Computational particle physics

Why I wish I had taken a course like FYS3150/4150

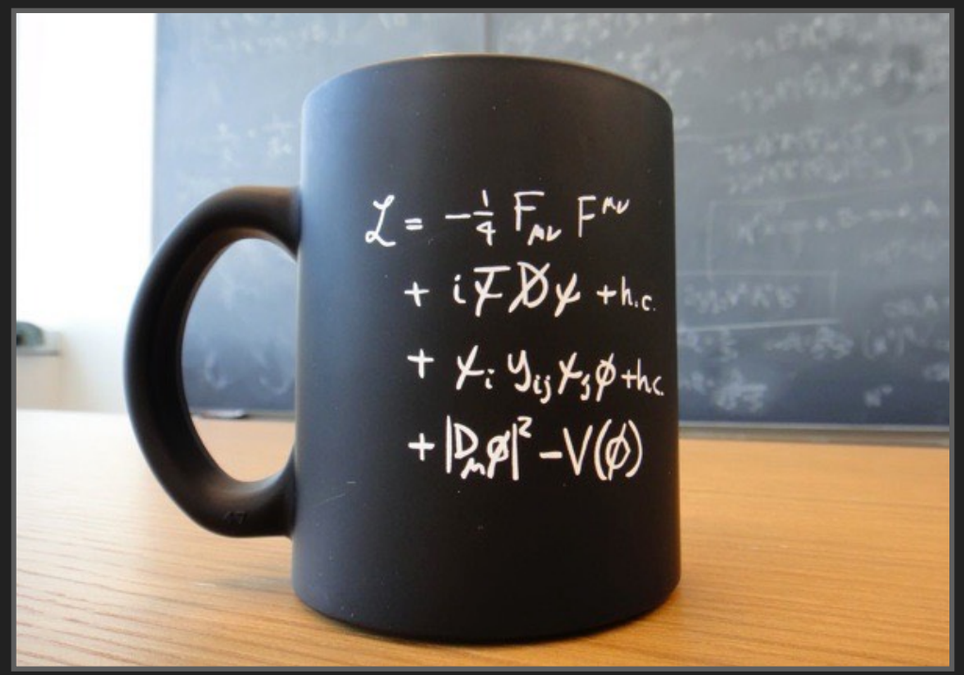
Outline

1. **Chasing new physics with global fits**
2. **In high-dimensional parameter space no one can hear you scream…**
3. **FYS3150 is everywhere!**

1. Chasing new physics…

$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$$
$$+ i\bar{\psi} \not{D} \psi + h.c.$$
$$+ \chi_i y_{ij} \chi_j \phi + h.c.$$
$$+ |D_\mu \phi|^2 - V(\phi)$$

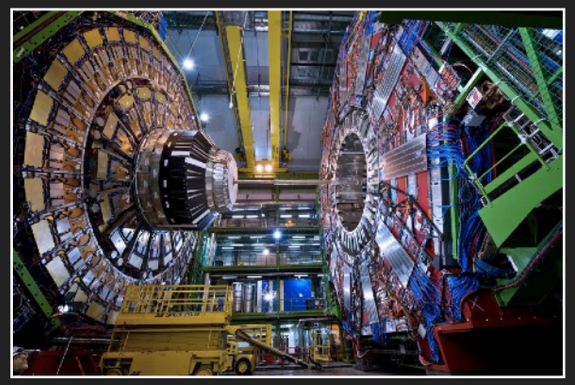
Standard model, schmandard model…

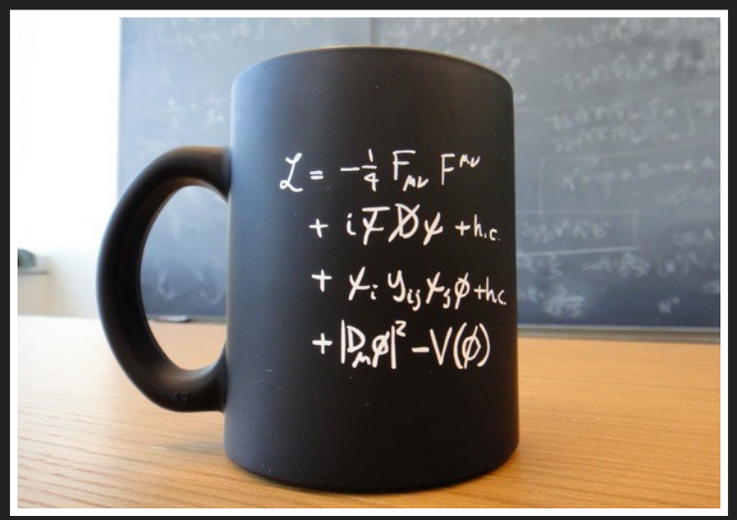


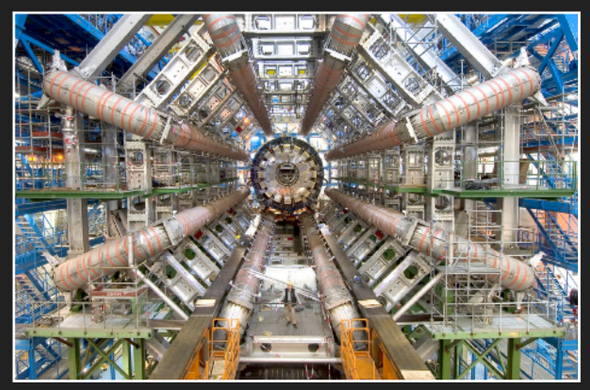
- What is dark matter? And dark energy?
- Why is the Higgs boson light?
- Why are the neutrinos *really* light?
- Why do the elementary particles come in three «generations»?
- How did the Universe end up with a tiny bit more matter (us) than antimatter?
- …

Need some «new physics», a.k.a. «beyond-the-Standard-Model» (BSM) physics

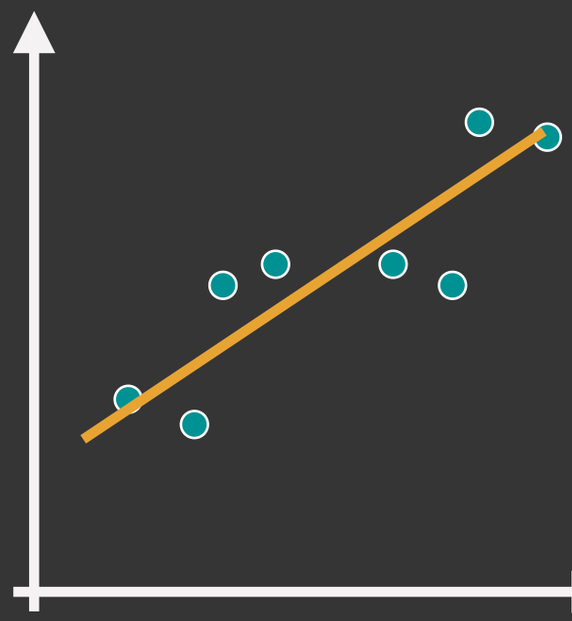
…with global fits




$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$
$+ i\bar{\psi} \not{D} \psi + h.c.$
$+ \psi_i y_{ij} \psi_j \phi + h.c.$
$+ |D_\mu \phi|^2 - V(\phi)$

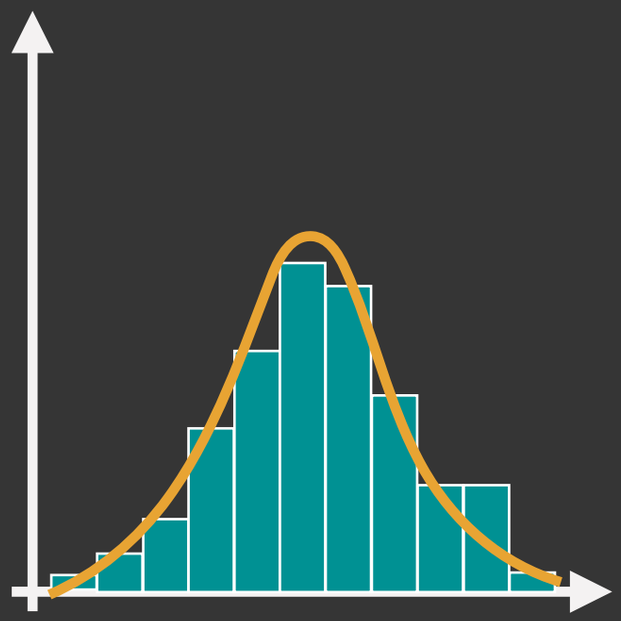


Statistical fit

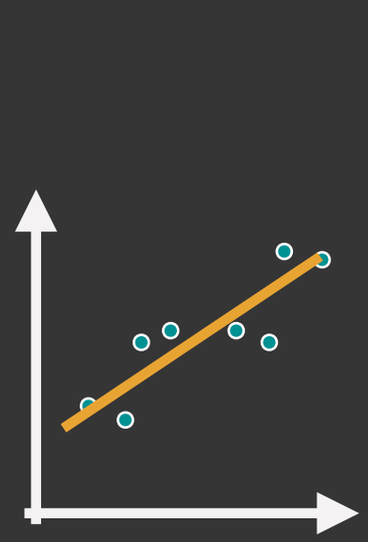


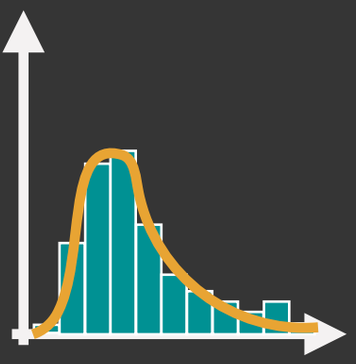
Some observable
Some model

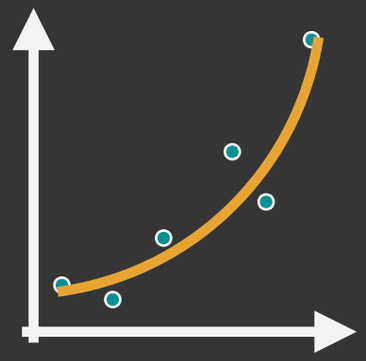
Some other observable
Some other model



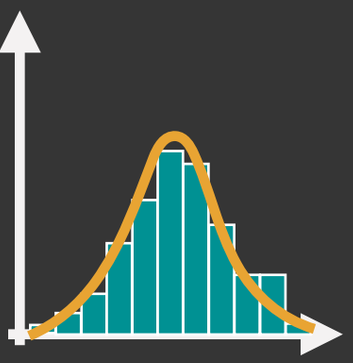
Global fit

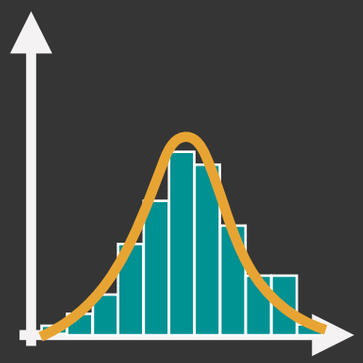


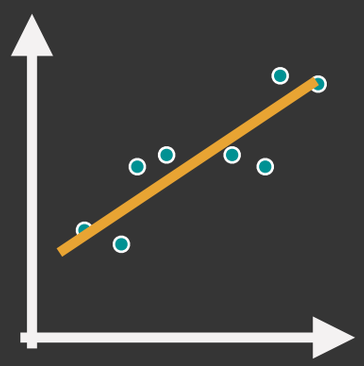


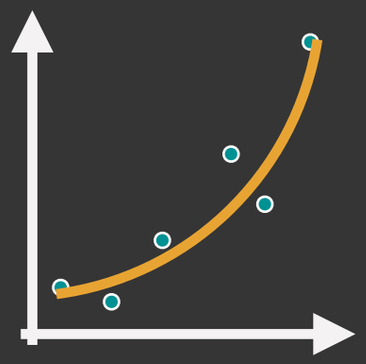


Many observables
One theory

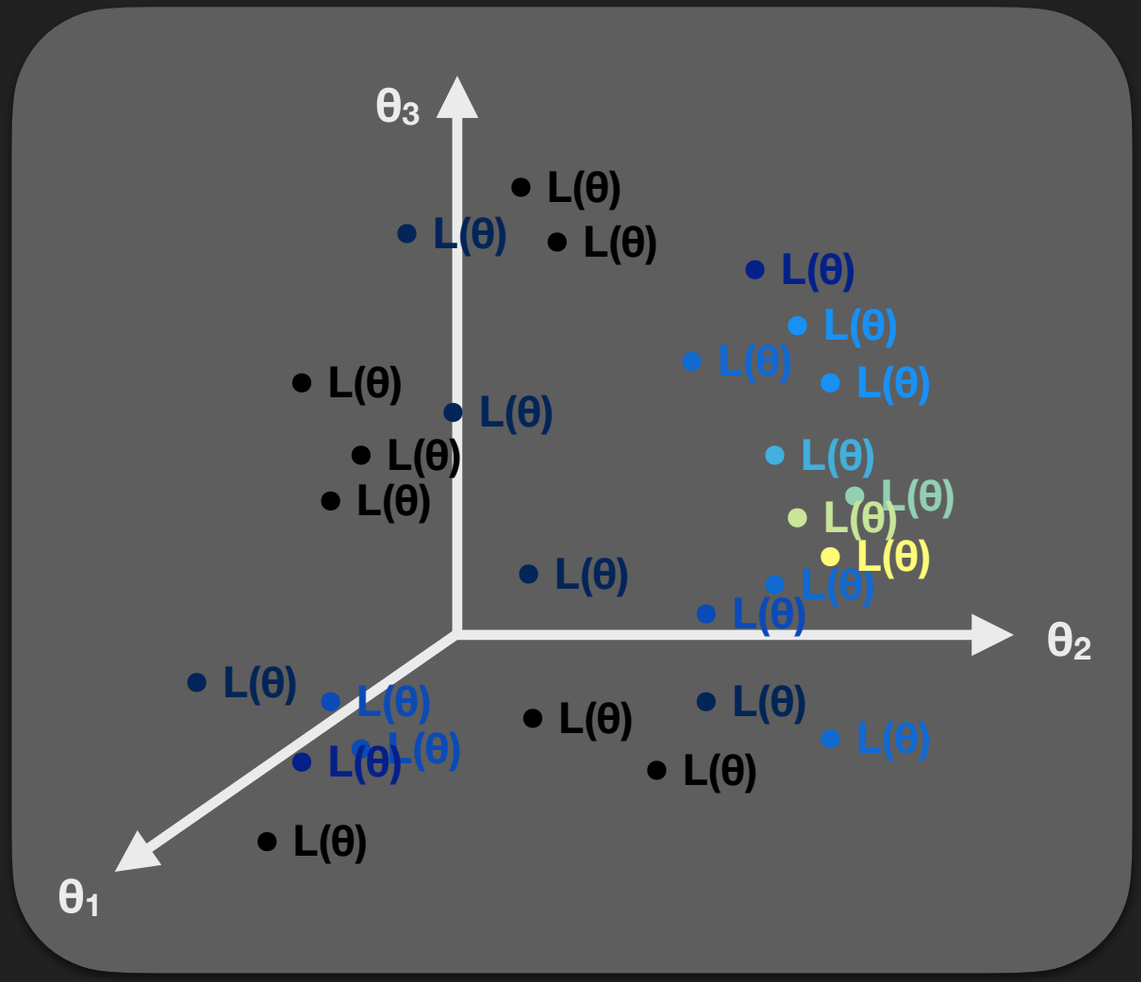




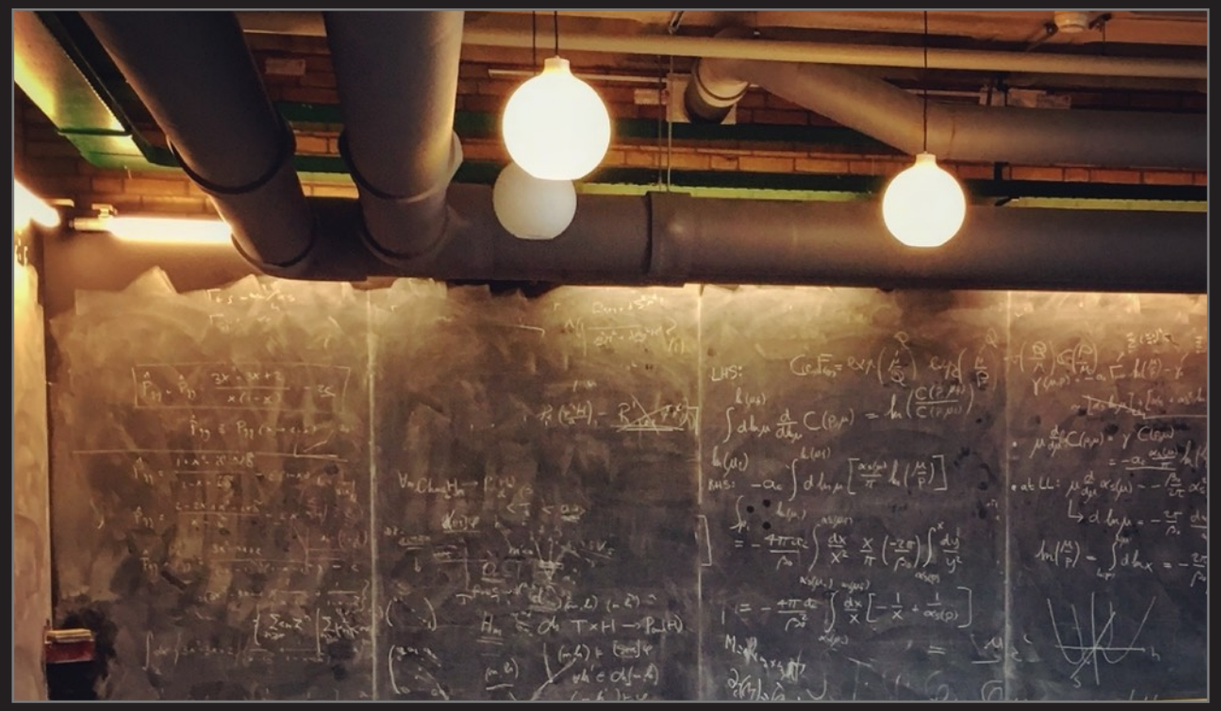




- **Explore the model parameter space** (θ_1, θ_2, θ_3, …)
- At every point θ: **calculate predictions(θ) → evaluate likelihood L(θ)**



- Region of highest L(θ) or lnL(θ): **model's best simultaneous fit to all data**
  (but not necessarily a *good* fit, or the most probable θ…)



JHEP

Published for SISSA by Springer

Received: September 15, 2017
Revised: February 5, 2018
Accepted: March 18, 2018
Published: March 27, 2018

Search for electroweak production of charginos and neutralinos in multilepton final states in proton-proton collisions at $\sqrt{s} = 13$ TeV

The CMS collaboration

E-mail: cms-publication-committee-chair@cern.ch

Abstract: Results are presented from a search for the direct electroweak production of charginos and neutralinos in signatures with either two or more leptons (electrons or muons) of the same electric charge, or with three or more leptons, which can include up to two hadronically decaying tau leptons. The results are based on a sample of proton-proton collision data collected at $\sqrt{s} = 13$ TeV, recorded with the CMS detector at the LHC, corresponding to an integrated luminosity of 35.9 fb^{-1}. The observed event yields are consistent with the expectations based on the standard model. The results are interpreted in simplified models of supersymmetry describing various scenarios for the production and decay of charginos and neutralinos. Depending on the model parameters chosen, mass values between 180 GeV and 1150 GeV are excluded at 95% CL. These results significantly extend the parameter space probed for these particles in searches at the LHC. In addition, results are presented in a form suitable for alternative theoretical interpretations.

