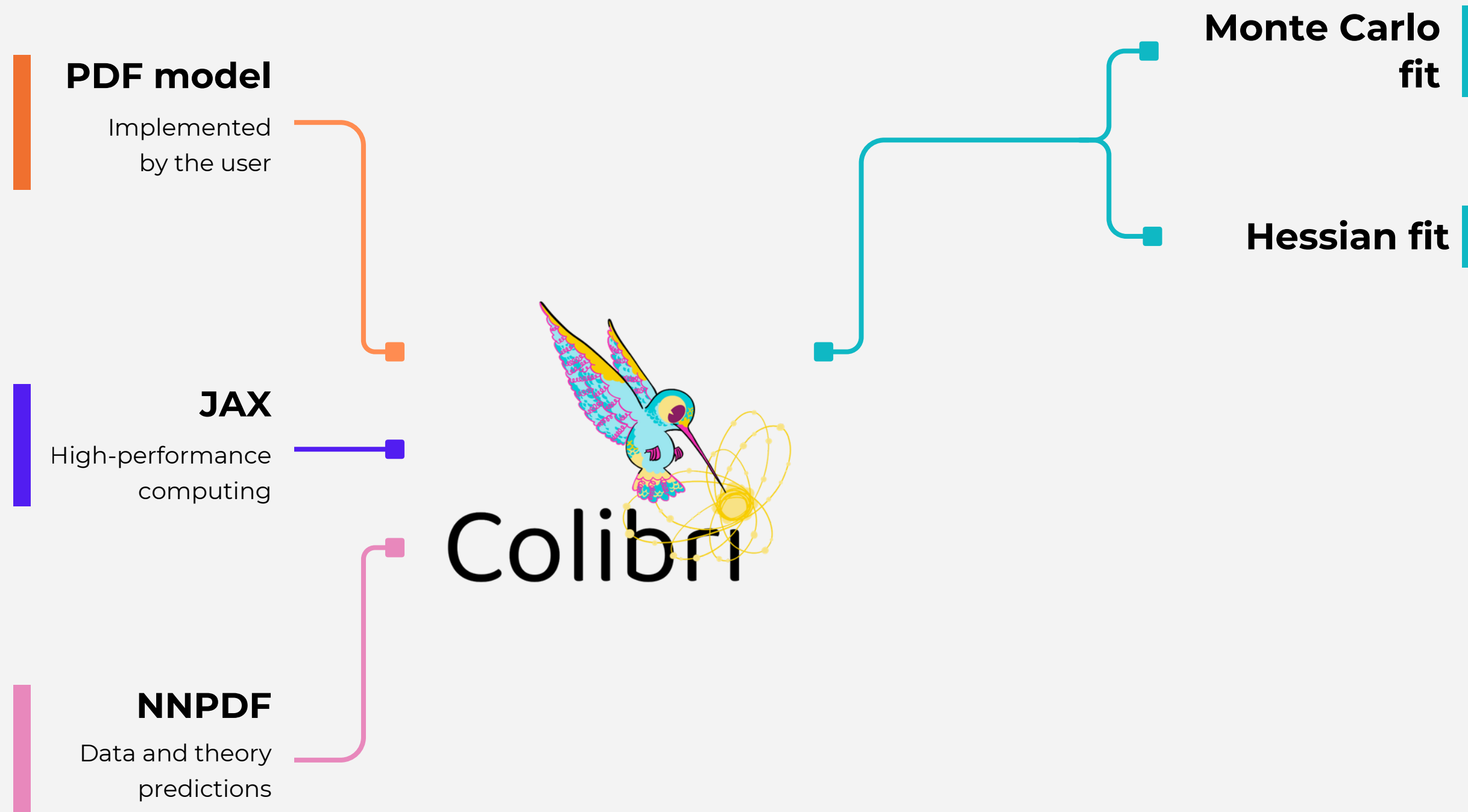
Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.