Keywords: Hadron-Hadron scattering (experiments), Supersymmetry

ArXiv ePrint: 1709.05406

https://doi.org/10.1007/JHEP03(2018)166

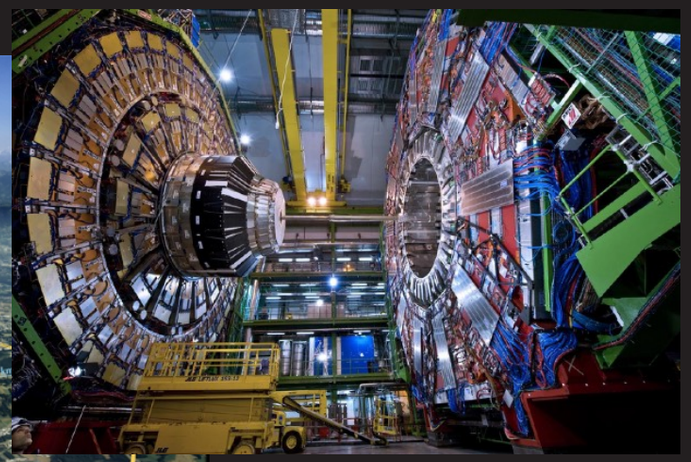
JHEP03(2018)166

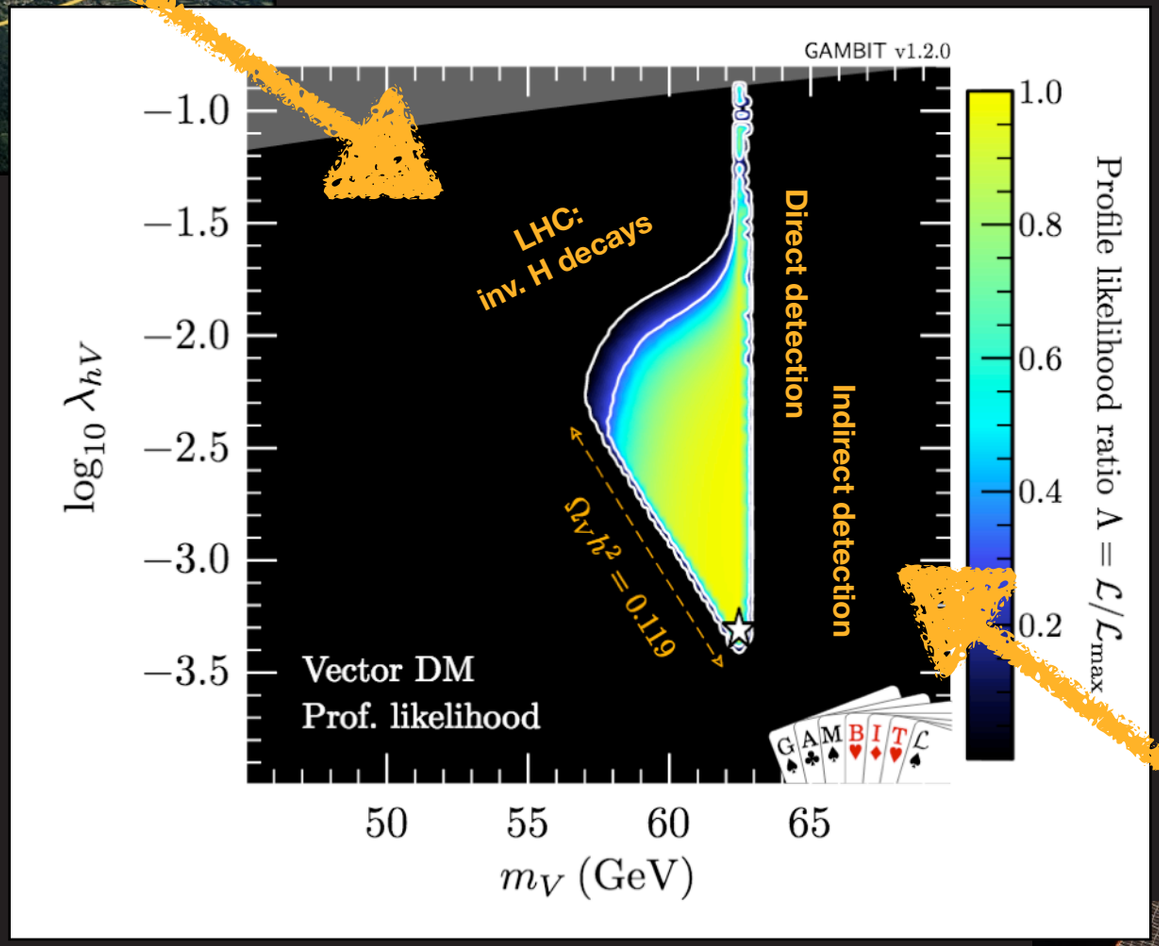
```
// Increment signal region counters: 2 same-sign leptons
if (preselection && nSignalLeptons==2 && nSignalTaus==0 && met>60 && conversion_veto)
  if (signalLeptons.at(0)->pid()*signalLeptons.at(1)->pid()>0) {
    if ((signalLeptons.at(0)->abspid()==11 && signalLeptons.at(0)->pT()>25) || (signal

      bool pp = false;
      bool mm = false;
      if(signalLeptons.at(0)->pid() > 0)pp = true;
      if(signalLeptons.at(0)->pid() < 0)mm = true;

      if (num_ISRjets==0) {

        // The 0 jet regions
        if(mT < 100 && pT_ll < 50 && met < 100) _numSR["SS01"]++;
        if(mT < 100 && pT_ll < 50 && met >= 100 && met < 150 && pp) _numSR["SS02"]++;
        if(mT < 100 && pT_ll < 50 && met >= 100 && met < 150 && mm) _numSR["SS03"]++;
        if(mT < 100 && pT_ll < 50 && met >= 150 && met < 200) _numSR["SS04"]++;
        if(mT < 100 && pT_ll < 50 && met > 200) _numSR["SS05"]++;
        if(mT < 100 && pT_ll > 50 && met < 100) _numSR["SS06"]++;
        if(mT < 100 && pT_ll > 50 && met >= 100 && met < 150 && pp) _numSR["SS07"]++;
        if(mT < 100 && pT_ll > 50 && met >= 100 && met < 150 && mm) _numSR["SS08"]++;
        if(mT < 100 && pT_ll > 50 && met >= 150 && met < 200) _numSR["SS09"]++;
        if(mT < 100 && pT_ll > 50 && met > 200) _numSR["SS10"]++;
```

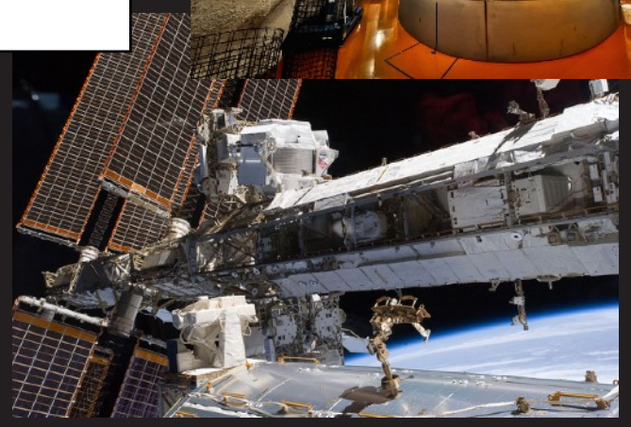
1. akvelles@r000u07l02:~ (ssh)

```
*******************************************************************************
* Welcome to MARCONI /                                                        *
*            MARCONI-fusion @ CINECA - NeXtScale cluster - CentOS 7.2!        *
*                                                                             *
* KNL partition - 3600 Compute nodes with:                                    *
*    - 1*68-core Intel(R) Knights Landing @ 1.40GHz                           *
*    - 16 GB MCDRAM + 96 GB RAM                                               *
* SKL partition - 1512+792 nodes with:                                        *
*    - 2*24-core Intel Xeon 8160 CPU @ 2.10GHz                                *
*    - 192 GB DDR4 RAM                                                        *
* Intel OmniPath (100Gb/s) high-performance network                           *
* SLURM 18.08                                                                 *
*                                                                             *
* For a guide on Marconi:                                                     *
* wiki.u-gov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide           *
* For support: superc@cineca.it                                               *
*******************************************************************************
*   This system is in its complete configuration and is in full-production    *
===============================================================================
===============================================================================
 IN EVIDENCE:
 - An automatic cleaning procedure for the $CINECA_SCRATCH is active, each day
   all files older than 40 days will be cancelled.
 - A new version of "module" is installed and based on profiles.
   command to identify the correct profile ("modmap -h" for help)
 - The modules of Intel and Intelmpi version 2017 have been moved
   profile to "archive". Consequently, all libraries and tools co
   2017 Intel/Intelmpi and located under "base" and "advanced" pr
   been moved to "archive" profile.
===============================================================
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = (unset)
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
manpath: can't set the locale; make sure $LC_* and $LANG are corr
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8)
[akvelles@r000u07l02 ~]$
[akvelles@r000u07l02 ~]$
```


LHCb
ATLAS
CERN
SUISSE
FRANCE
CMS
LHC 27 km




GAMBIT v1.2.0
LHC:
inv. H decays
Direct detection
Indirect detection
$\Omega_V h^2 = 0.119$
Vector DM
Prof. likelihood
$\log_{10} \lambda_{hV}$
m_V (GeV)
Profile likelihood ratio $\Lambda = \mathcal{L}/\mathcal{L}_{\max}$



GAMBIT: The Global And Modular BSM Inference Tool

gambit.hepforge.org github.com/GambitBSM EPJC 77 (2017) 784 arXiv:1705.07908

- Extensive model database, beyond SUSY
- Fast definition of new datasets, theories
- Extensive observable/data libraries
- Plug&play scanning/physics/likelihood packages
- Various statistical options (frequentist /Bayesian)
- Fast LHC likelihood calculator
- Massively parallel
- Fully open-source

Members of: ATLAS, Belle-II, CLiC, CMS, CTA, Fermi-LAT, DARWIN, IceCube, LHCb, SHiP, XENON

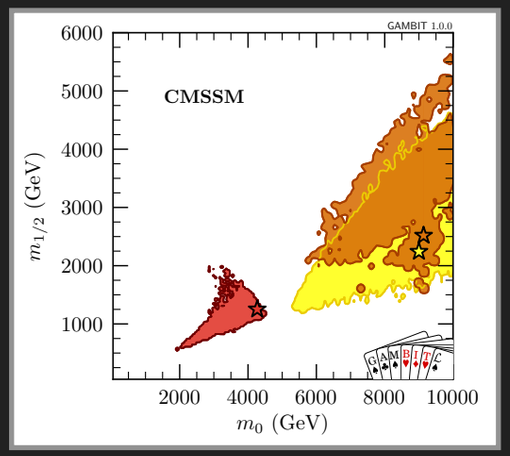
Authors of: BubbleProfiler, Capt'n General, Contur, DarkAges, DarkSUSY, DDCalc, DirectDM, Diver, EasyScanHEP, ExoCLASS, FlexibleSUSY, gamLike, GM2Calc, HEPLike, IsaTools, MARTY, nuLike, PhaseTracer, PolyChord, Rivet, SOFTSUSY, SuperIso, SUSY-AI, xsec, Vevacious, WIMPSim

Recent collaborators: P Athron, C Balázs, A Beniwal, S Bloor, T Bringmann, A Buckley, J-E Camargo-Molina, C Chang, M Chrzaszcz, J Conrad, J Cornell, M Danninger, J Edsjö, T Emken, A Fowlie, T Gonzalo, W Handley, J Harz, S Hoof, F Kahlhoefer, A Kvellestad, P Jackson, D Jacob, C Lin, N Mahmoudi, G Martinez, MT Prim, A Raklev, C Rogan, R Ruiz, P Scott, N Serra, P Stöcker , W. Su, A Vincent, C Weniger, M White, Y Zhang, ++

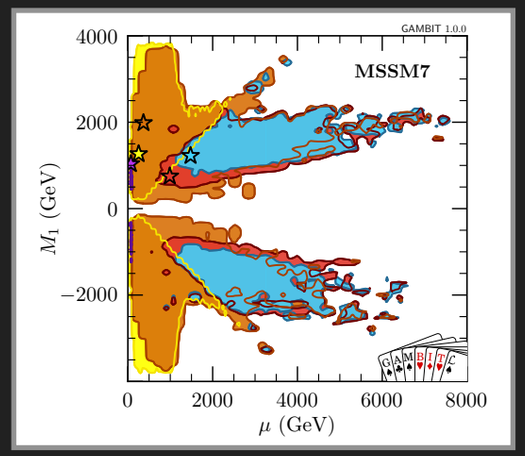
70+ participants in many experiments and numerous major theory codes

gambitbsm.github.io

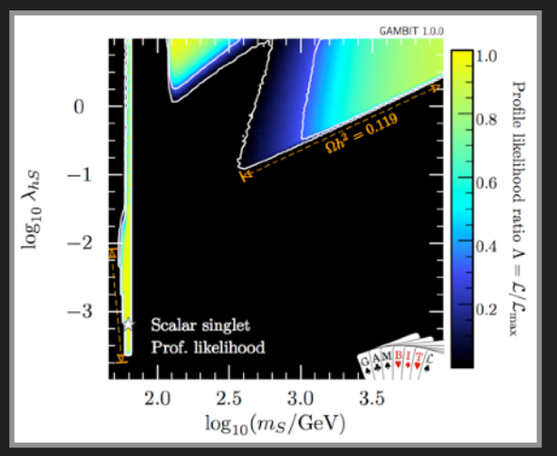
www.mn.uio.no/fysikk/english/research/projects/gambit/



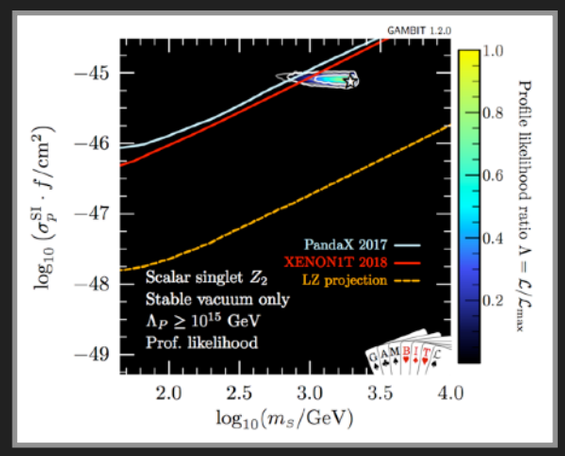
GUT-scale SUSY: 1705.07935



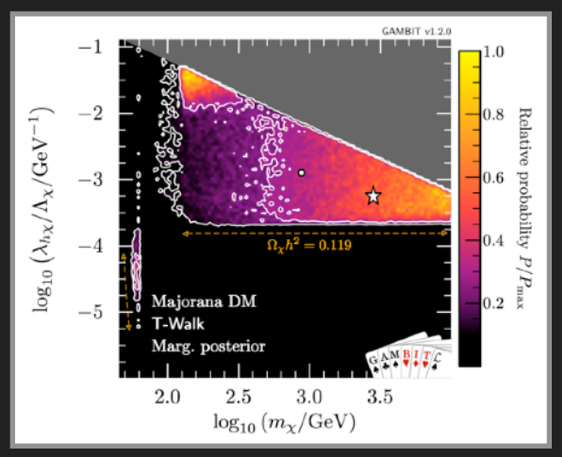
MSSM7: 1705.07917



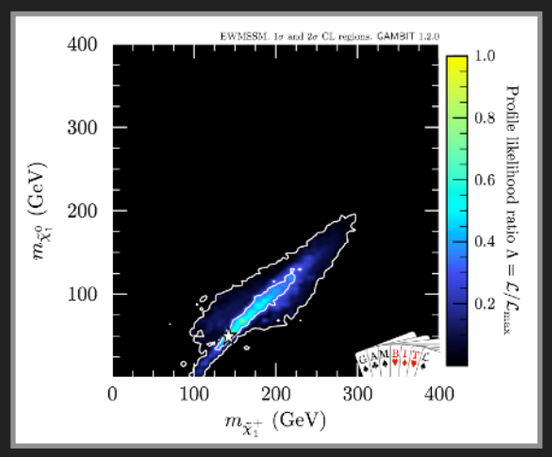
Scalar Higgs portal DM: 1705.07931



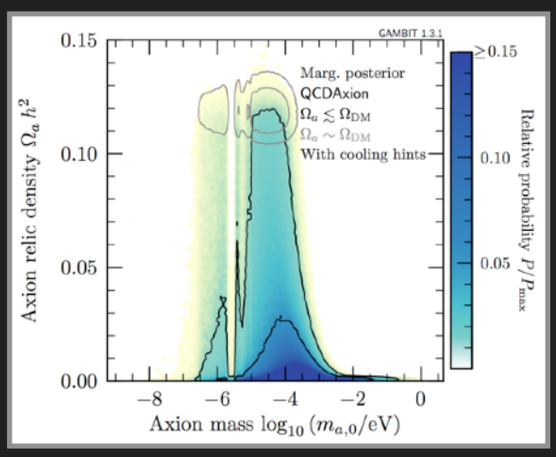
Scalar Higgs portal DM w/ vac. stability: 1806.11281



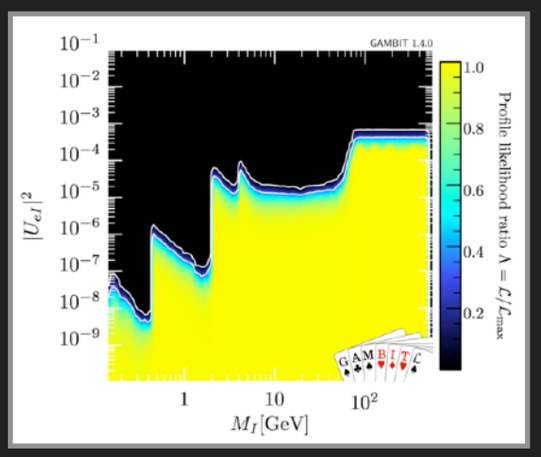
Vector and fermion Higgs portal DM: 1808.10465



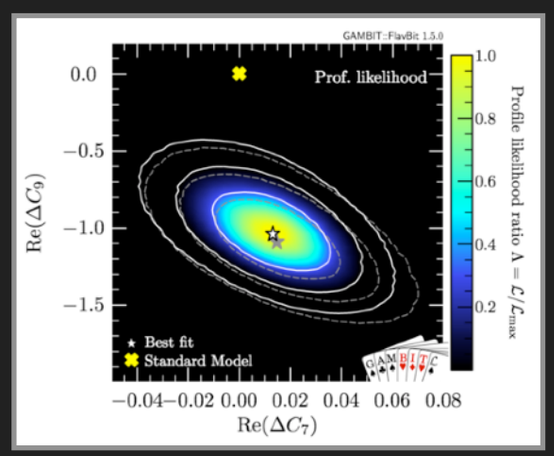
EW-MSSM: 1809.02097



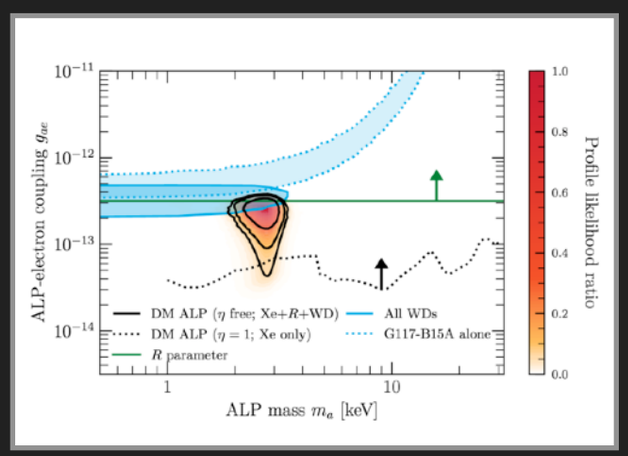
Axion-like particles: 1810.07192



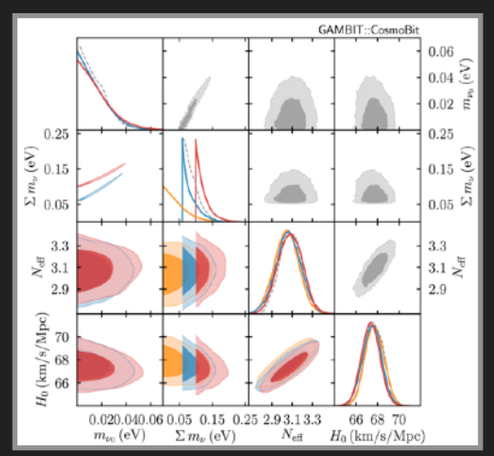
Right-handed neutrinos: 1908.02302



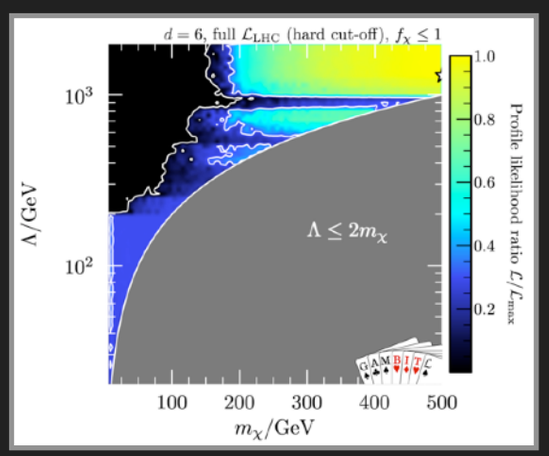
Flavour EFT: 2006.03489



More axion-like particles: 2006.03489



Neutrinos and cosmo: 2009.03287



Dark matter EFTs:2106.02056

2. In high-dimensional parameter space no one can hear you scream…

[large number of observables]

$\times$

[long calculation time per observable per parameter point]

$\times$

[huge number of points required to explore parameter space]

$\approx$

∞

[large number of observables]

$\times$

[long calculation time per observable per parameter point]

$\times$

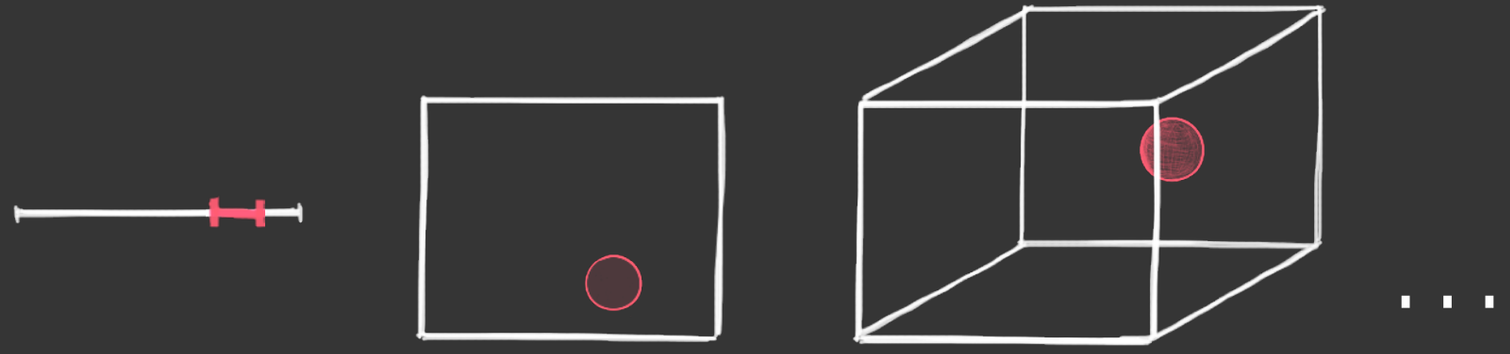
[huge number of points required to explore parameter space]

$\approx$

∞

Finding interesting parameter regions gets harder with increasing number of dimensions…

$$\lim_{D \to \infty} \frac{V_{\text{interesting}}}{V_{\text{total}}} = 0$$

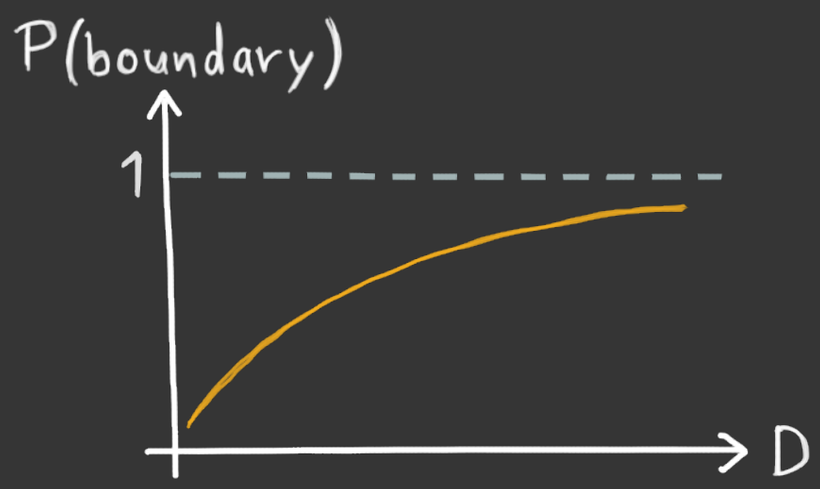


…so simply picking points «at random» will be **highly inefficient**…

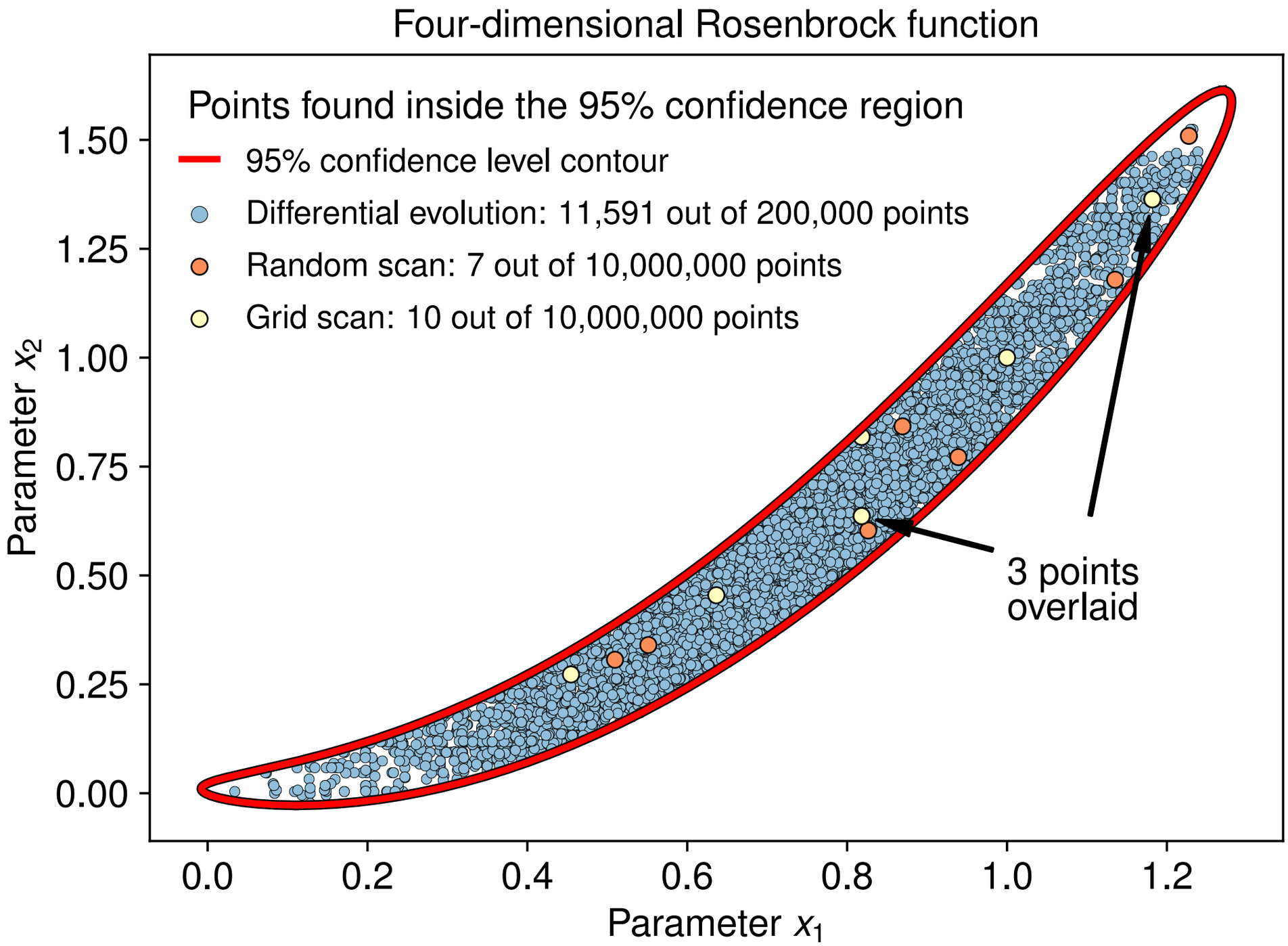
…and it will mainly explore the **boundary** of the parameter space!

$$\vec{x} = (x_1, x_2, \ldots, x_D) \quad x_i \sim U(0,1)$$

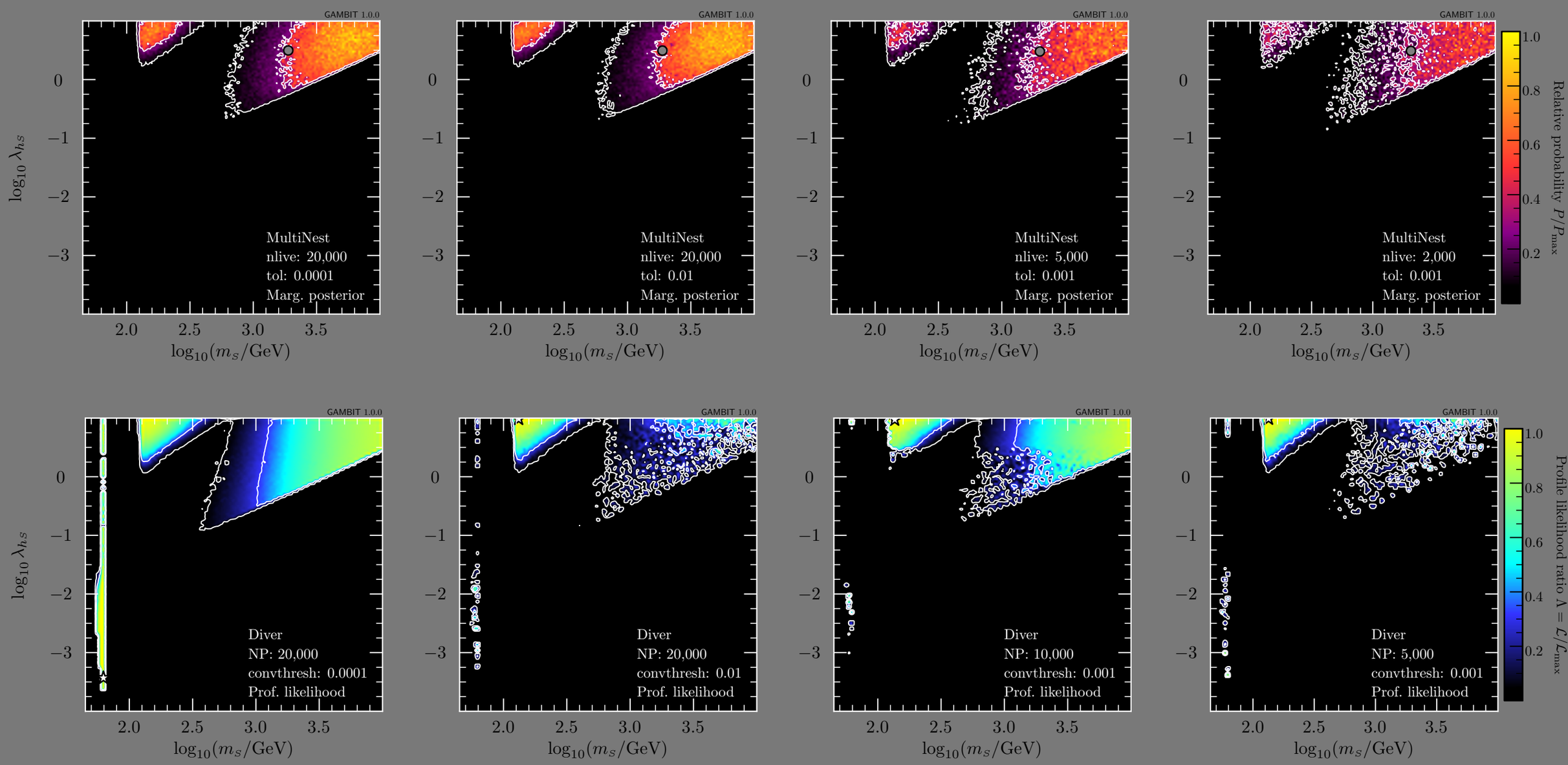
$$P(\text{boundary}) = 1 - P(\text{not boundary}) = 1 - p^D$$



Need to use some **smart** sampling algorithms!



arxiv:2012.09874



For comparisons of sampling algorithms, see **arxiv:1705.07959** and **arxiv:2101.04525**

Mostly used in GAMBIT: **differential evolution** (Diver), **nested sampling** (MultiNest, PolyChord w/ fast-slow)

[large number of observables]

$\times$

[long calculation time per observable per parameter point]

$\times$

[huge number of points required to explore parameter space]

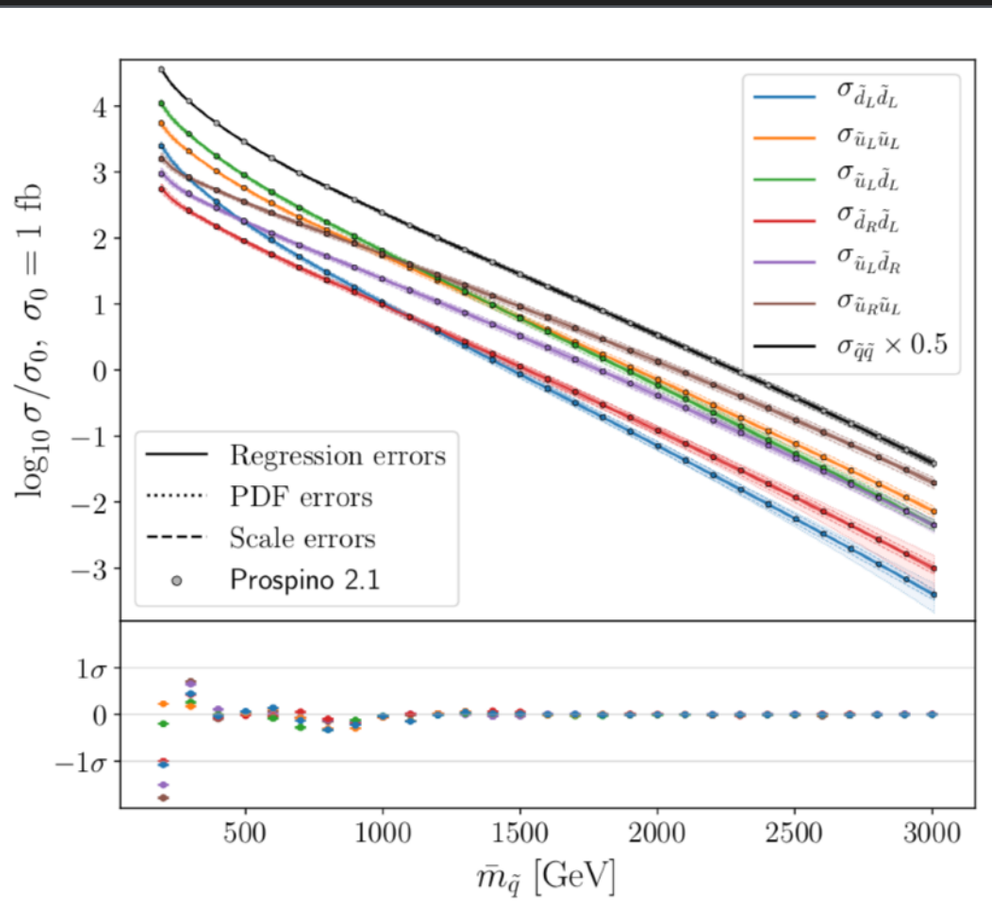
$\approx$

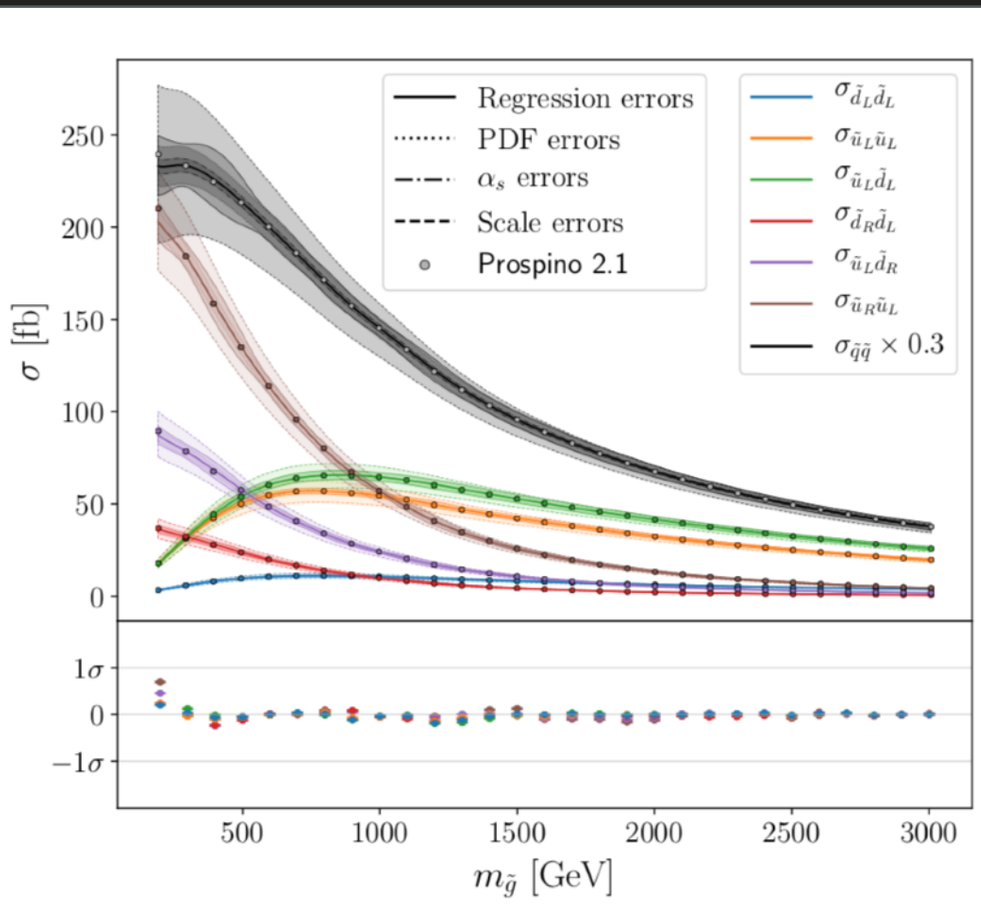
∞

- A typical BSM global fit requires **O(10M) parameter samples**

- Need **fast** theory predictions with **reliable uncertainty estimates**

- Rev. Thomas Bayes to the rescue!
- Bayesian probability distributions for *functions*?
  → *Gaussian process regression!*
- Train fast approximations to replace the slow physics computations





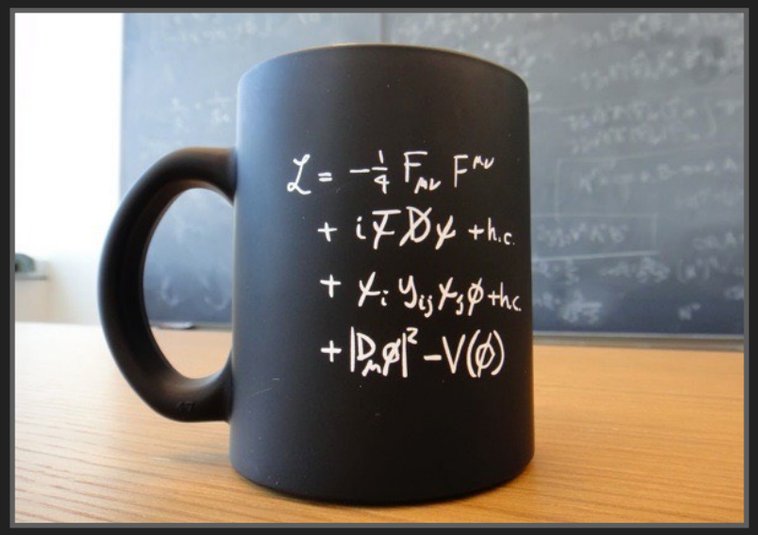
3. FYS3150 is everywhere!

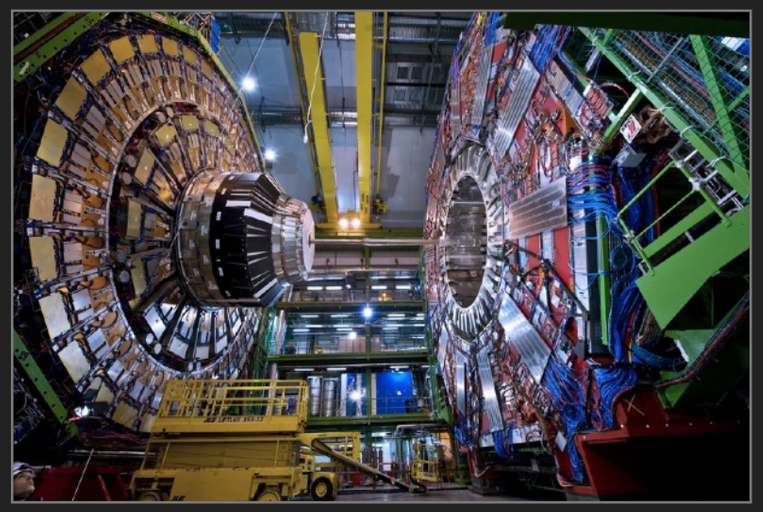
- C++ (and C, Python, Fortran, bash scripts, …)
- Compilation, linking
- Proper code structure
- Parallelisation
- Git
- Debugging!
- Numerical errors
- Code optimisation

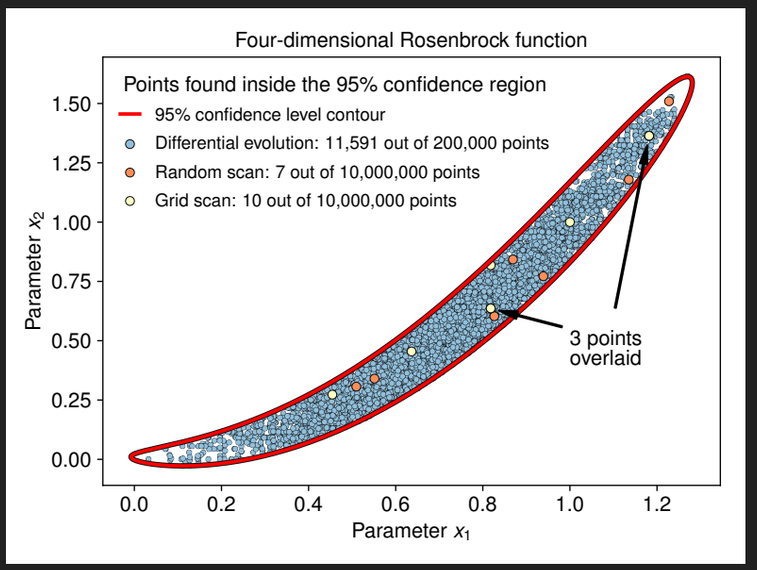
```
// Increment signal region counters: 2 same-sign leptons
if (preselection && nSignalLeptons==2 && nSignalTaus==0 && met>60 && conversion_veto)
  if (signalLeptons.at(0)->pid()*signalLeptons.at(1)->pid()>0) {
    if ((signalLeptons.at(0)->abspid()==11 && signalLeptons.at(0)->pT()>25) || (signal

      bool pp = false;
      bool mm = false;
      if(signalLeptons.at(0)->pid() > 0)pp = true;
      if(signalLeptons.at(0)->pid() < 0)mm = true;

      if (num_ISRjets==0) {

        // The 0 jet regions
        if(mT < 100 && pT_ll < 50 && met < 100) _numSR["SS01"]++;
        if(mT < 100 && pT_ll < 50 && met >= 100 && met < 150 && pp) _numSR["SS02"]++;
        if(mT < 100 && pT_ll < 50 && met >= 100 && met < 150 && mm) _numSR["SS03"]++;
        if(mT < 100 && pT_ll < 50 && met >= 150 && met < 200) _numSR["SS04"]++;
        if(mT < 100 && pT_ll < 50 && met > 200) _numSR["SS05"]++;
        if(mT < 100 && pT_ll > 50 && met < 100) _numSR["SS06"]++;
        if(mT < 100 && pT_ll > 50 && met >= 100 && met < 150 && pp) _numSR["SS07"]++;
        if(mT < 100 && pT_ll > 50 && met >= 100 && met < 150 && mm) _numSR["SS08"]++;
        if(mT < 100 && pT_ll > 50 && met >= 150 && met < 200) _numSR["SS09"]++;
        if(mT < 100 && pT_ll > 50 && met > 200) _numSR["SS10"]++;
```

```
1. akvelles@r000u07l02:~ (ssh)
*******************************************************************************
* Welcome to MARCONI /                                                        *
*            MARCONI-fusion @ CINECA - NeXtScale cluster - CentOS 7.2!        *
*                                                                             *
* KNL partition - 3600 Compute nodes with:                                    *
*    - 1*68-core Intel(R) Knights Landing @ 1.40GHz                           *
*    - 16 GB MCDRAM + 96 GB RAM                                               *
* SKL partition - 1512+792 nodes with:                                        *
*    - 2*24-core Intel Xeon 8160 CPU @ 2.10GHz                                *
*    - 192 GB DDR4 RAM                                                        *
* Intel OmniPath (100Gb/s) high-performance network                           *
* SLURM 18.08                                                                 *
*                                                                             *
* For a guide on Marconi:                                                     *
* wiki.u-gov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide           *
* For support: superc@cineca.it                                               *
*******************************************************************************
*   This system is in its complete configuration and is in full-production    *
*******************************************************************************
```