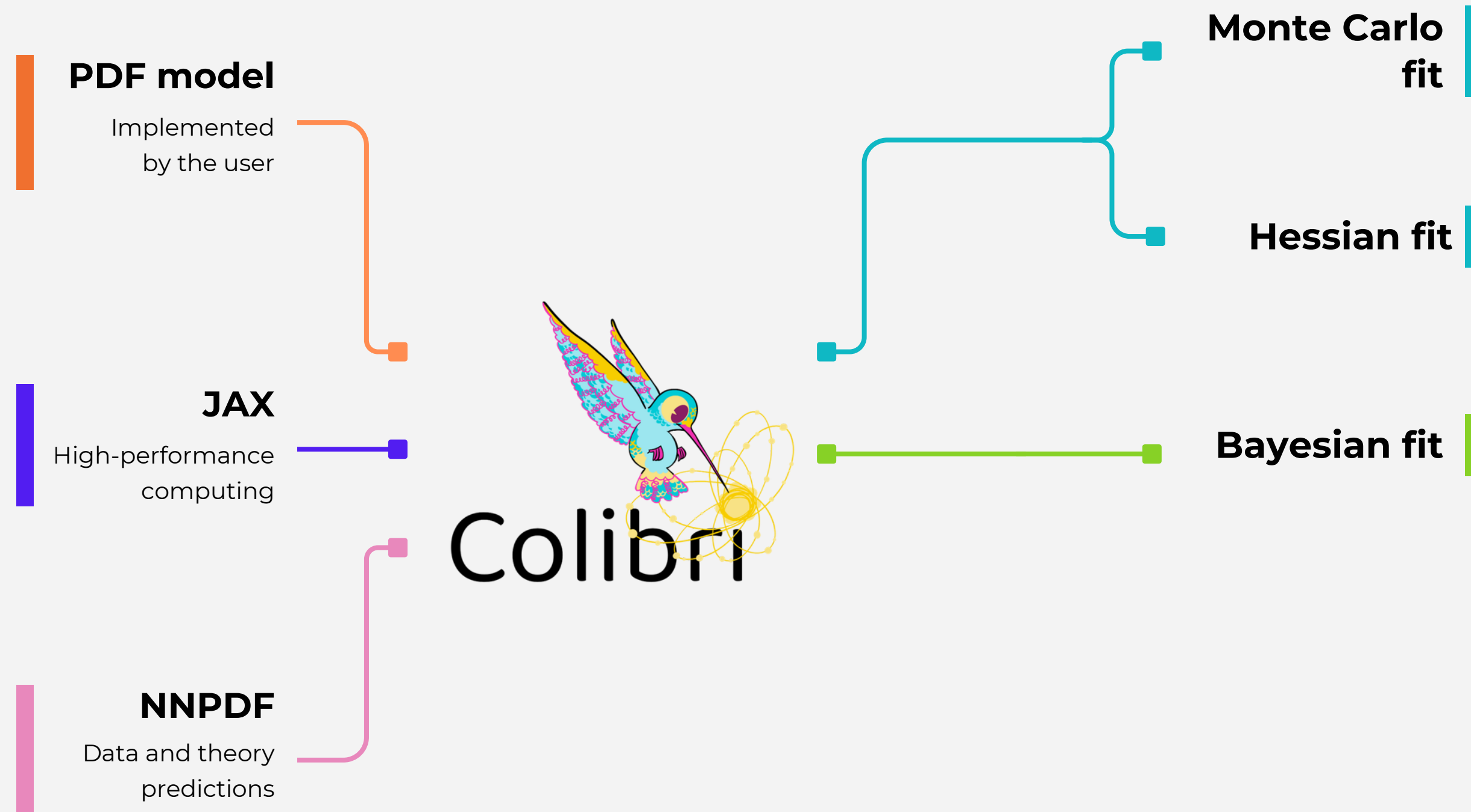
Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.