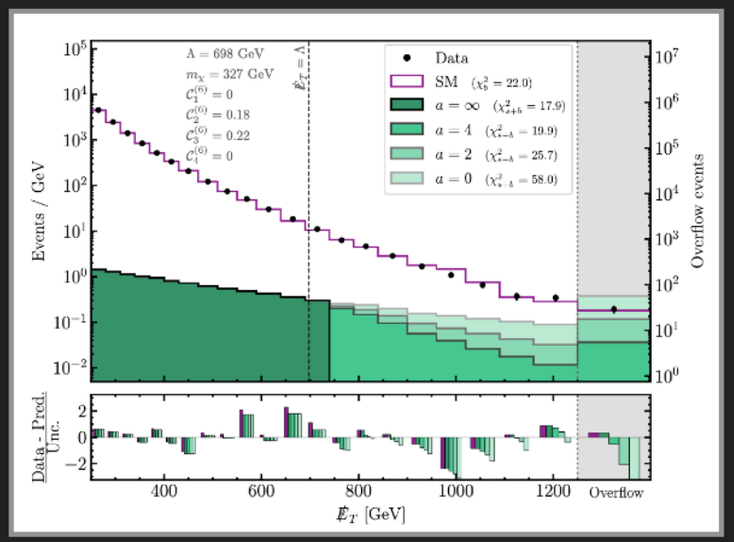
- Predict particle masses?
  Solve numerical boundary value problem!
- Predict particle properties?
  Diagonalise a matrix numerically!
- Predict interaction probabilities?
  Numerical integration!
- Simulate particle interactions?
  Monte Carlo methods!
- Simulate detector effects?
  Monte Carlo methods!
- Explore the theory parameter space?
  Monte Carlo methods! (e.g. MCMC)
- Avoid being fooled by some data/simulations?
  Understand probability!

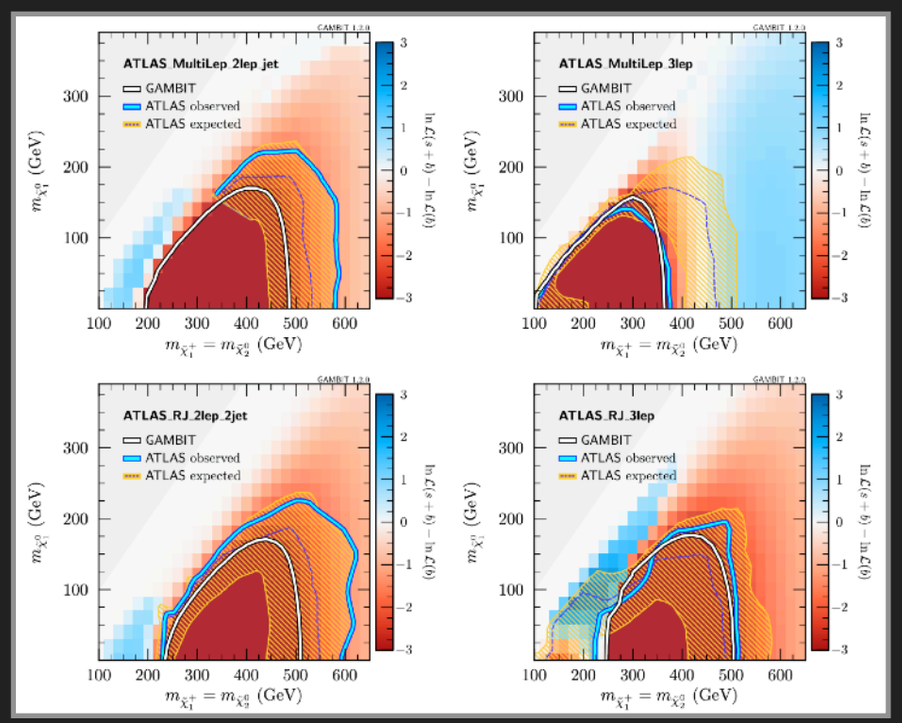


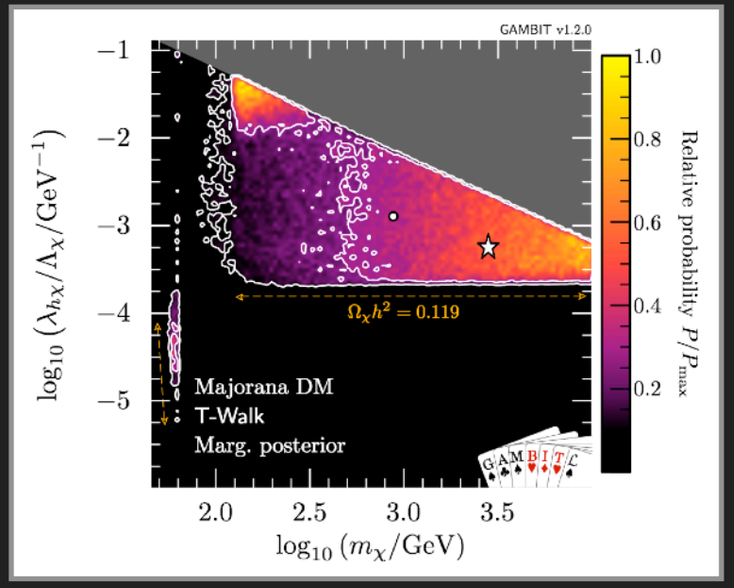


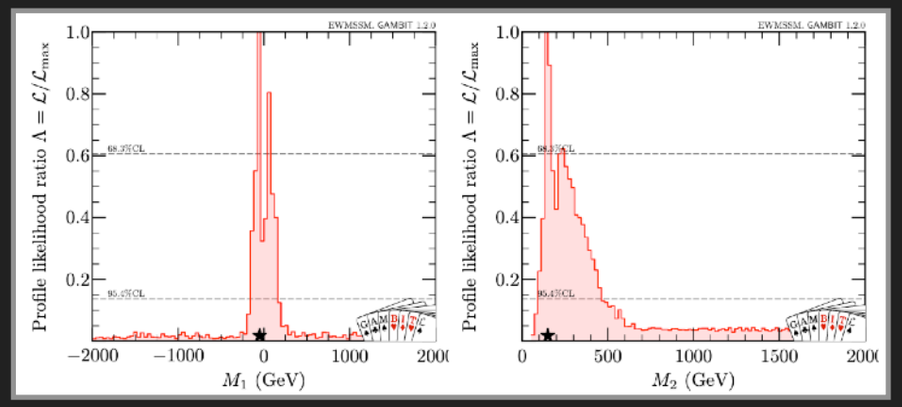


- Visualisation
- Scientific writing





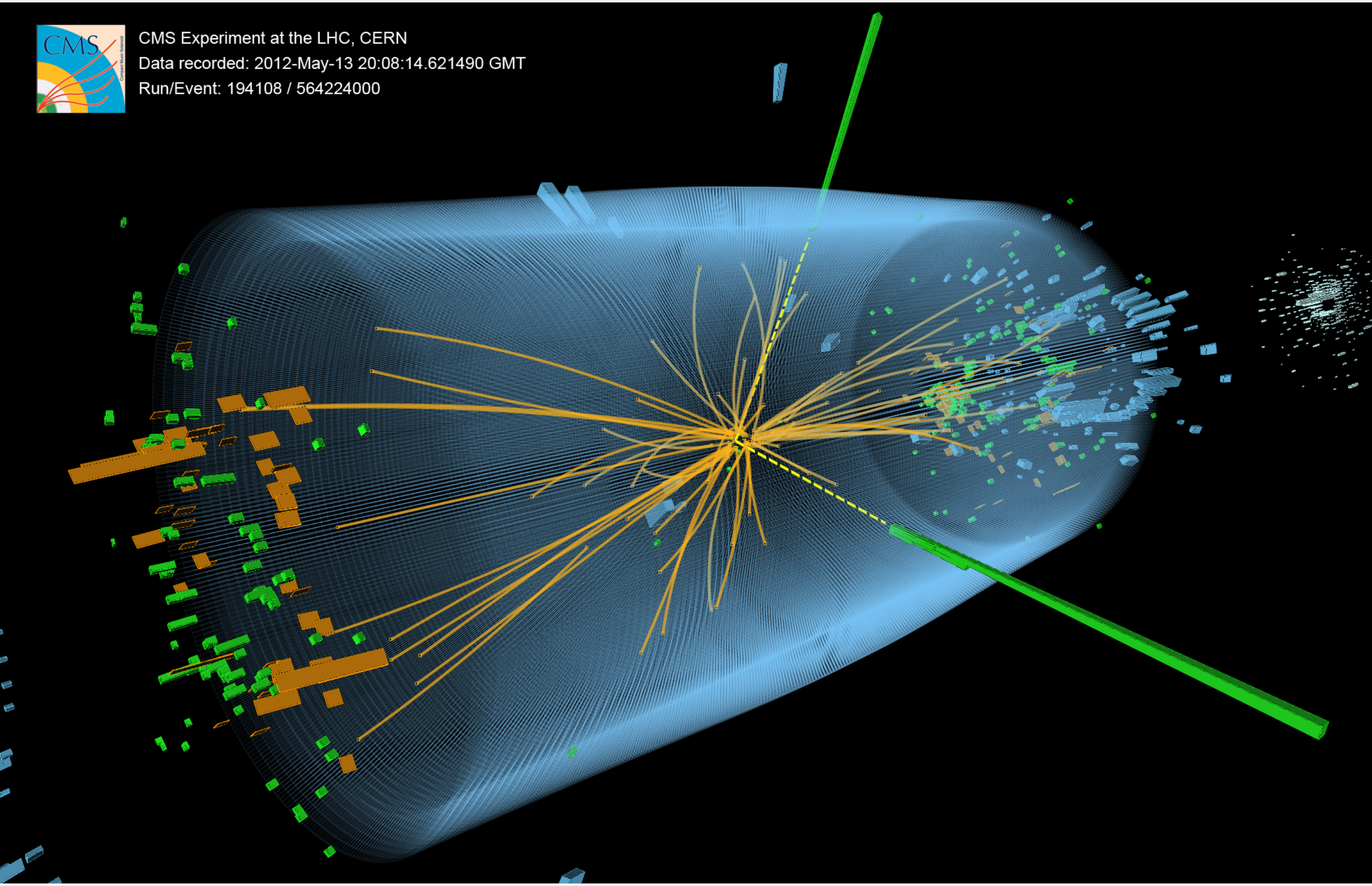




Anders Kvellestad

Bonus tracks

3. Modelling a collision at the LHC


CMS Experiment at the LHC, CERN
Data recorded: 2012-May-13 20:08:14.621490 GMT
Run/Event: 194108 / 564224000

Step 1: **Generating «the hard process»**

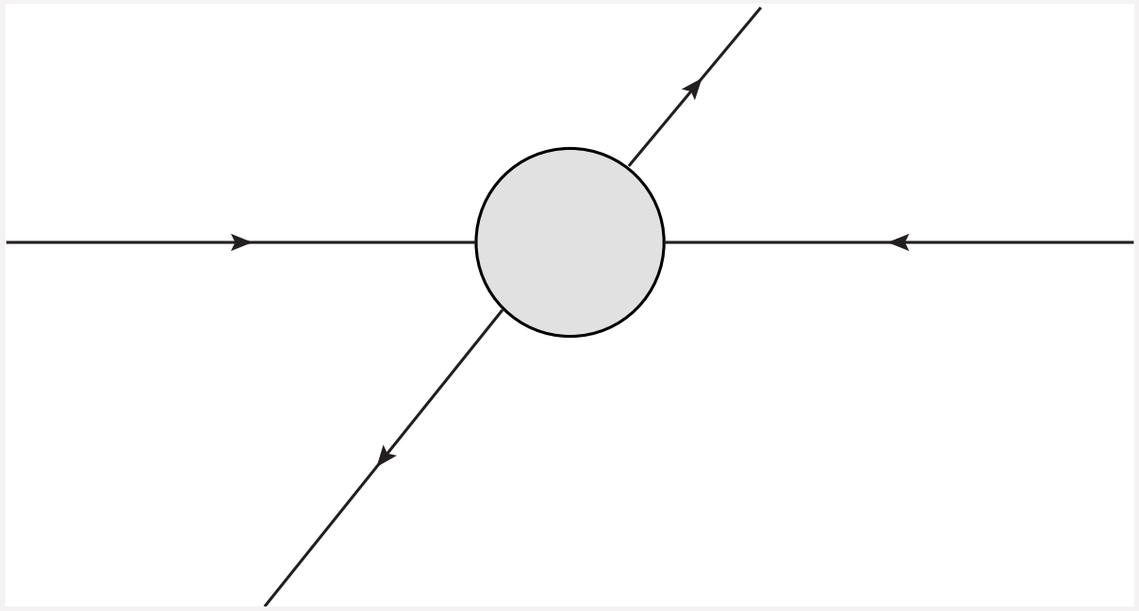


Illustration by A. Papaefstathiou, arXiv:1108.5599

Step 2: **Decay of heavy, short-lived states** («heavy resonance decay»)

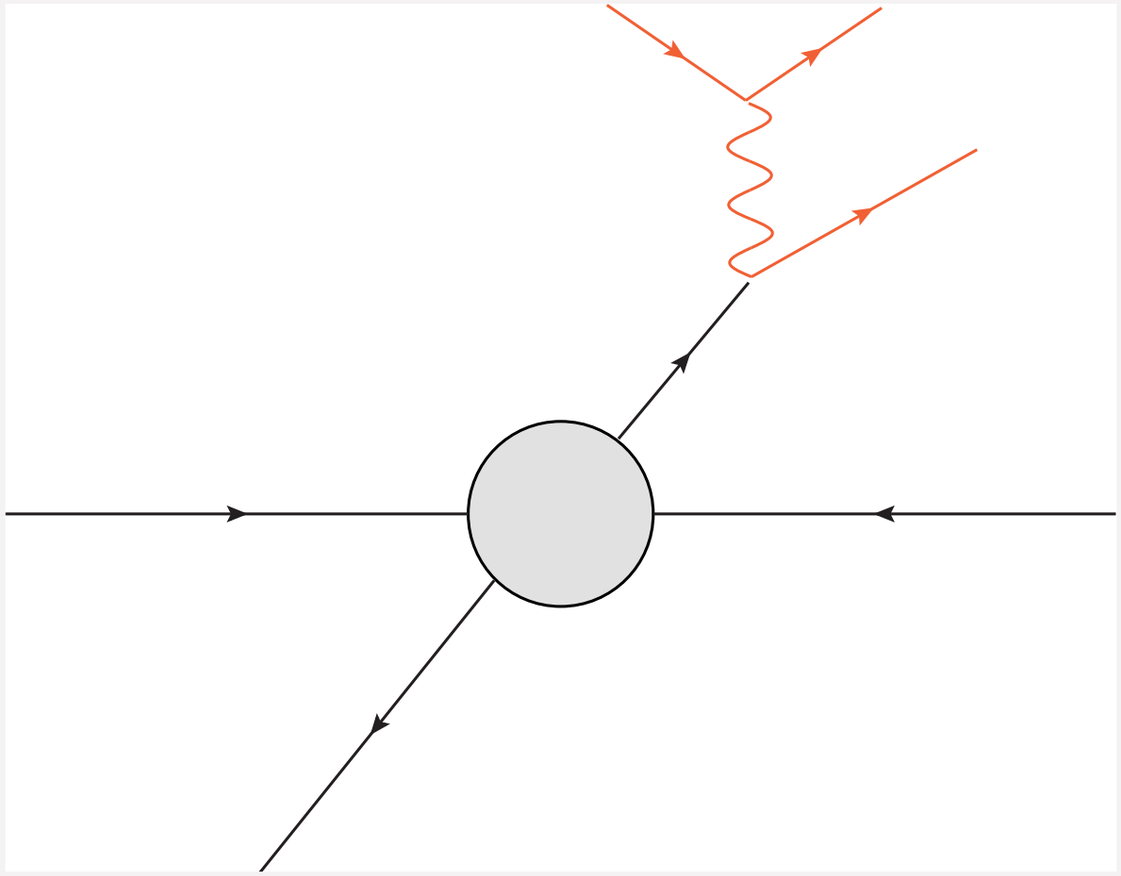


Illustration by A. Papaefstathiou, arXiv:1108.5599

Step 3: **Initial and final state radiation** («parton showers»)

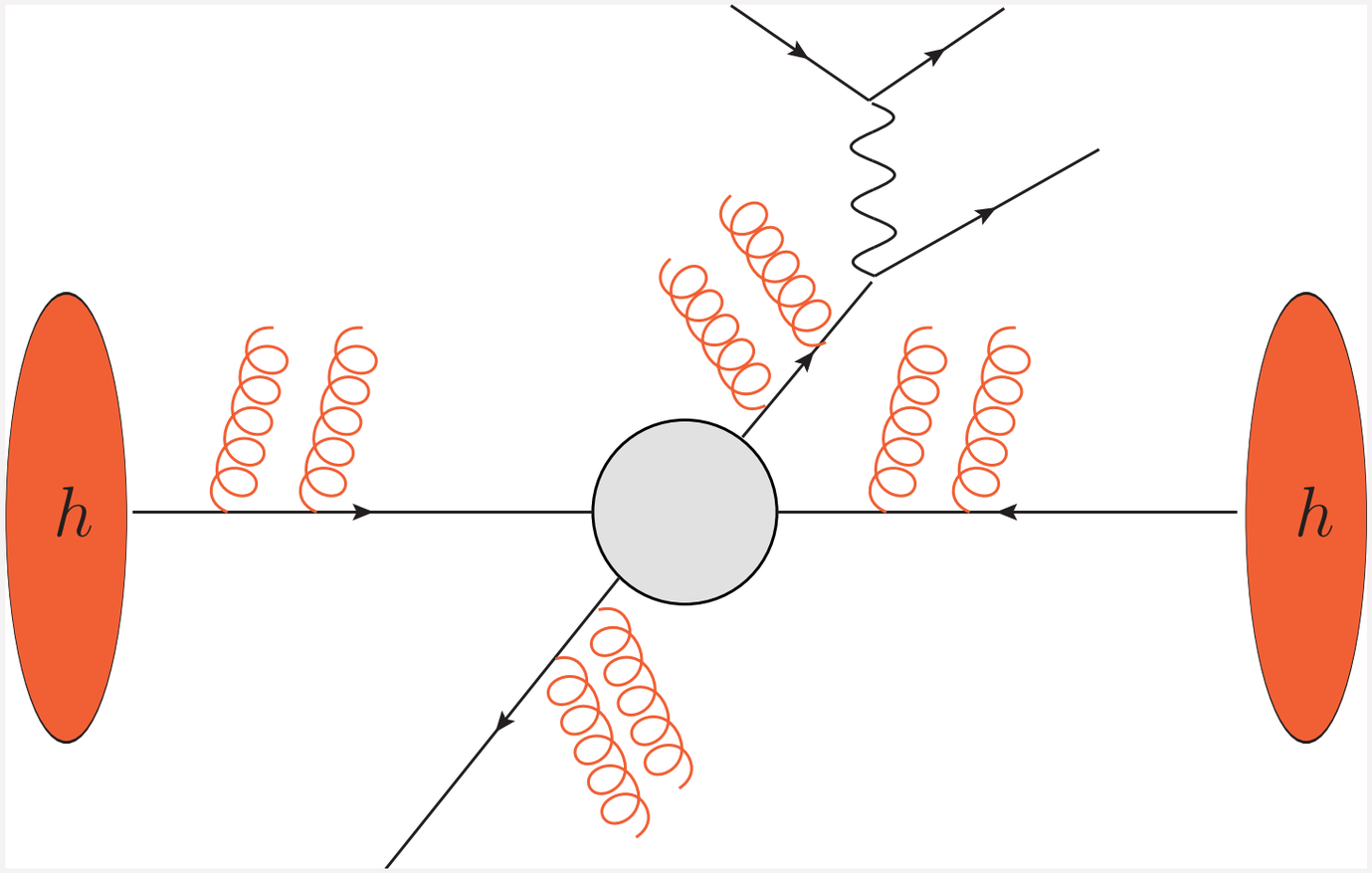


Illustration by A. Papaefstathiou, arXiv:1108.5599

Step 4: **Secondary interactions** («multi-parton interactions»)

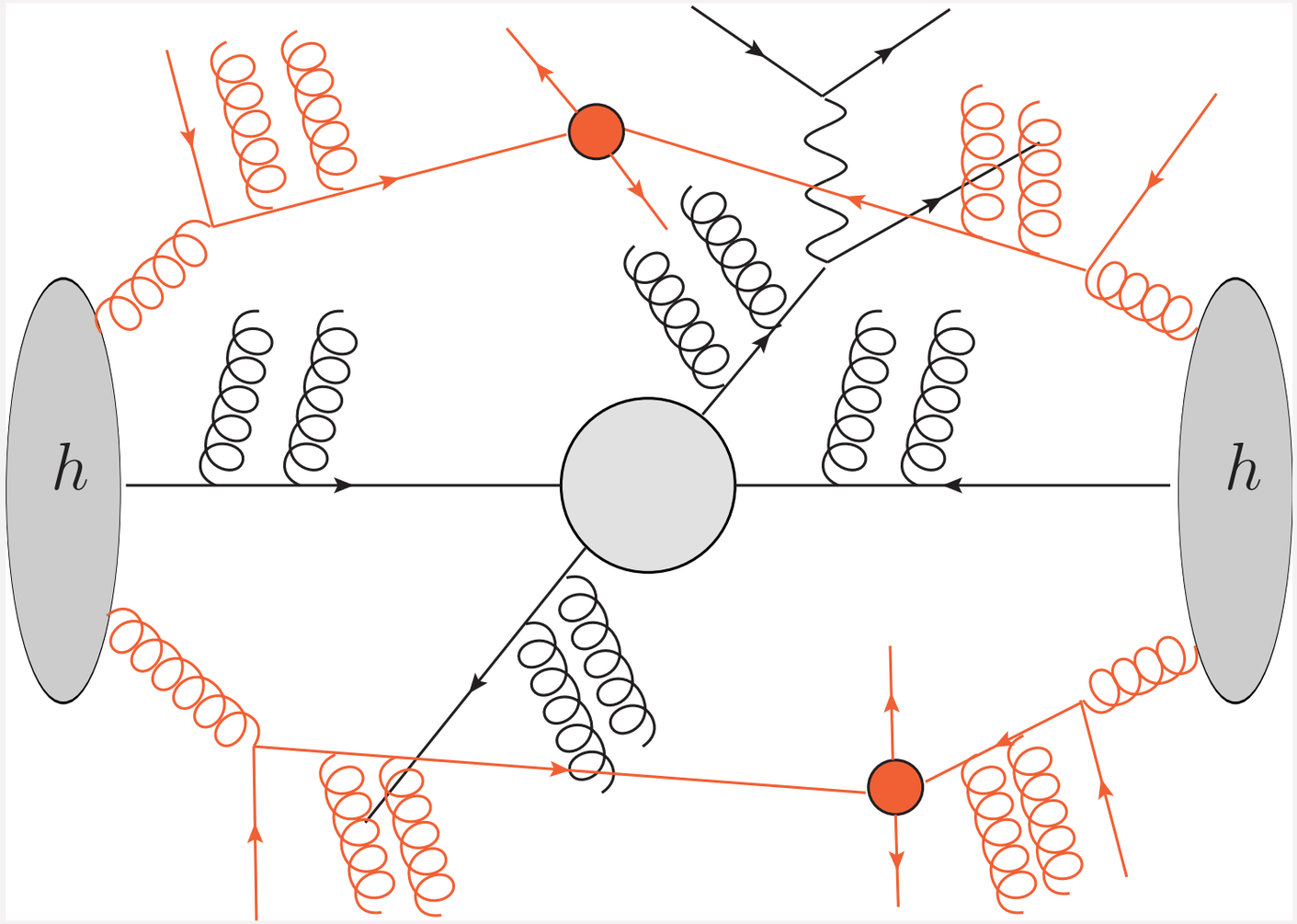


Illustration by A. Papaefstathiou, arXiv:1108.5599

Step 5: **Hadron formation and decays** (non-perturbative)

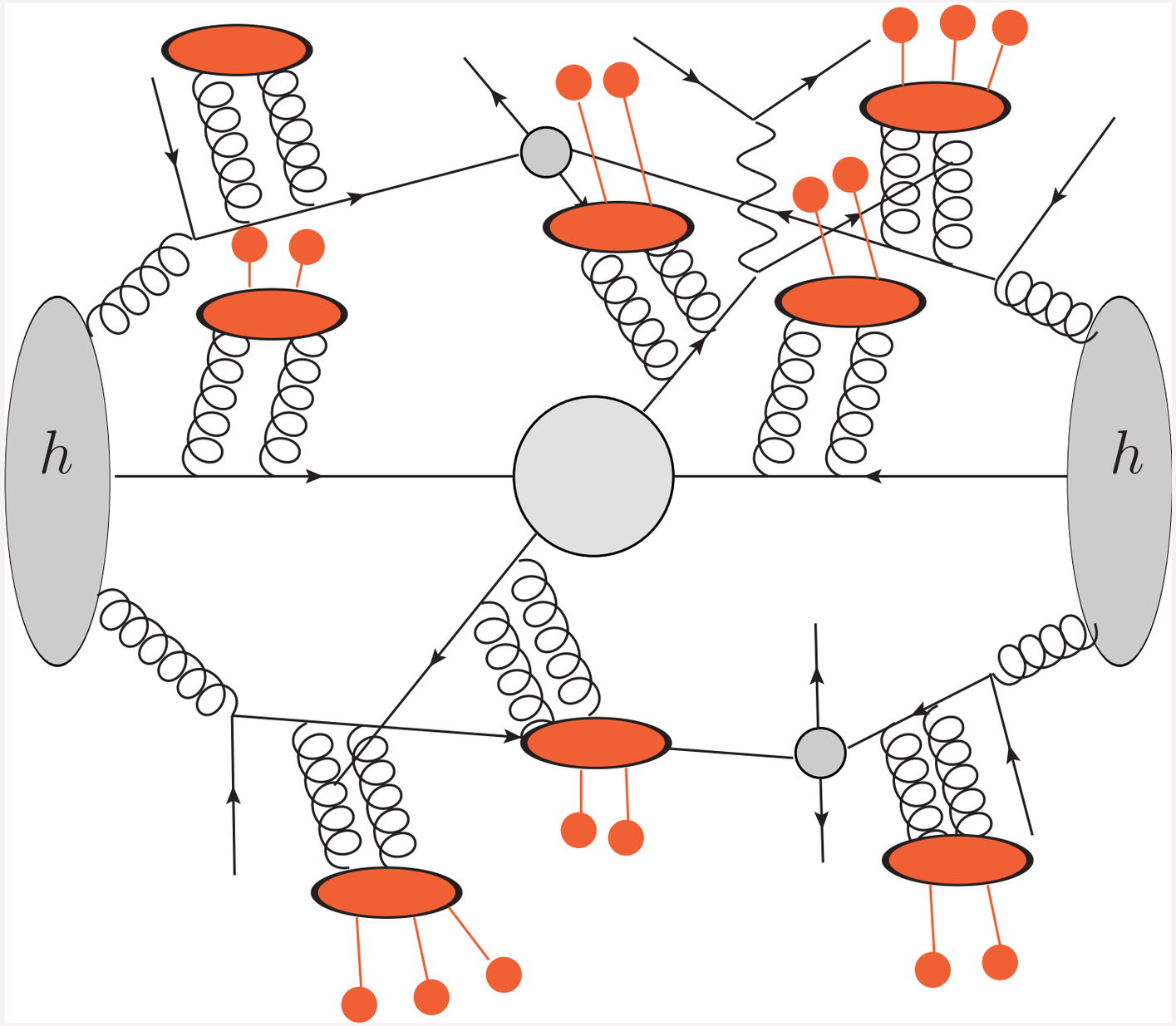
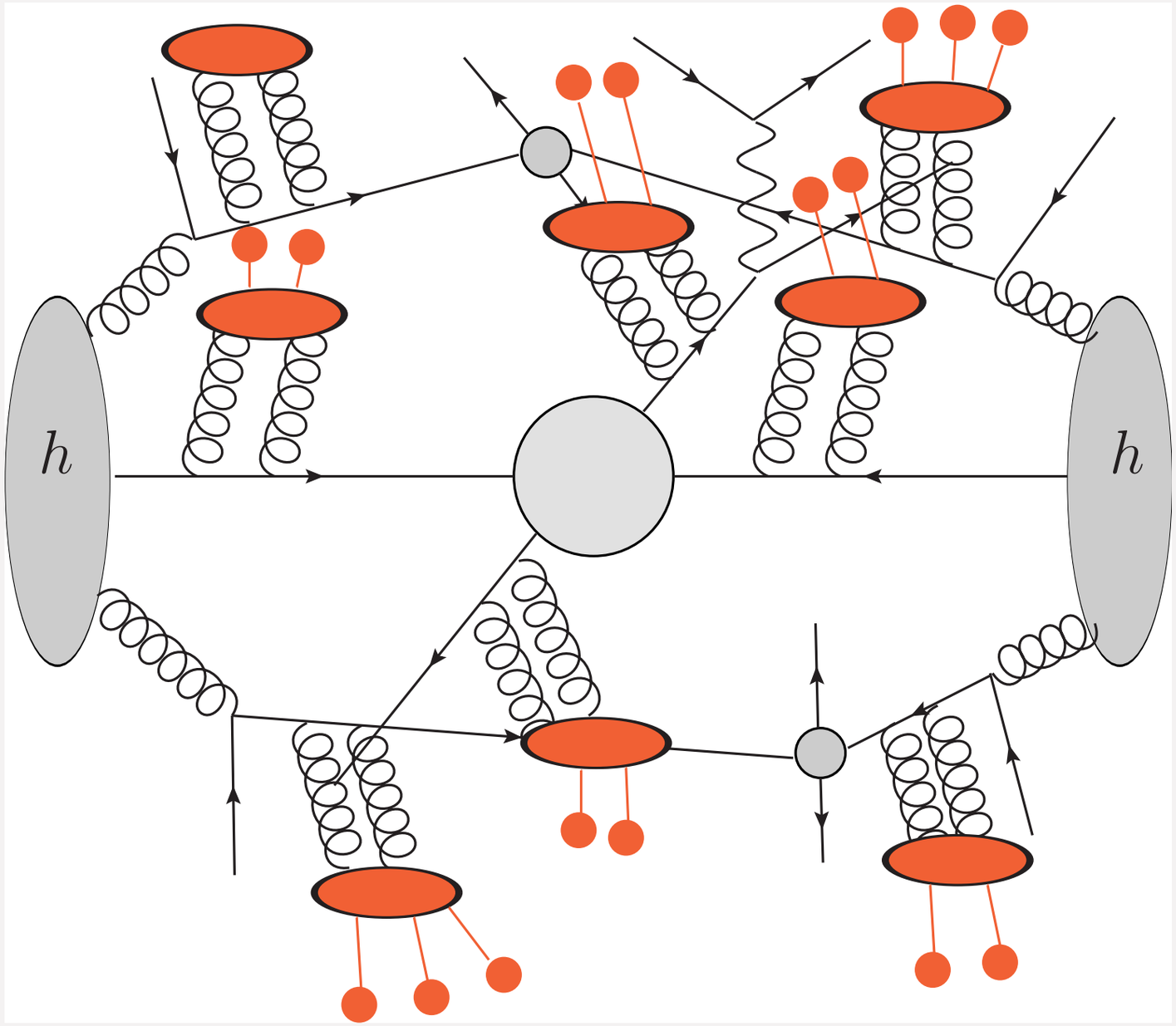


Illustration by A. Papaefstathiou, arXiv:1108.5599

Step 5: **Hadron formation and decays** (non-perturbative)

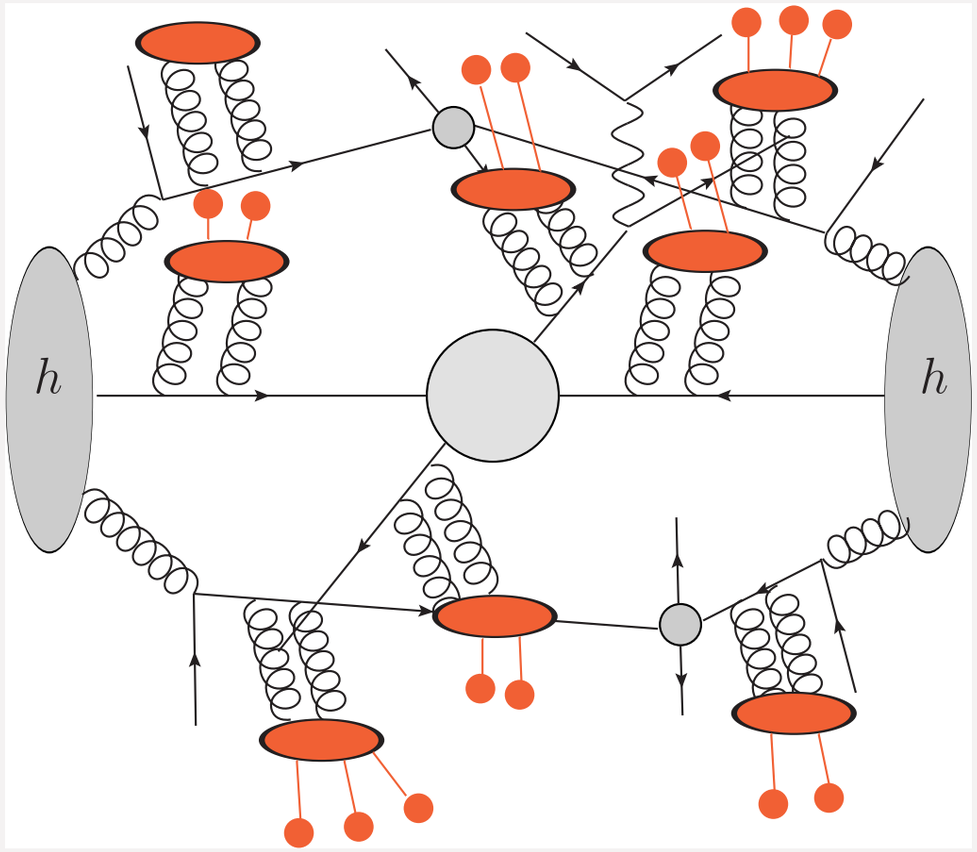


Many-dimensional phase space!

Illustration by A. Papaefstathiou, arXiv:1108.5599

Need random numbers to:

- select the hard process from all the allowed processes
- select momentum (fraction of proton momentum) for each incoming quark/gluon
- select scattering directions
- select decay process and kinematics for each unstable particle
- pick a branching pattern for initial and final state radiation (distributes the energy across more partons)
- hadron formation and decay kinematics
- particle-detector interactions
- ...

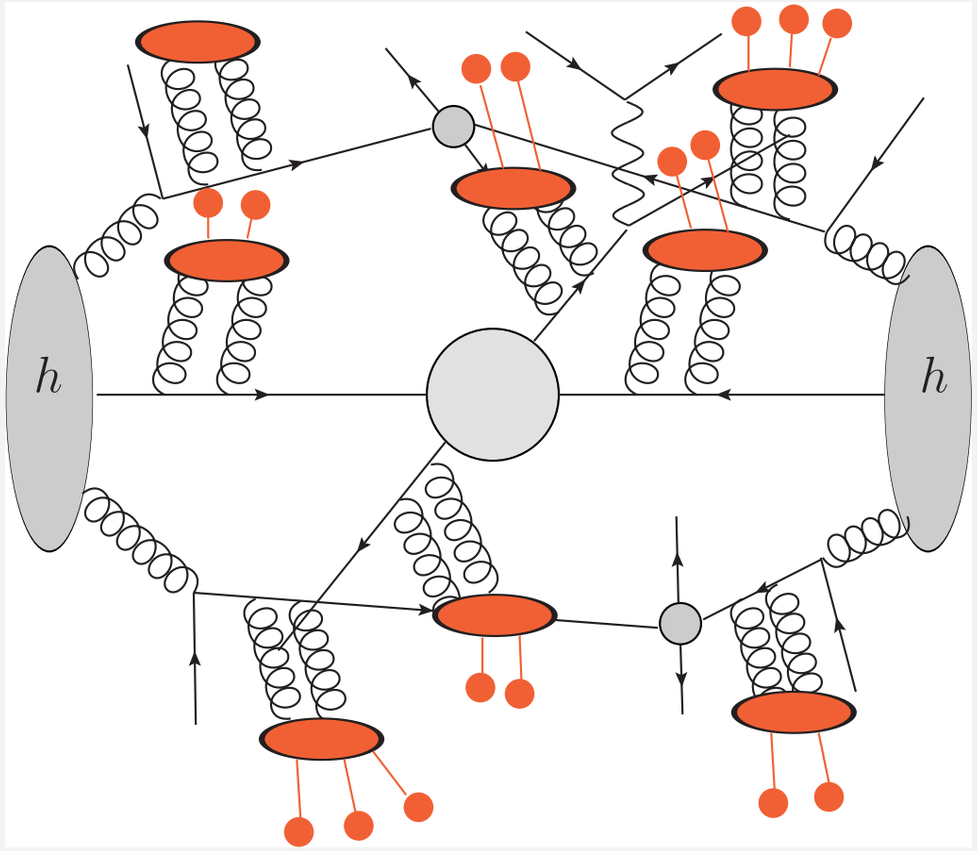


$$p(\text{obs}) = \int p(\text{obs}, \text{unobs})\, d(\text{unobs})$$

$$= \int p(\text{obs}|\text{unobs})\, p(\text{unobs})\, d(\text{unobs})$$

Need random numbers to:

- select the hard process from all the allowed processes
- select momentum (fraction of proton momentum) for each incoming quark/gluon
- **select scattering directions**
- select decay process and kinematics for each unstable particle
- pick a branching pattern for initial and final state radiation (distributes the energy across more partons)
- hadron formation and decay kinematics
- **particle-detector interactions**
- …



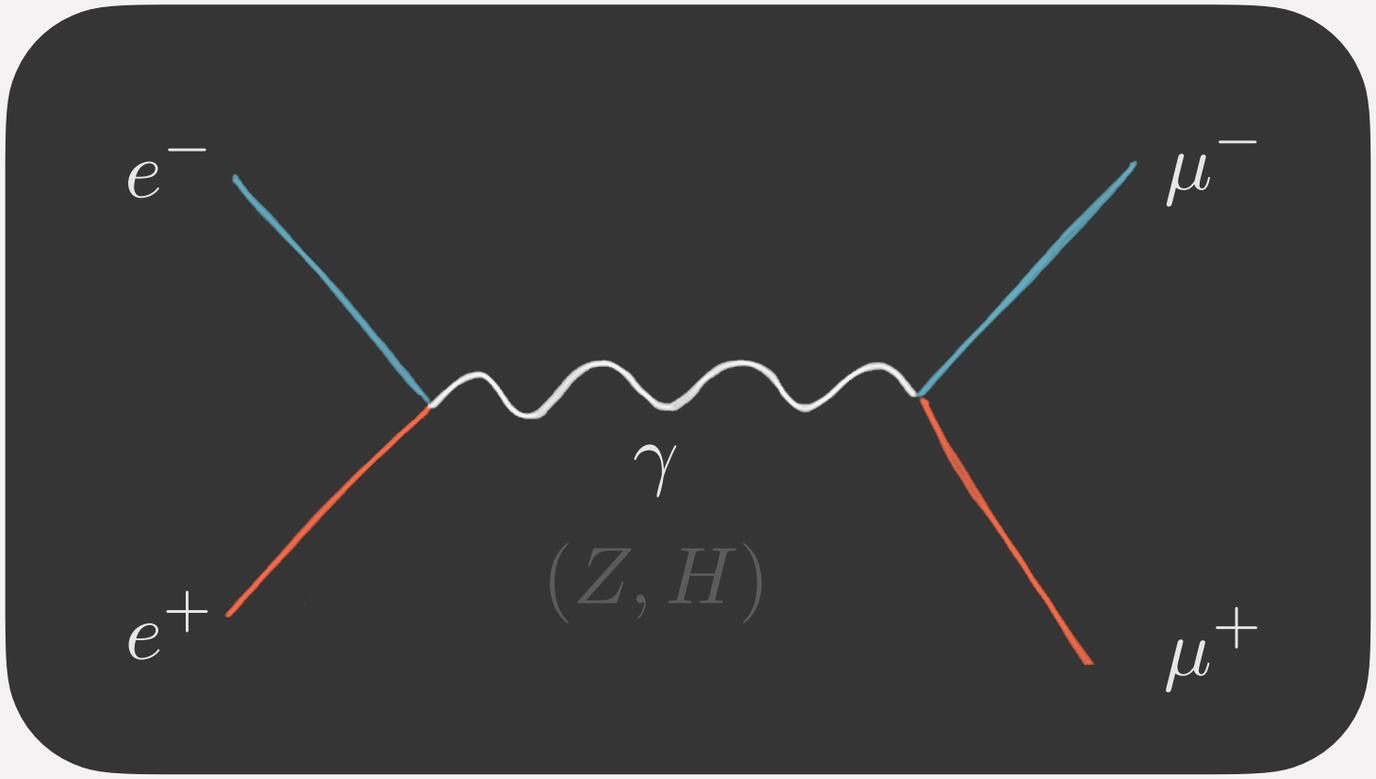
$$p(\text{obs}) = \int p(\text{obs}, \text{unobs})\, d(\text{unobs})$$