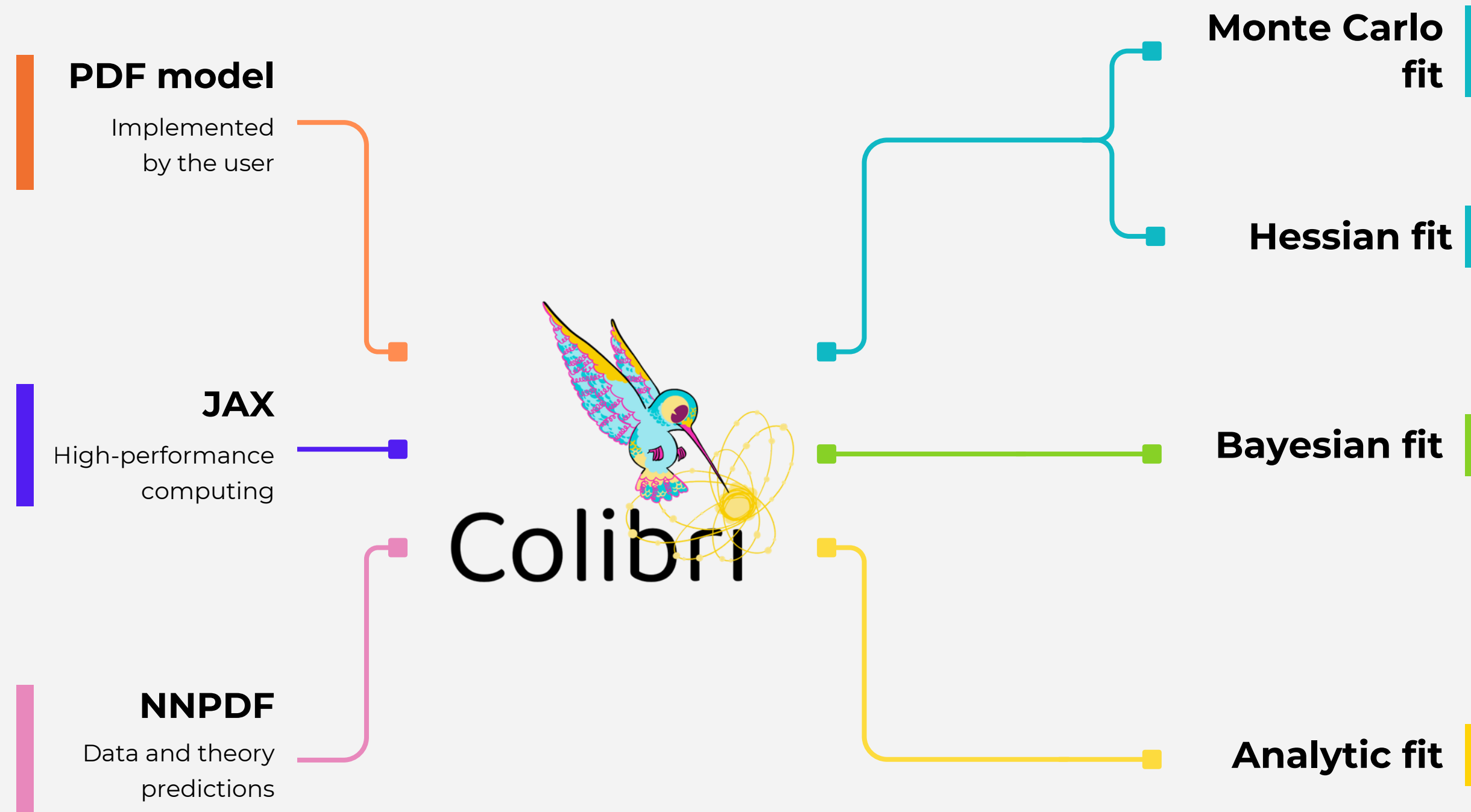
Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.



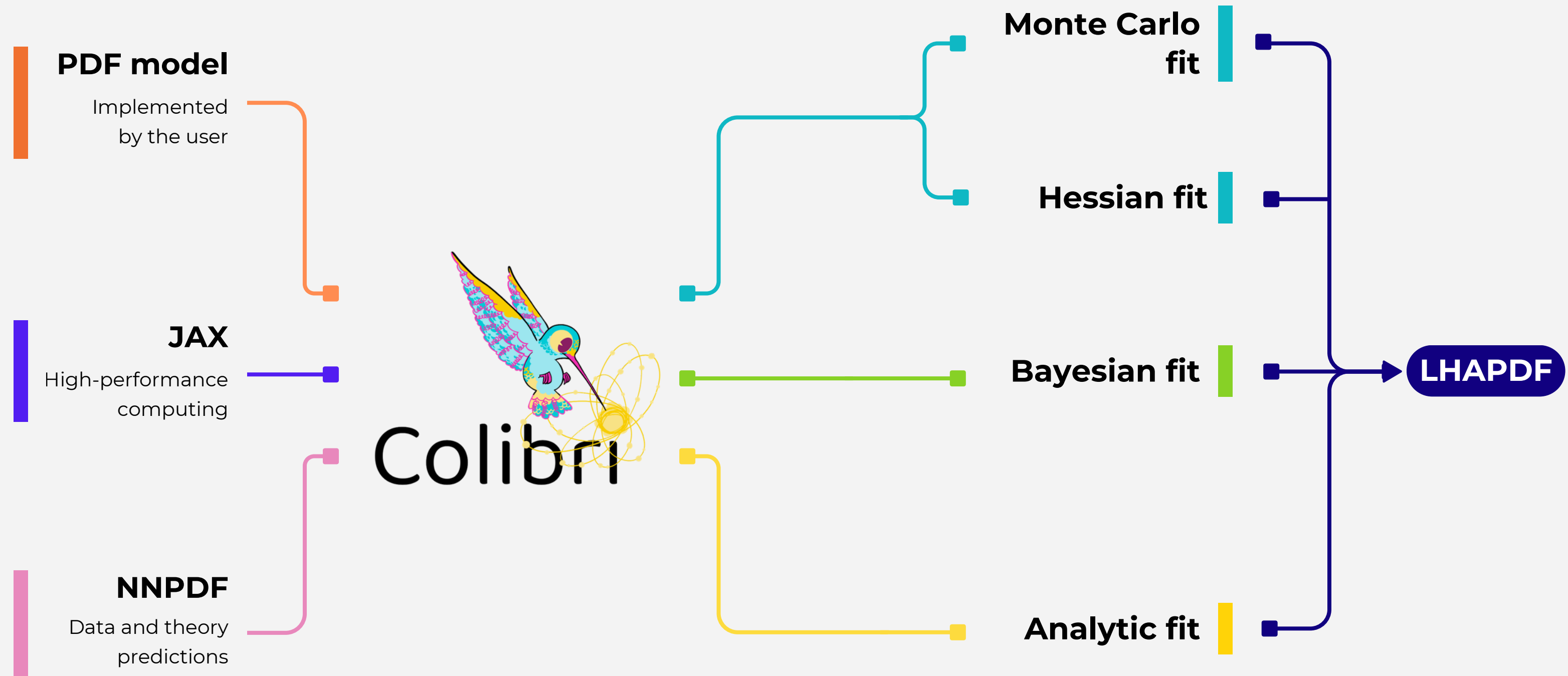
Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.

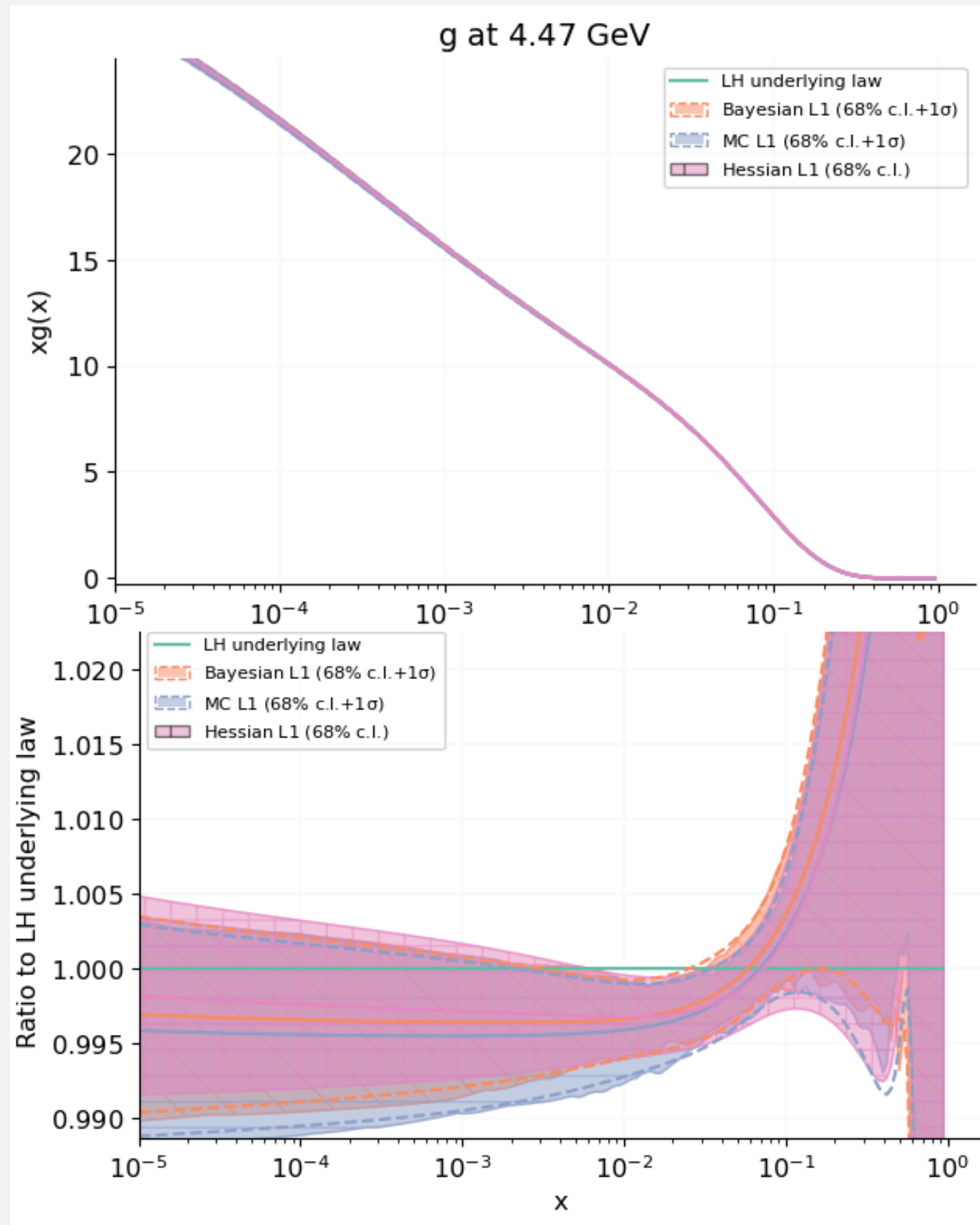


Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.



Colibri is an open-source PDF fitting