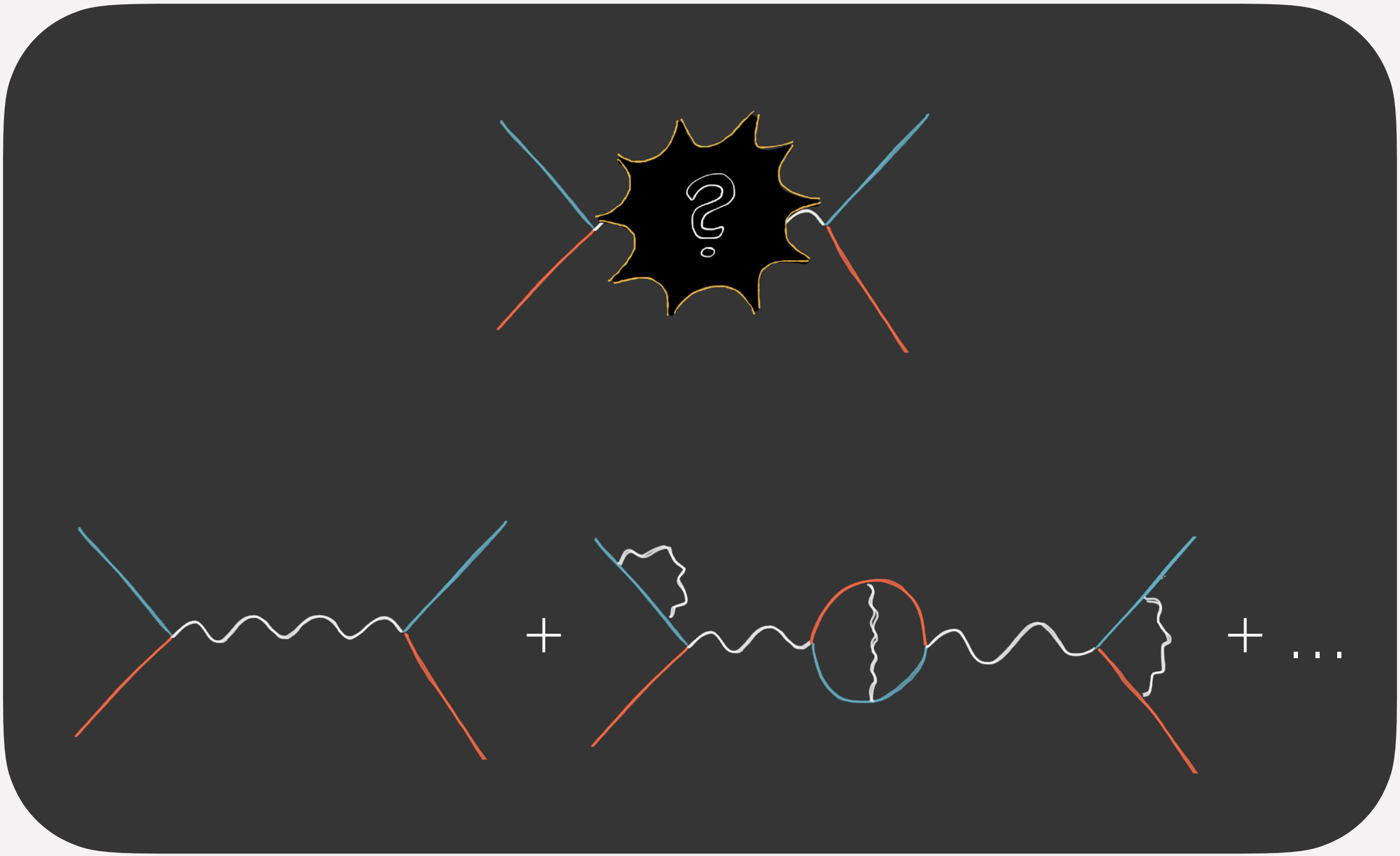
$$= \int p(\text{obs}|\text{unobs})\, p(\text{unobs})\, d(\text{unobs})$$

4. Example: where did my muons go?

The simplest process in particle physics?

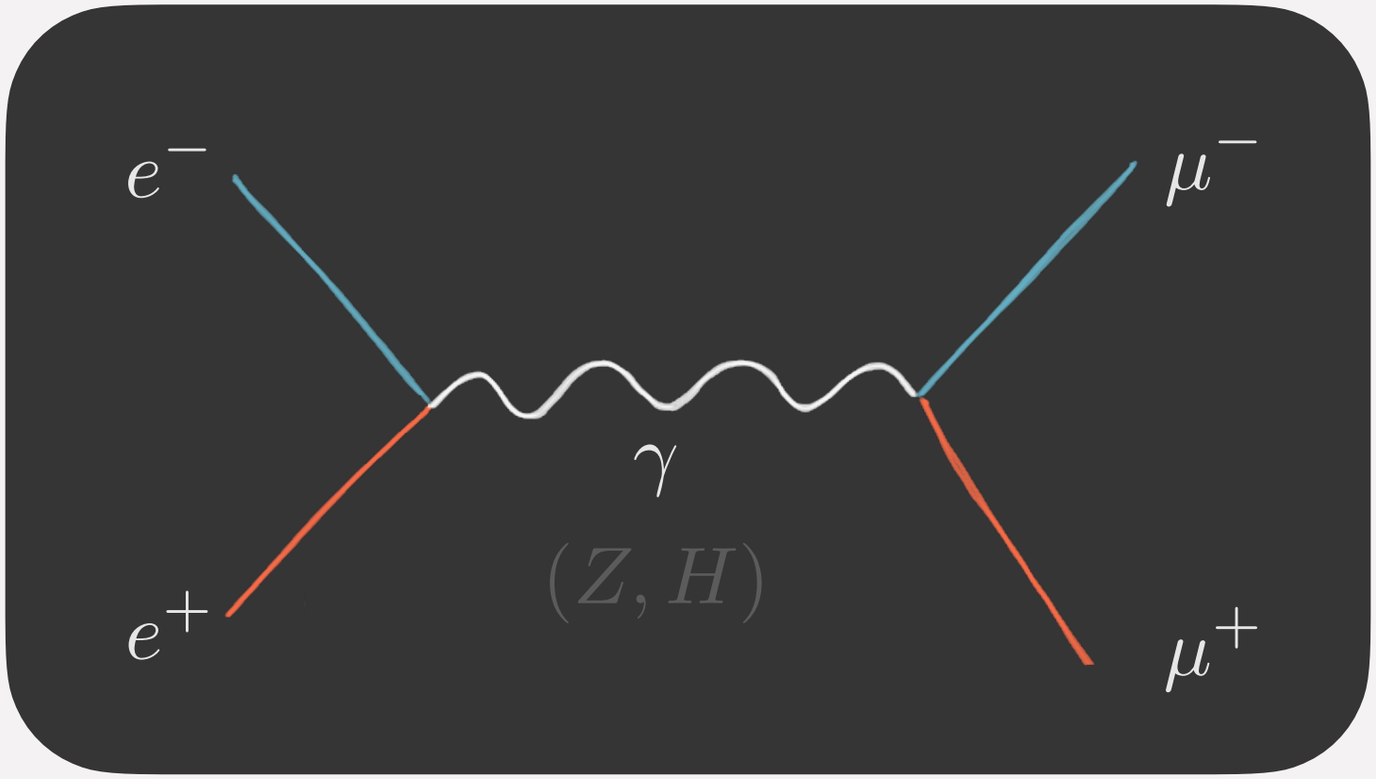
- Electron-positron pair annihilates and creates a muon-antimuon pair
- Only consider the process with a photon propagator (can also have Z and H)
- Only work at lowest order in perturbation theory





The simplest process in particle physics?

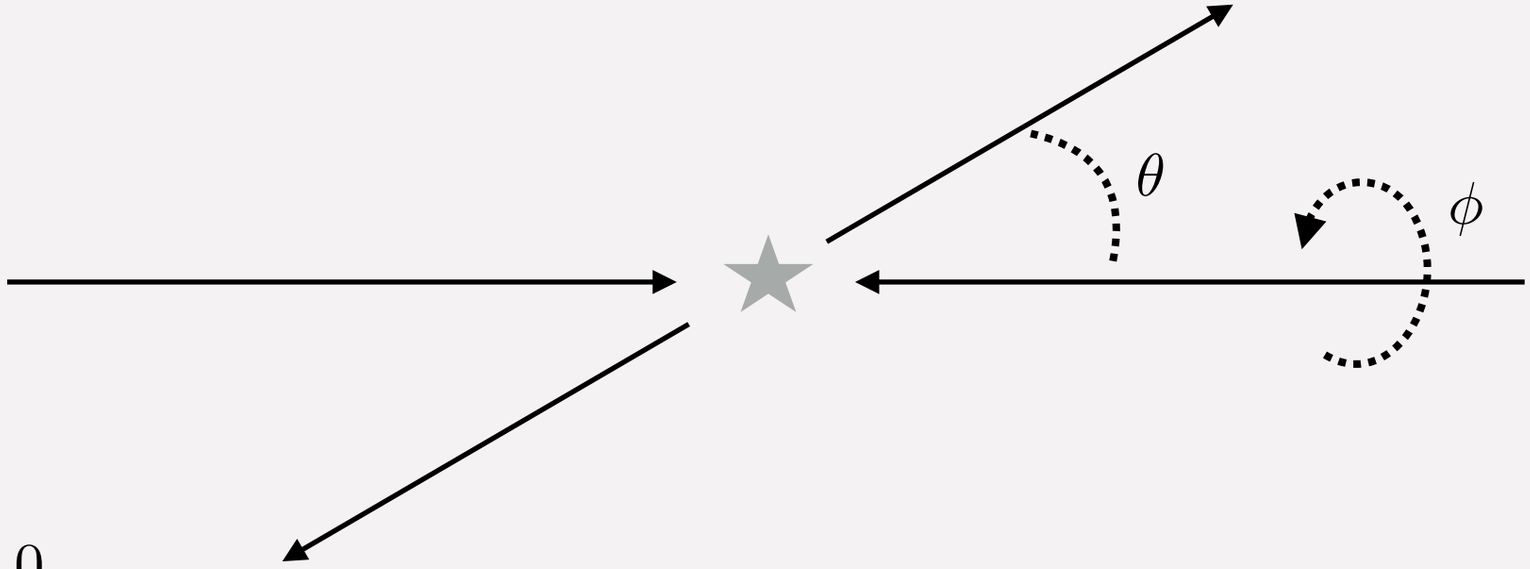
- Electron-positron pair annihilates and creates a muon-antimuon pair
- Only consider the process with a photon propagator (can also have Z and H)
- Only work at lowest order in perturbation theory
- **1) Generate the hard process 2) Simple detector simulation**



Generating the hard process

- 2 —> 2 process: the kinematics is fully determined by the direction of one outgoing muon
- QFT (QED) gives us the correct probability distribution for this direction
- Draw samples from this distribution, calculate all energies and momenta

$$\cos\theta \sim (1 + \cos^2\theta) \qquad \phi \sim U(0, 2\pi)$$



$$m \approx 0$$

$$E = E_{com}/2 \qquad p_x = E\sin\theta\cos\phi \qquad p_y = E\sin\theta\sin\phi \qquad p_z = E\cos\theta$$

Simple detector smearing:

- The outgoing particle has some «true» energy
- Our detector has some uncertainty — we will never measure exactly the true energy
- Simple model: the observed energy is drawn from a Gaussian centred on the true value

$$
\begin{aligned}
p(E_{\text{obs}}) &= \int p(E_{\text{obs}}, E_{\text{true}})\, dE_{\text{true}} \\
&= \int p(E_{\text{obs}}|E_{\text{true}})\, p(E_{\text{true}})\, dE_{\text{true}} \\
&= \int \mathcal{N}(E_{\text{obs}}; E_{\text{true}}, \sigma_E)\, p(E_{\text{true}})\, dE_{\text{true}}
\end{aligned}
$$

Anders Kvellestad

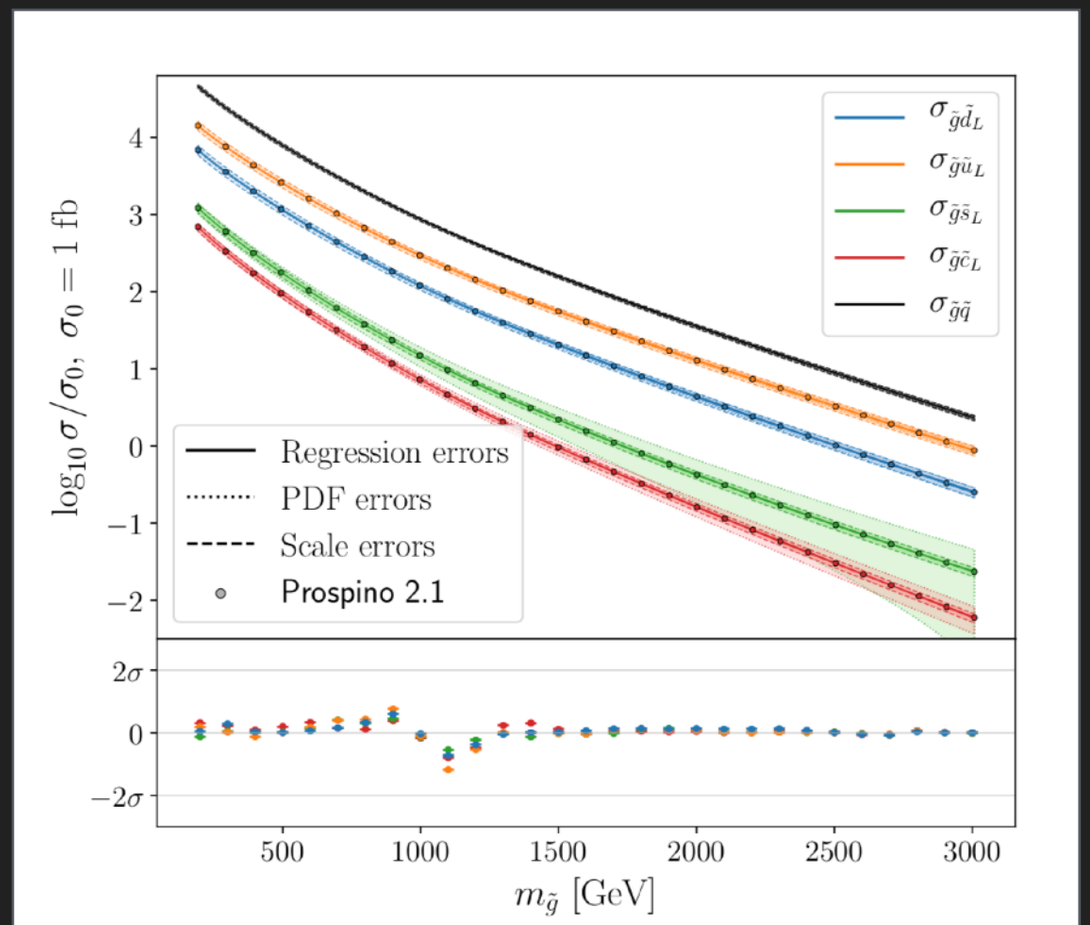
Anders Kvellestad

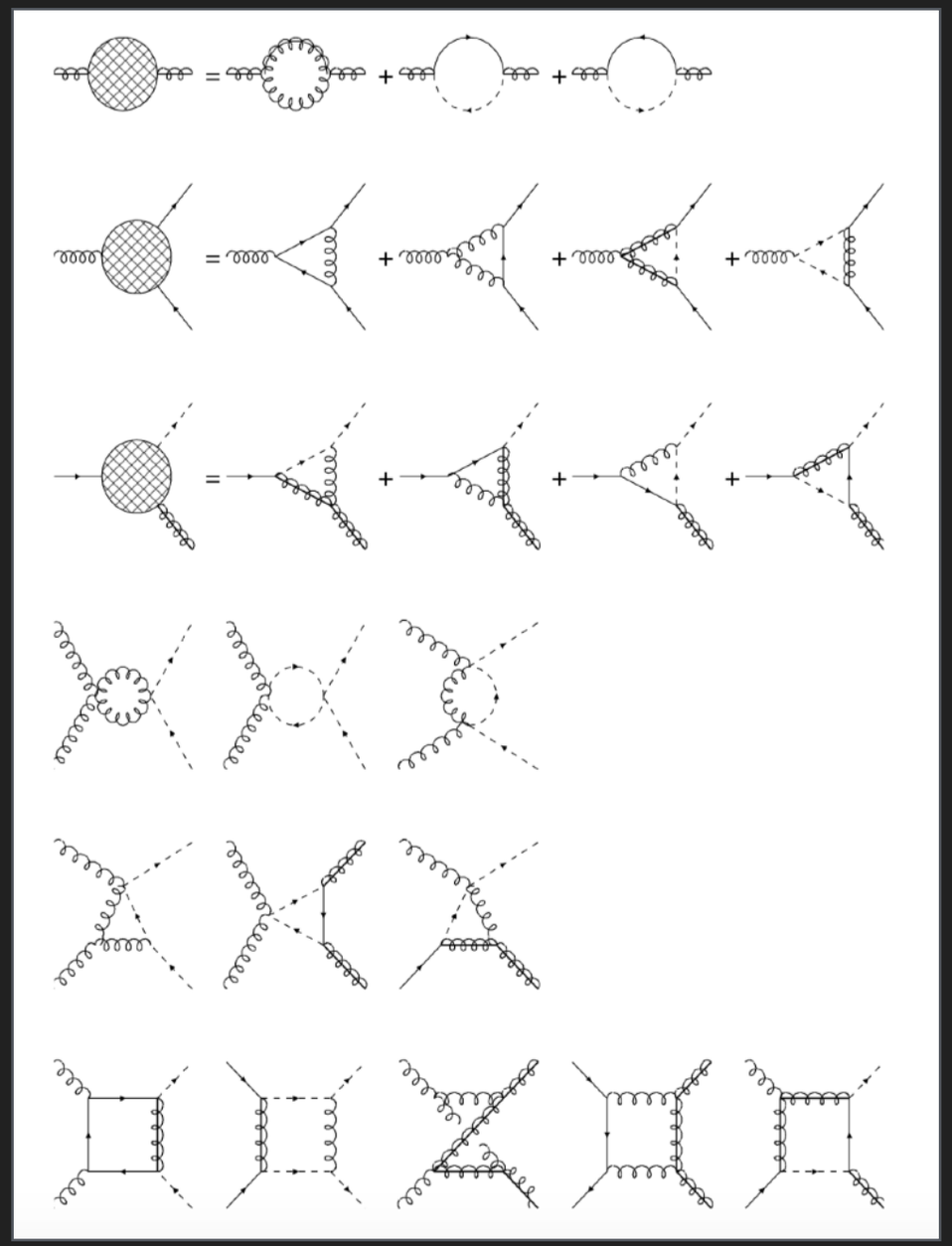
3. Rev. Bayes to the rescue!