tool where the user controls **what model** to fit and with **which inference method**.



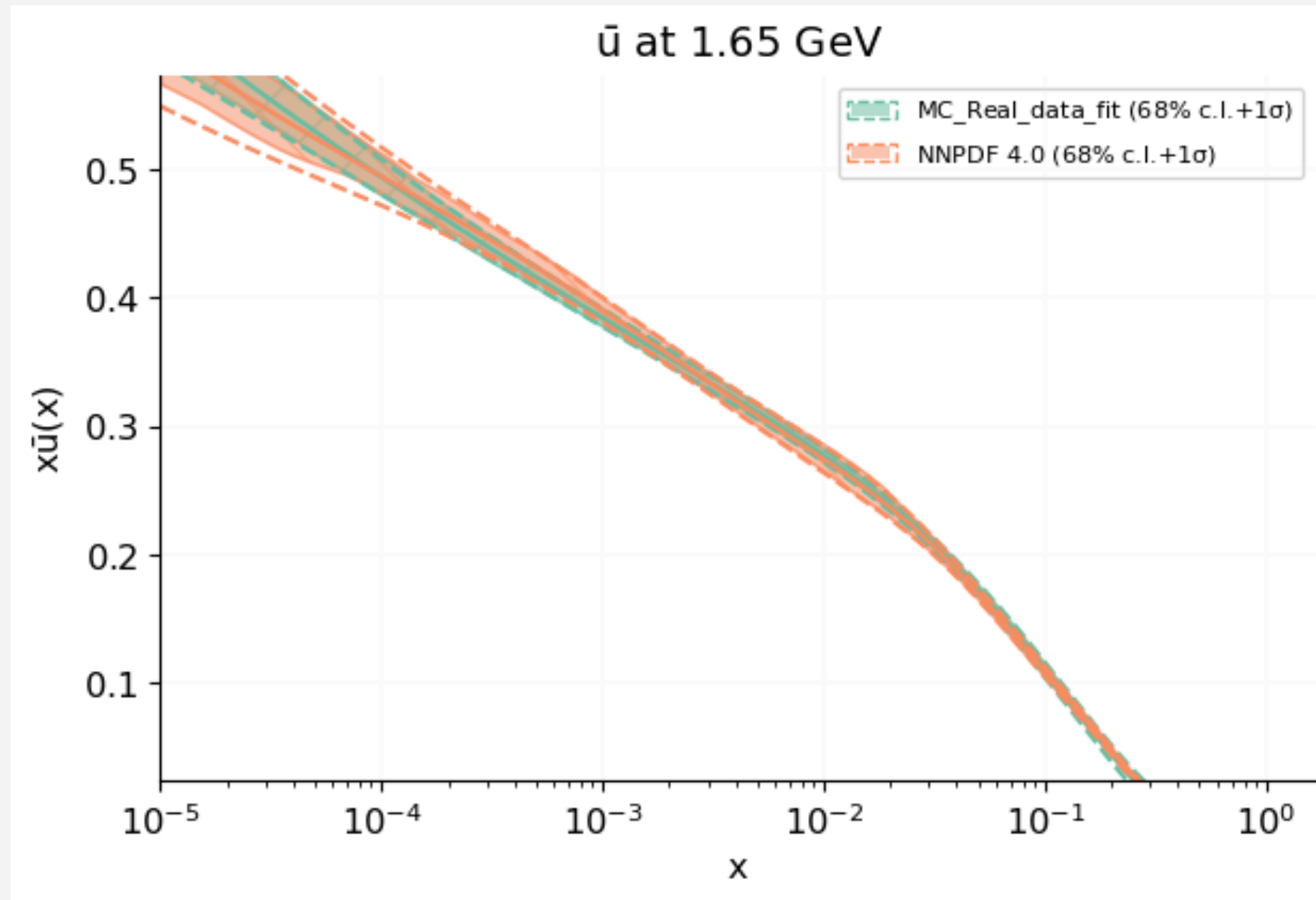


Example:

Les Houches parametrisation

- Polynomial parametrisation
- 13 free parameters

n3fit

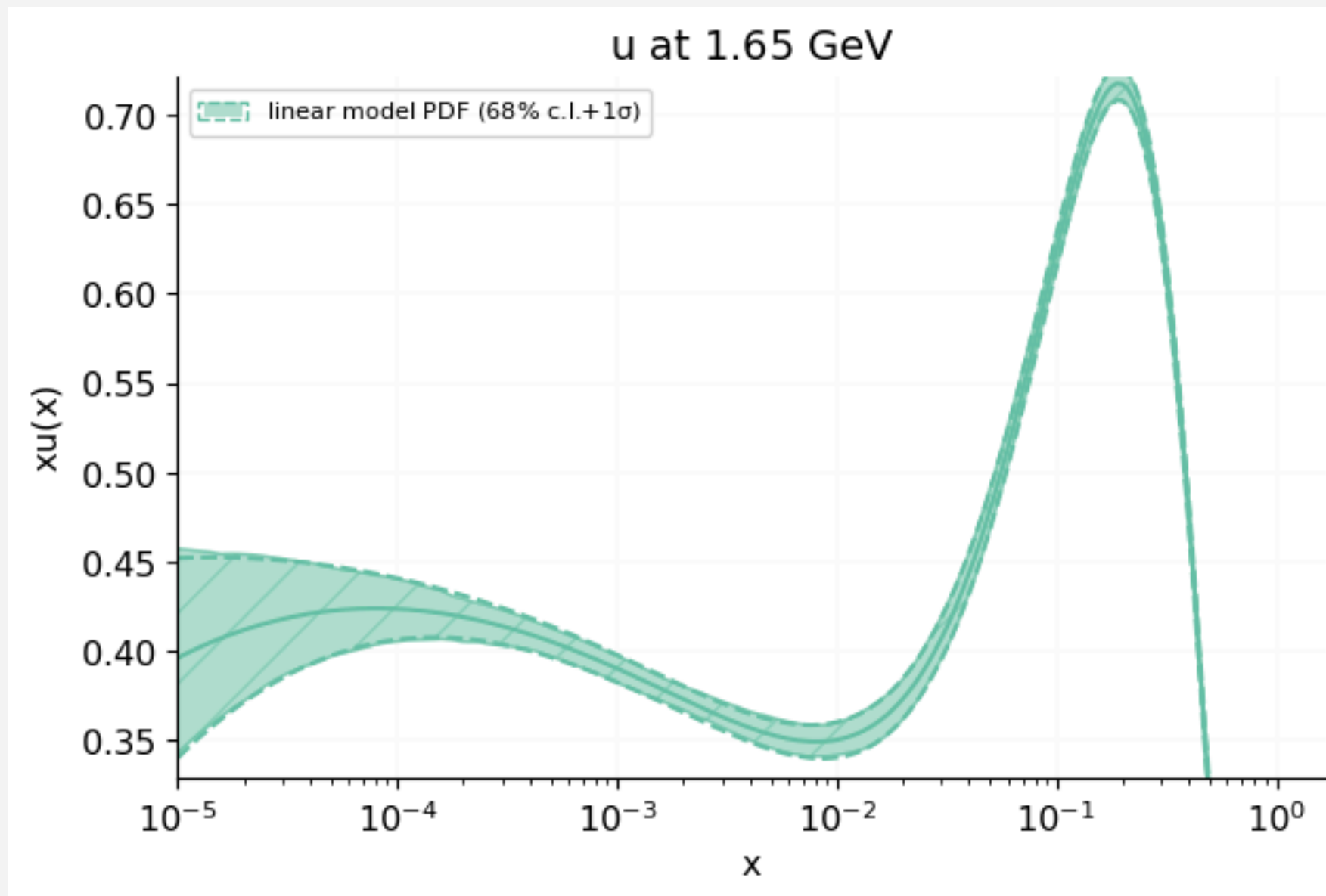


(credits: Ella Cole)

[10.1140/epjc/s10052-022-10328-7](https://doi.org/10.1140/epjc/s10052-022-10328-7), NNPDF 4.0

... any Neural Network
model could be implemented!