(Gaussian process regression)

One of the most CPU-expensive theory predictions:

High-accuracy production cross-sections (interaction probabilities) for LHC predictions

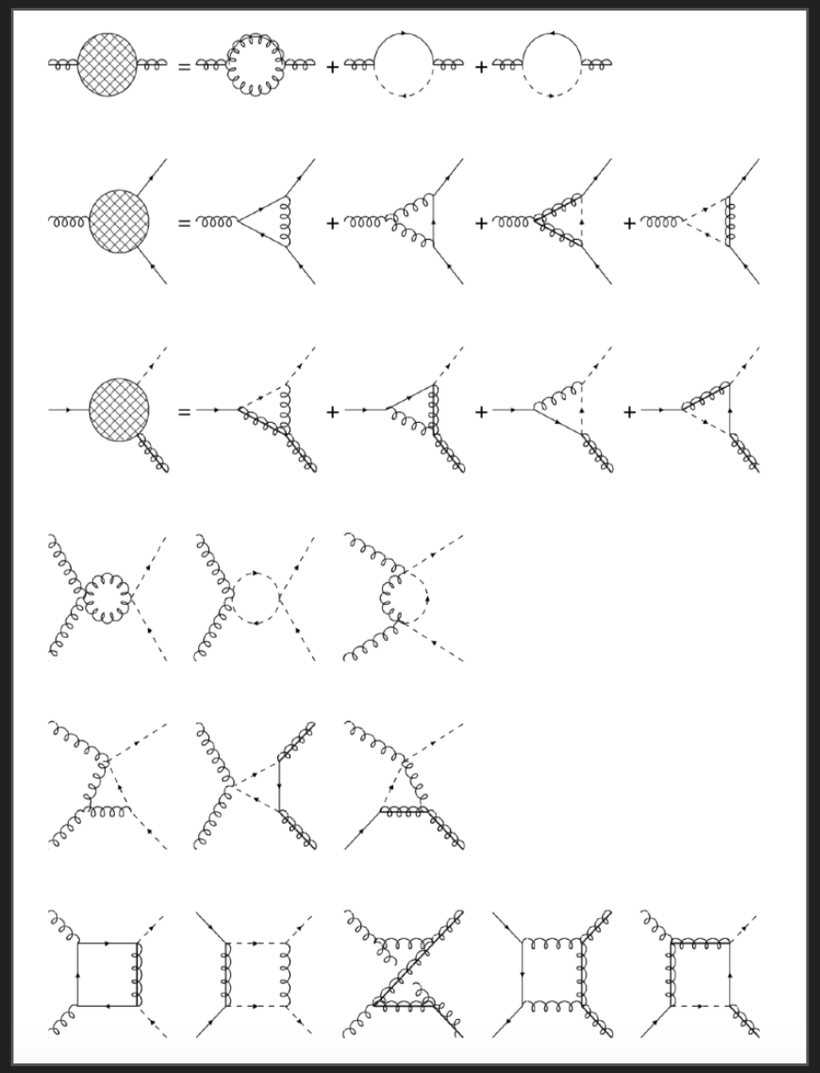




Evaluation time for a single parameter point:
~minutes/hours…

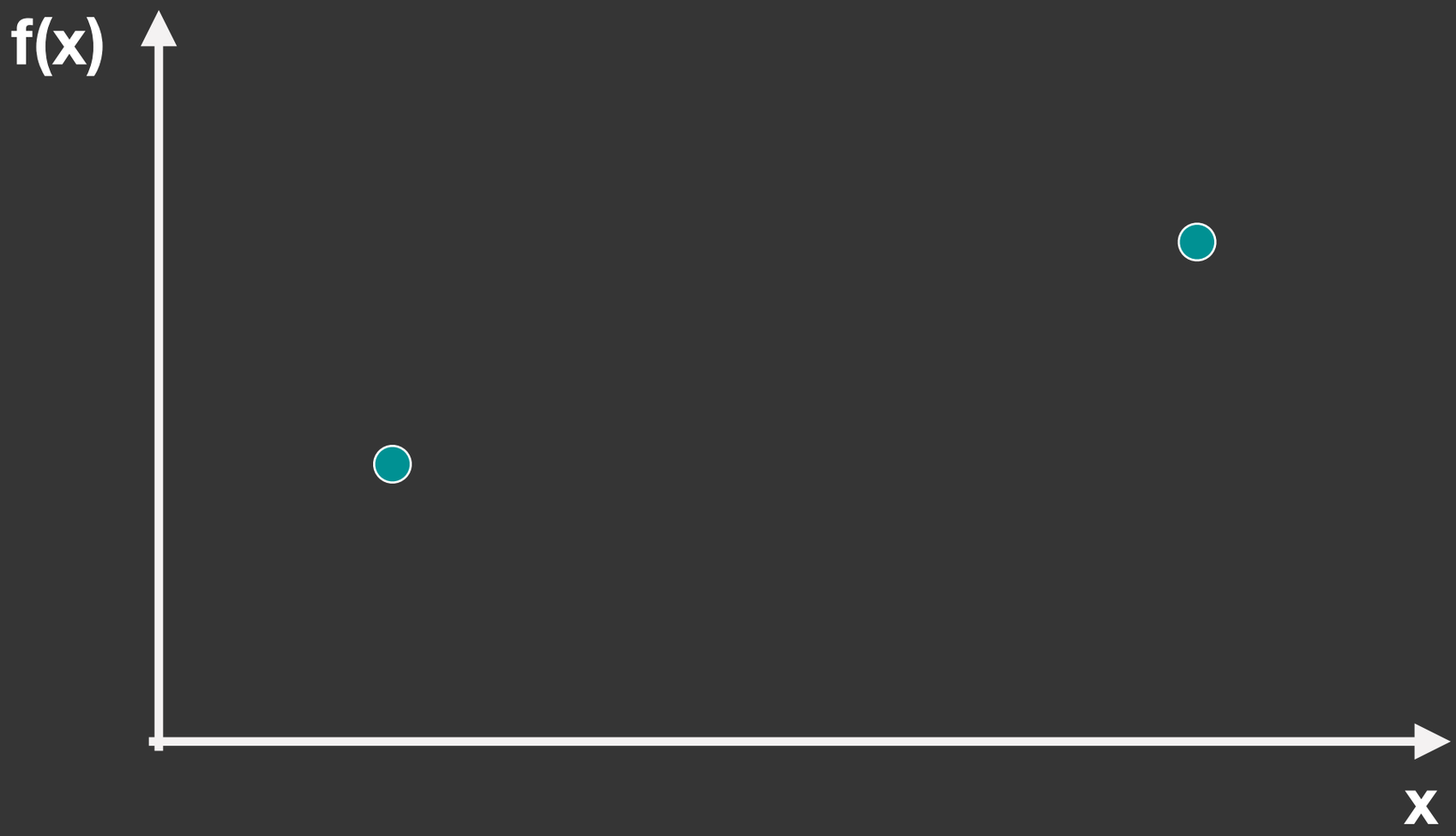
One of the most CPU-expensive theory predictions:

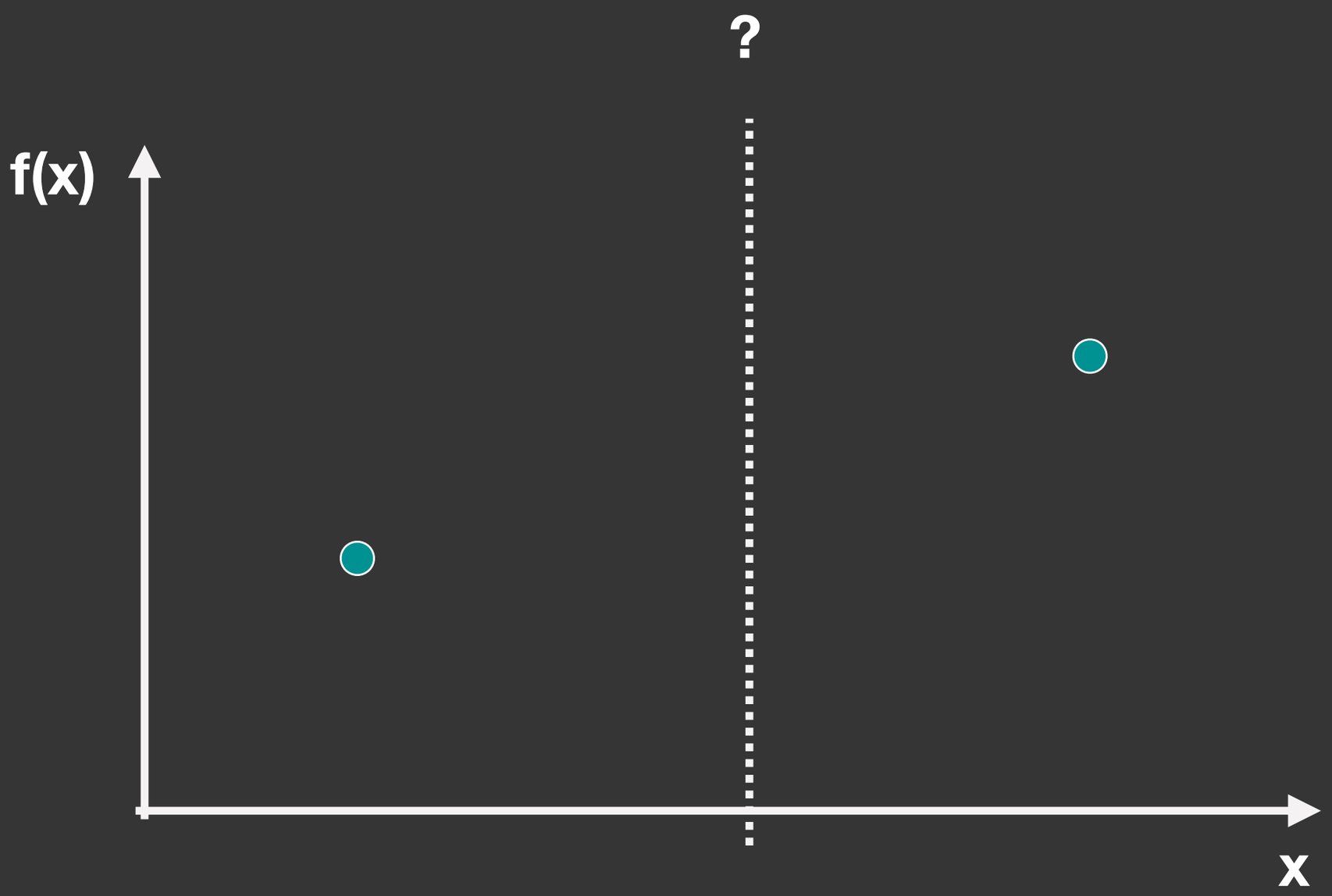
High-accuracy production cross-sections (interaction probabilities) for LHC predictions

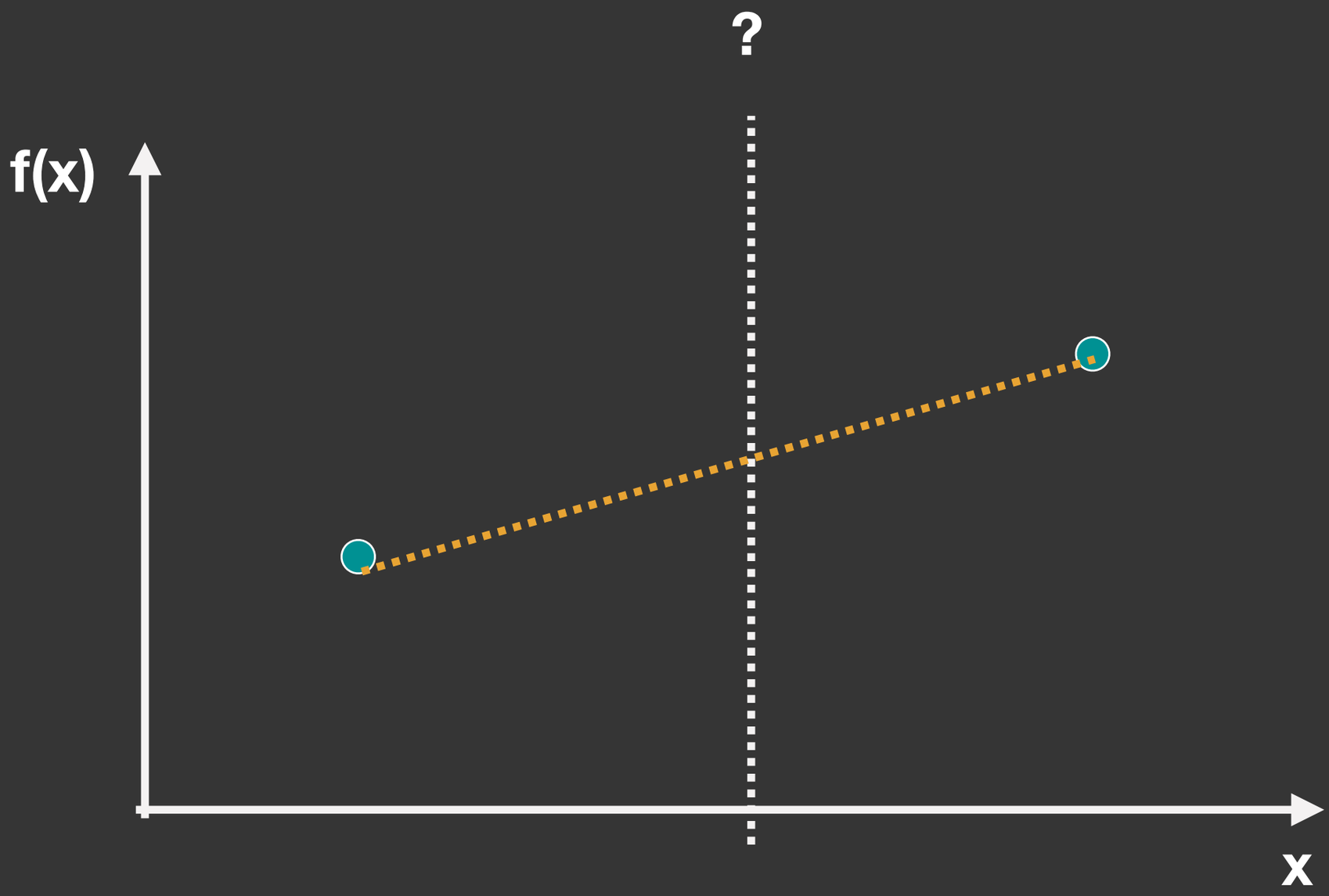


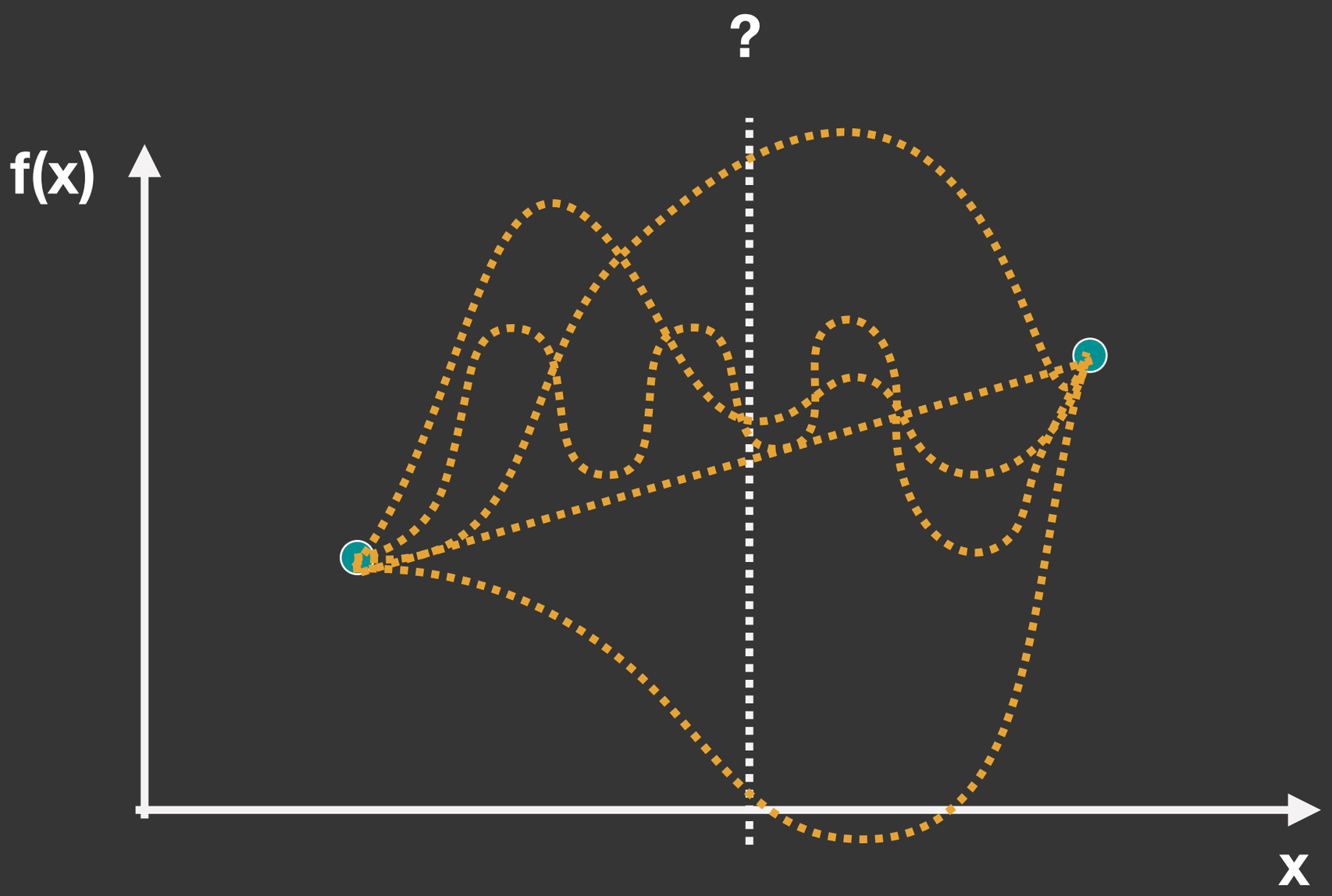
Evaluation time for a single parameter point:
~minutes/hours…