Linear Model



$$f_w(x) = \phi_0(x) + \sum_{k=1}^N \underbrace{w_k}_{\text{weights}} \underbrace{\phi_k(x)}_{\text{basis functions}}$$

First fully Bayesian PDF fit!

[arXiv:2507.16913](https://arxiv.org/abs/2507.16913), Mark N. Costantini et al.

application: [arXiv:2604.13157](https://arxiv.org/abs/2604.13157)

Bayesian Model Comparison

Bayesian Model Comparison

Comparing two models, M_1 and M_2 , we can compute the *Bayes Factor*:

$$B_{12} = \frac{Z_1}{Z_2} = \frac{p(D|M_1)}{p(D|M_2)}$$

Bayesian Model Comparison

Comparing two models, M_1 and M_2 , we can compute the *Bayes Factor*:

$$B_{12} = \frac{Z_1}{Z_2} = \frac{p(D|M_1)}{p(D|M_2)}$$

- Systematic comparison of what model is favoured by data

Bayesian Model Comparison

Comparing two models, M_1 and M_2 , we can compute the *Bayes Factor*:

$$B_{12} = \frac{Z_1}{Z_2} = \frac{p(D|M_1)}{p(D|M_2)}$$

- Systematic comparison of what model is favoured by data
- Automatically balances goodness-of-fit and model complexity

Bayesian Model Comparison

Comparing two models, M_1 and M_2 , we can compute the *Bayes Factor*:

$$B_{12} = \frac{Z_1}{Z_2} = \frac{p(D|M_1)}{p(D|M_2)}$$

- Systematic comparison of what model is favoured by data
- Automatically balances goodness-of-fit and model complexity

BAYESIAN PDF MODEL COMBINATION

Applications

Applications

1 Bayesian PDF fitting

Applications

- 1 **Bayesian PDF fitting**
- 2 **Benchmarking** different models/fitting methods

Applications

- 1 **Bayesian PDF fitting**
- 2 **Benchmarking** different models/fitting methods
- 3 Perform weighted **PDF combinations**

Applications

- 1 **Bayesian PDF fitting**
- 2 **Benchmarking** different models/fitting methods
- 3 Perform weighted **PDF combinations**

Further development

Applications

- 1 **Bayesian PDF fitting**
- 2 **Benchmarking** different models/fitting methods
- 3 Perform weighted **PDF combinations**

Further development

- 1 Implement more realistic polynomial models

Applications

- 1 Bayesian PDF fitting**
- 2 Benchmarking** different models/fitting methods
- 3** Perform weighted **PDF combinations**

Further development

- 1** Implement more realistic polynomial models
- 2** Implement a Bayesian Neural Network model

Applications

- 1 Bayesian PDF fitting**
- 2 Benchmarking** different models/fitting methods
- 3 Perform weighted PDF combinations**

Further development

- 1** Implement more realistic polynomial models
- 2** Implement a Bayesian Neural Network model
- 3** Simultaneous fit with SM/BSM parameters

Useful links

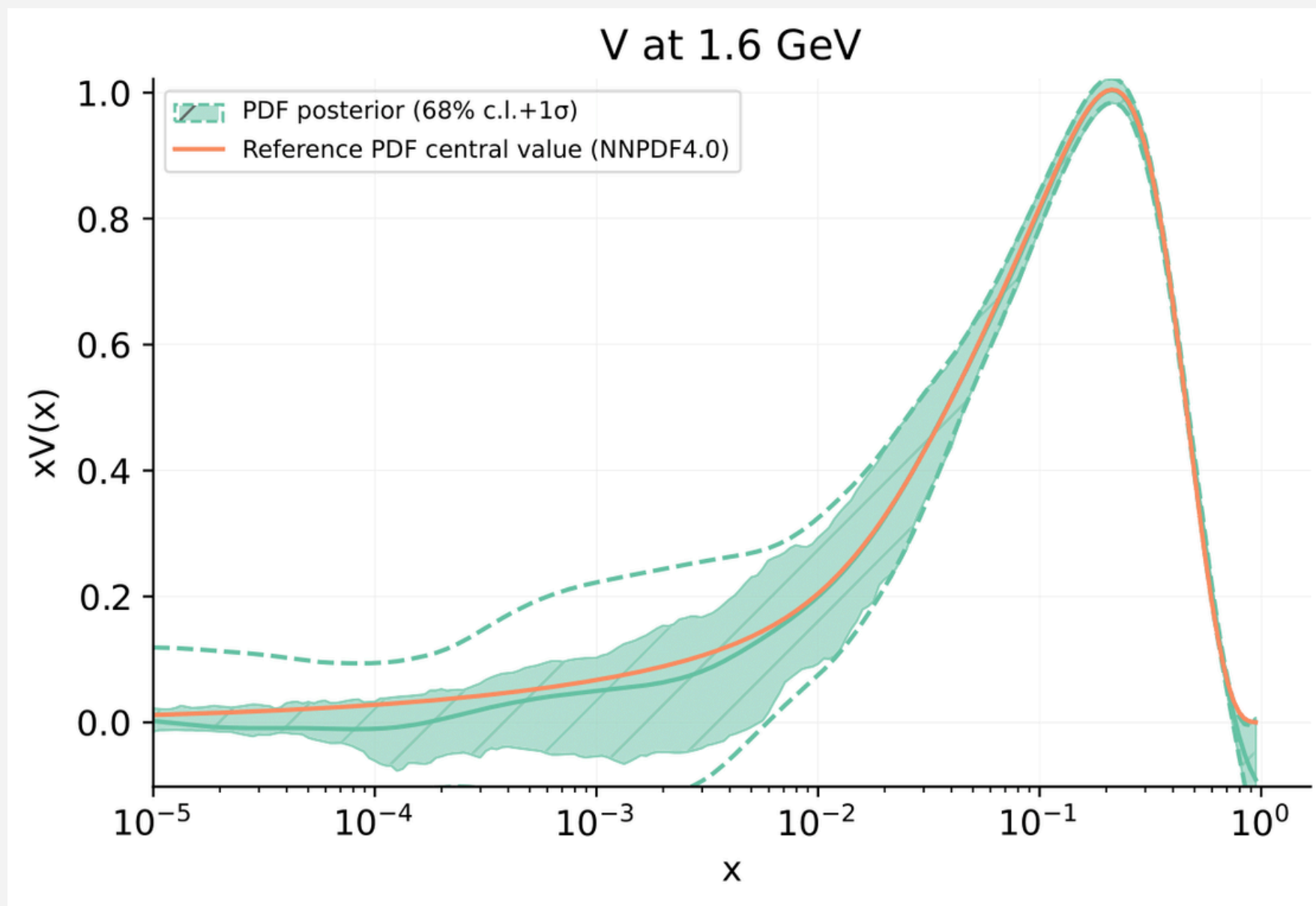
Repository: <https://github.com/HEP-PBSP/colibri>

Documentation: <https://hep-pbsp.github.io/colibri/>

Available models: <https://hep-pbsp.github.io/colibri/available-models/index.html>

Extra slides

Gaussian Processes



PDF described by an N -dimensional Gaussian distribution:

$$\mathbf{f} \sim N(\mathbf{m}, k)$$

Les Houches Parametrisation

$$\begin{aligned}xf_g(x, Q_0) &= A_g x^{\alpha_g} (1-x)^{\beta_g} \\xf_{u_v}(x, Q_0) &= A_{u_v} x^{\alpha_{u_v}} (1-x)^{\beta_{u_v}} (1 + \epsilon_{u_v} \sqrt{x} + \gamma_{u_v} x) \\xf_{d_v}(x, Q_0) &= A_{d_v} x^{\alpha_{d_v}} (1-x)^{\beta_{d_v}} (1 + \epsilon_{d_v} \sqrt{x} + \gamma_{d_v} x) \\xf_{\Sigma}(x, Q_0) &= A_{\Sigma} x^{\alpha_{\Sigma}} (1-x)^{\beta_{\Sigma}}.\end{aligned}$$

Sum rules:

$$A_g \int_0^1 x^{\alpha_g} (1-x)^{\beta_g} dx + A_{\Sigma} \int_0^1 x^{\alpha_{\Sigma}} (1-x)^{\beta_{\Sigma}} dx = 1$$

$$A_{u_v} \int x^{\alpha_{u_v}-1} (1-x)^{\beta_{u_v}} (1 + \epsilon_{u_v} \sqrt{x} + \gamma_{u_v} x) dx = 2$$

$$A_{d_v} \int x^{\alpha_{d_v}-1} (1-x)^{\beta_{d_v}} (1 + \epsilon_{d_v} \sqrt{x} + \gamma_{d_v} x) dx = 1$$

Assumptions:

- $\Sigma = \underbrace{u + \bar{u}}_{= 0.4\Sigma} + \underbrace{d + \bar{d}}_{= 0.4\Sigma} + \underbrace{s + \bar{s}}_{= 0.2\Sigma}$
- $\bar{d} = \bar{u}$
- $\epsilon_g, \gamma_g, \epsilon_{\Sigma}, \gamma_{\Sigma} = 0$



**13 free
parameters**