Let’s see if we can speed things up with some smart regression/interpolation…

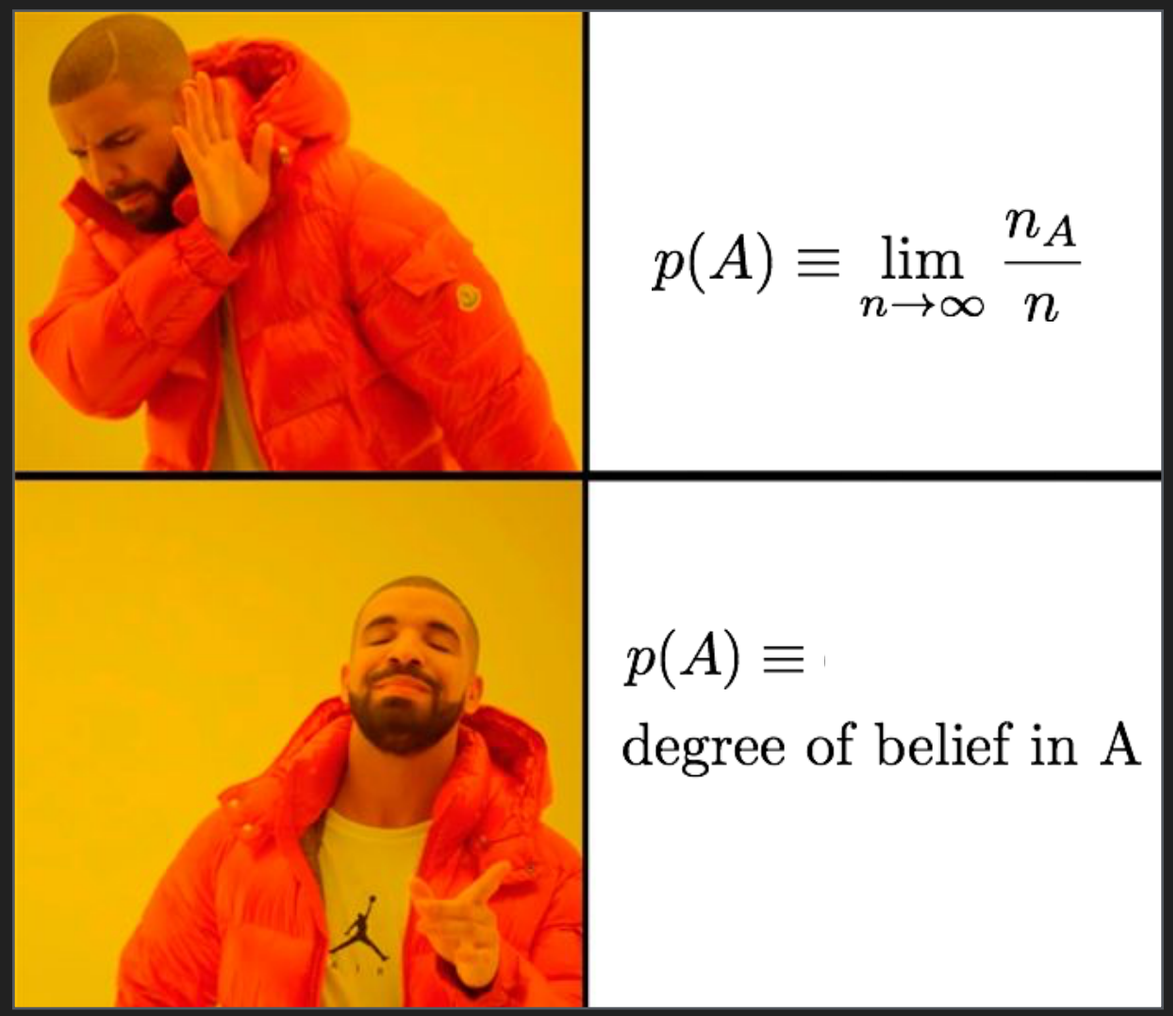

f(x)
x


?
f(x)
x

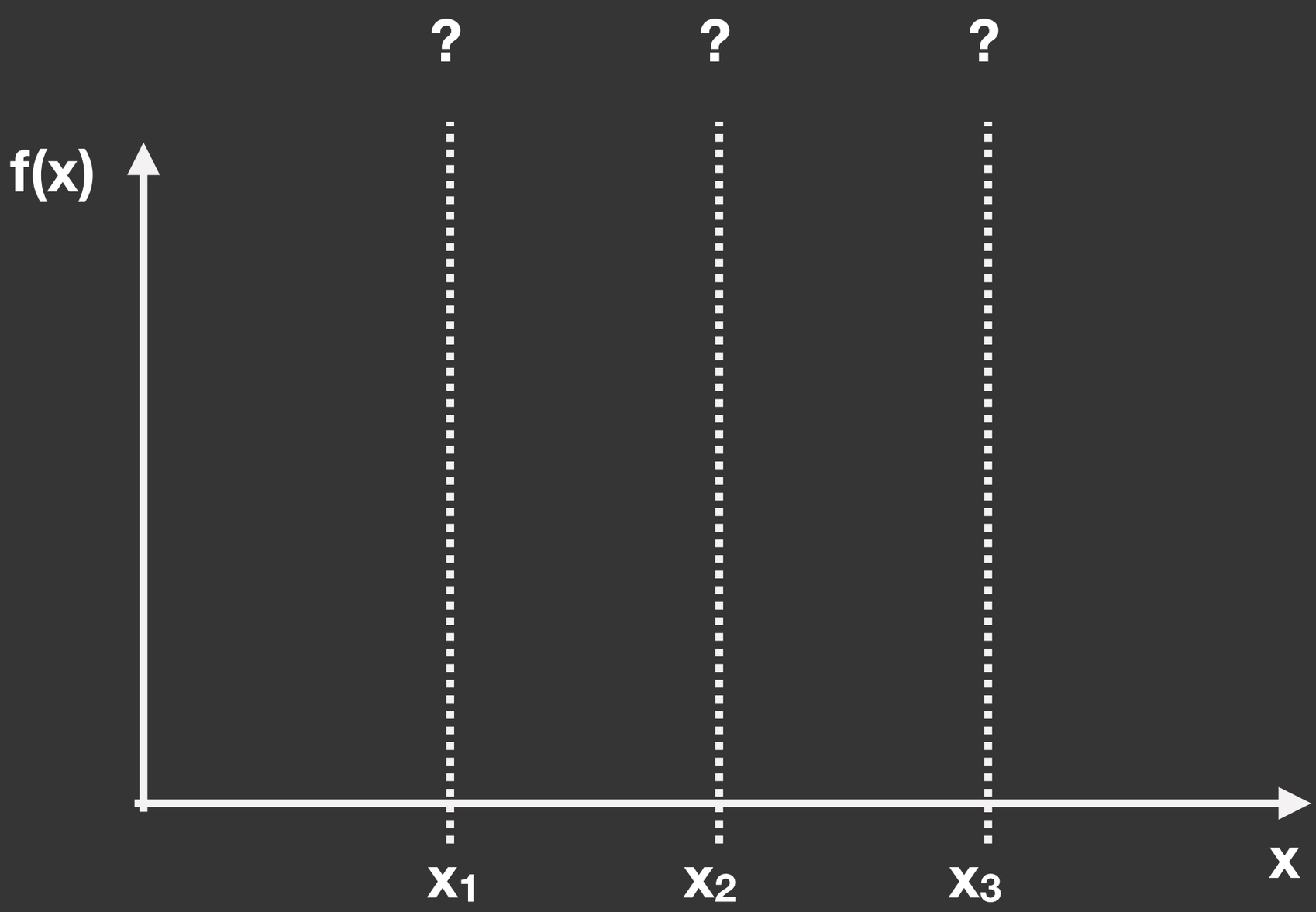

?
f(x)
x


?
f(x)
x

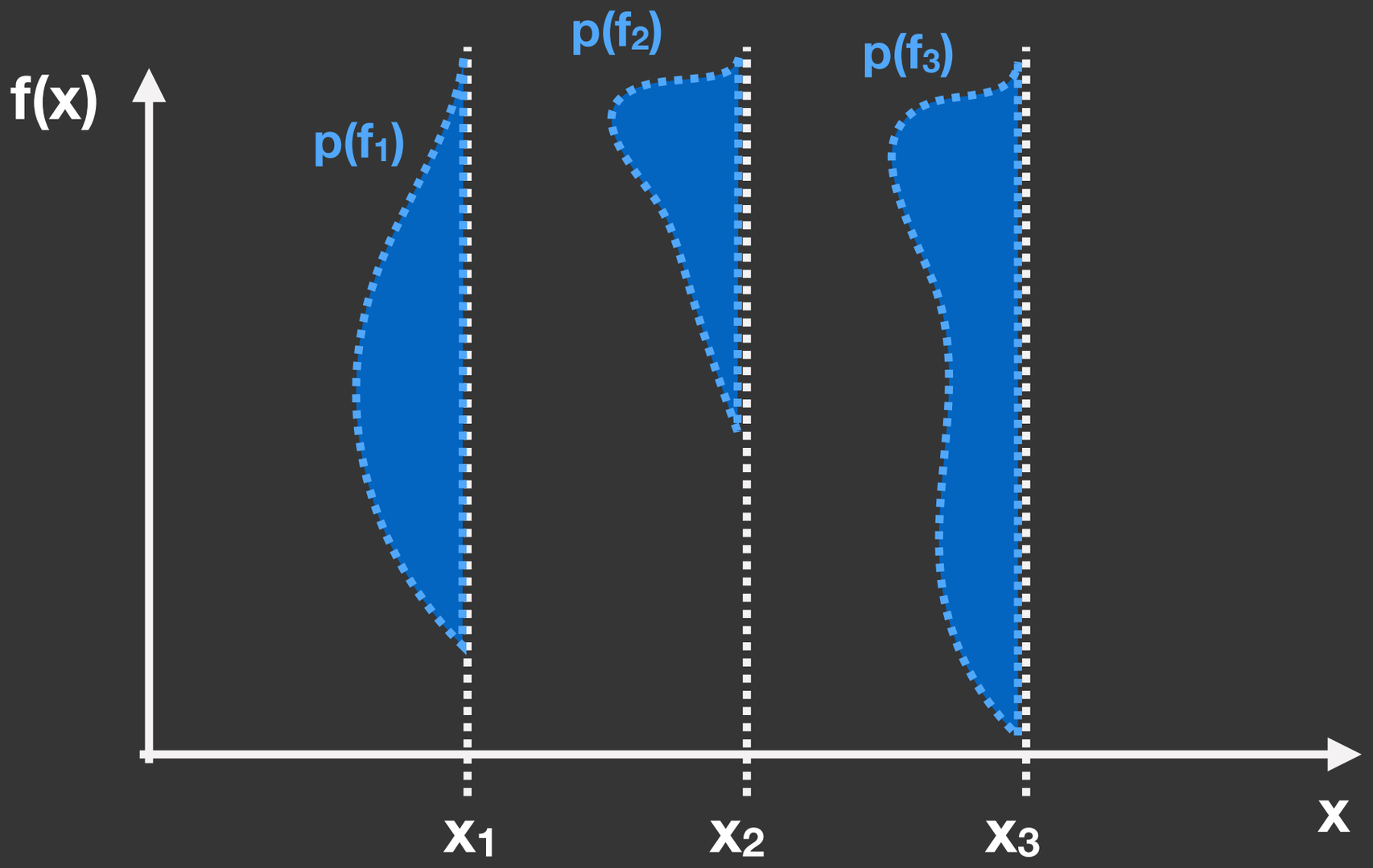
- Need to make an educated guess for the unknown **f** at any given **x**
- Being sensible **Bayesians**, we know we should **quantify our beliefs about f** using **probabilities**



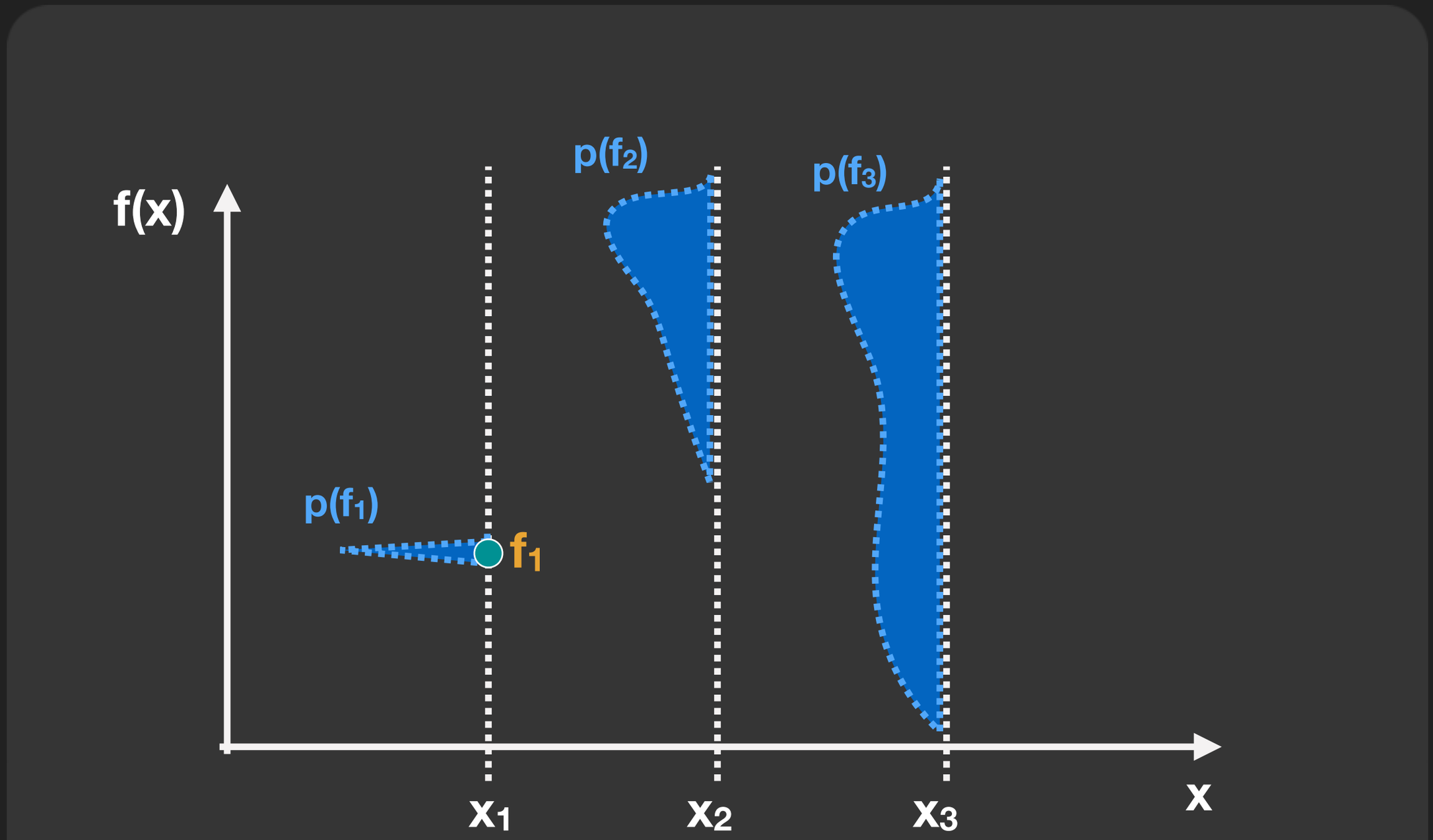
Consider three unknown function values: f_1, f_2, f_3



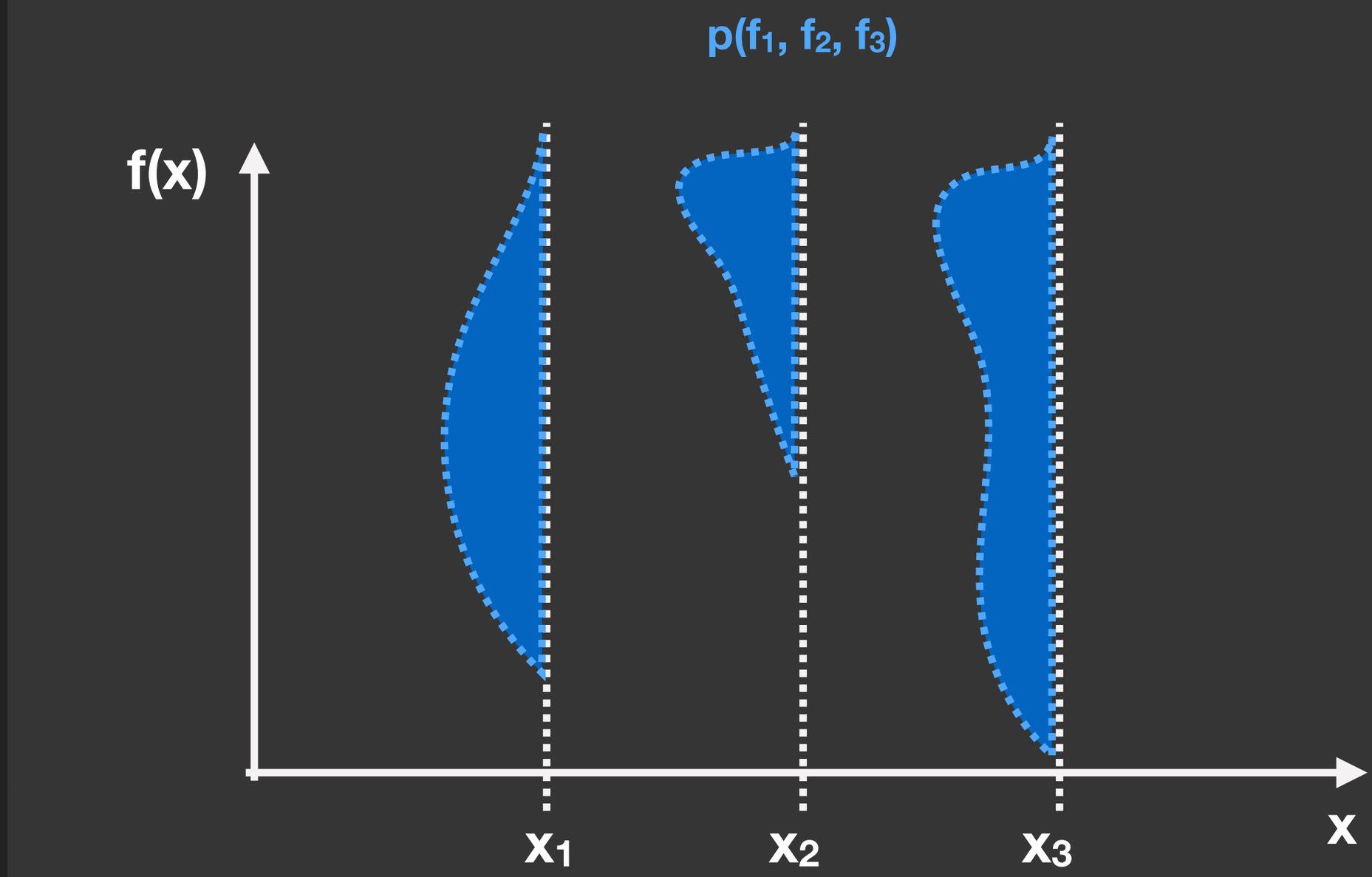
Could formulate three independent prior beliefs…



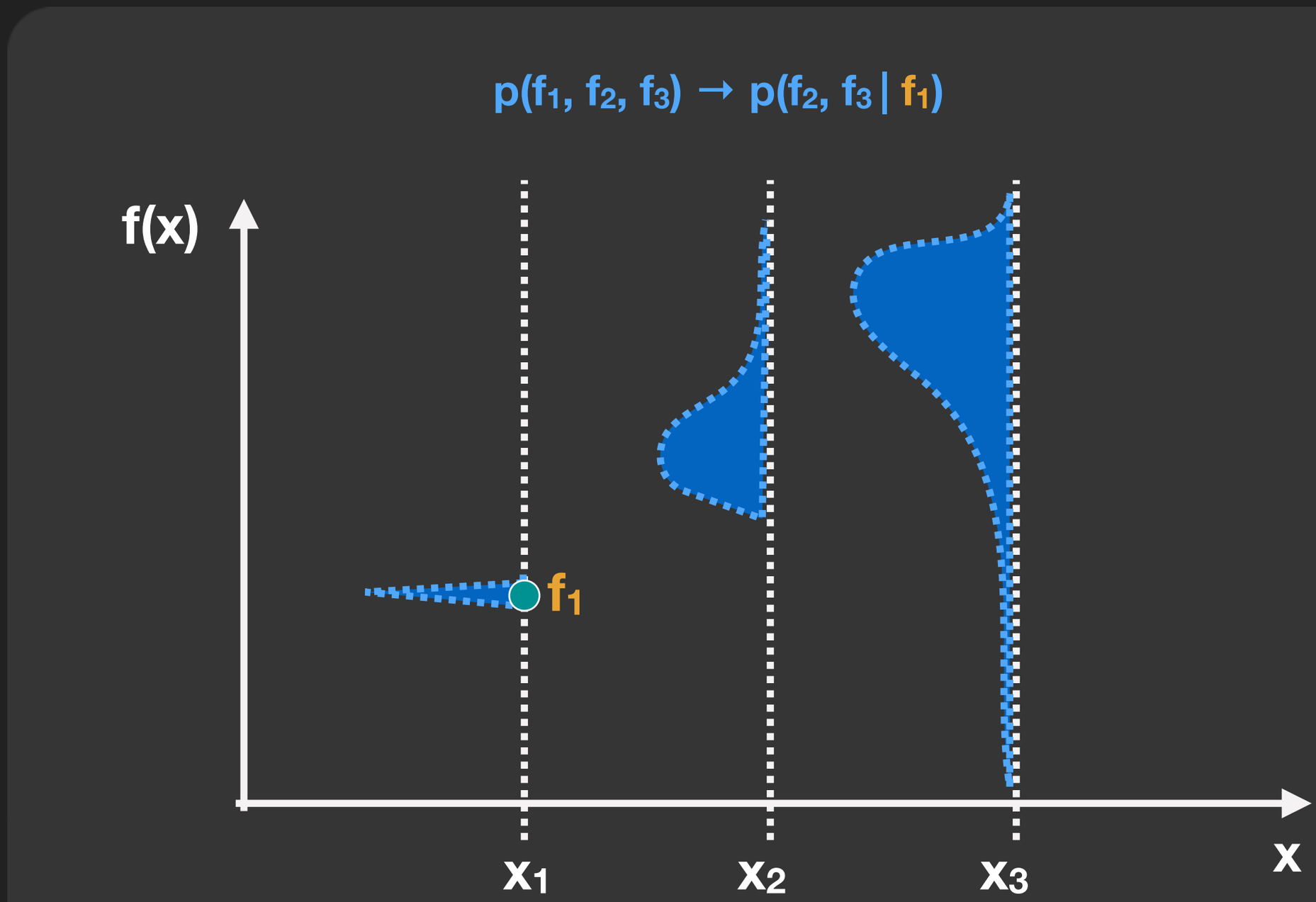
…but then learning what $\mathbf{f_1}$ is won't tell us anything about $\mathbf{f_2}$ or $\mathbf{f_3}$



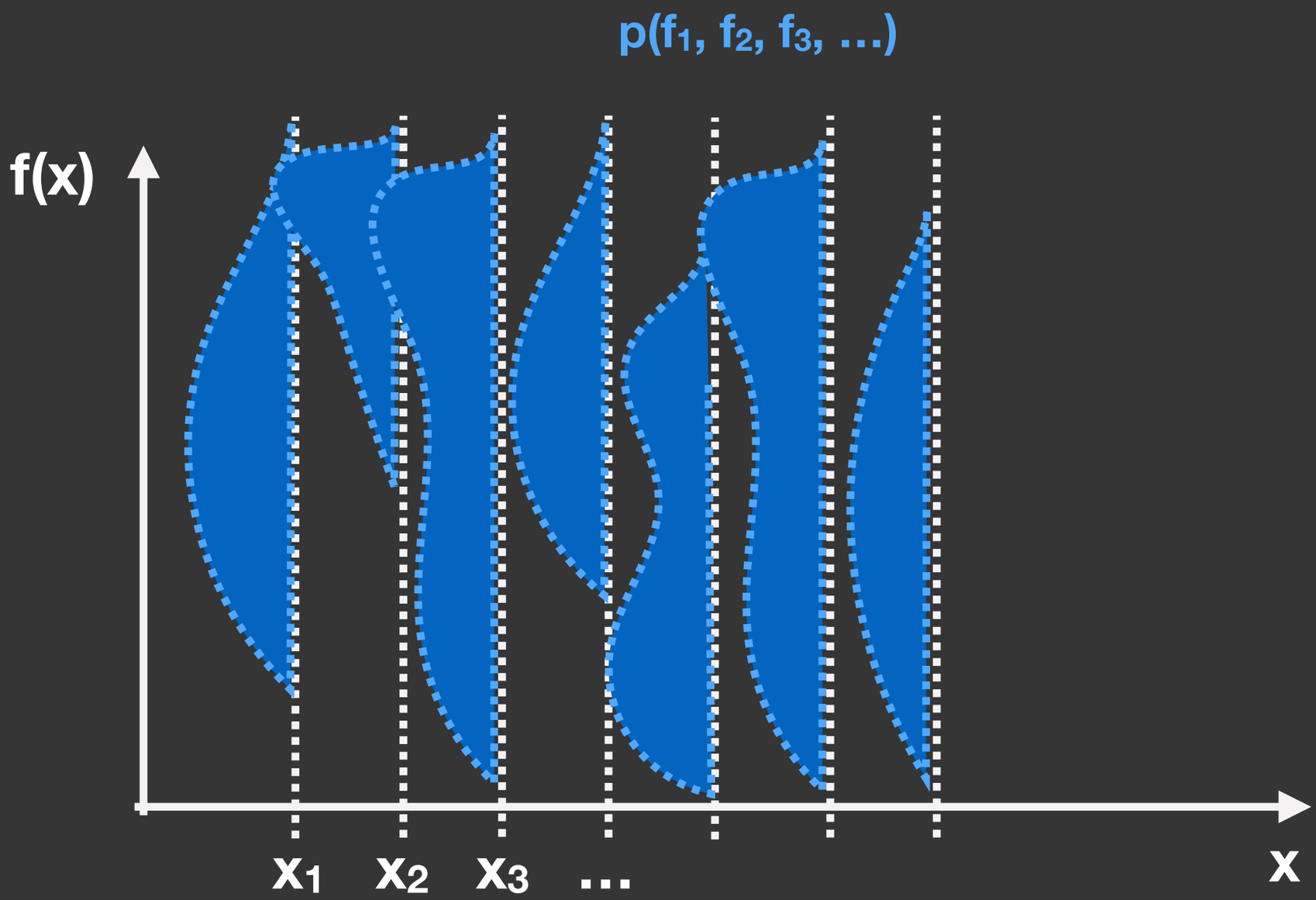
Need a **joint prior** → includes our belief about how $\mathbf{f_1}$, $\mathbf{f_2}$ and $\mathbf{f_3}$ are **correlated**



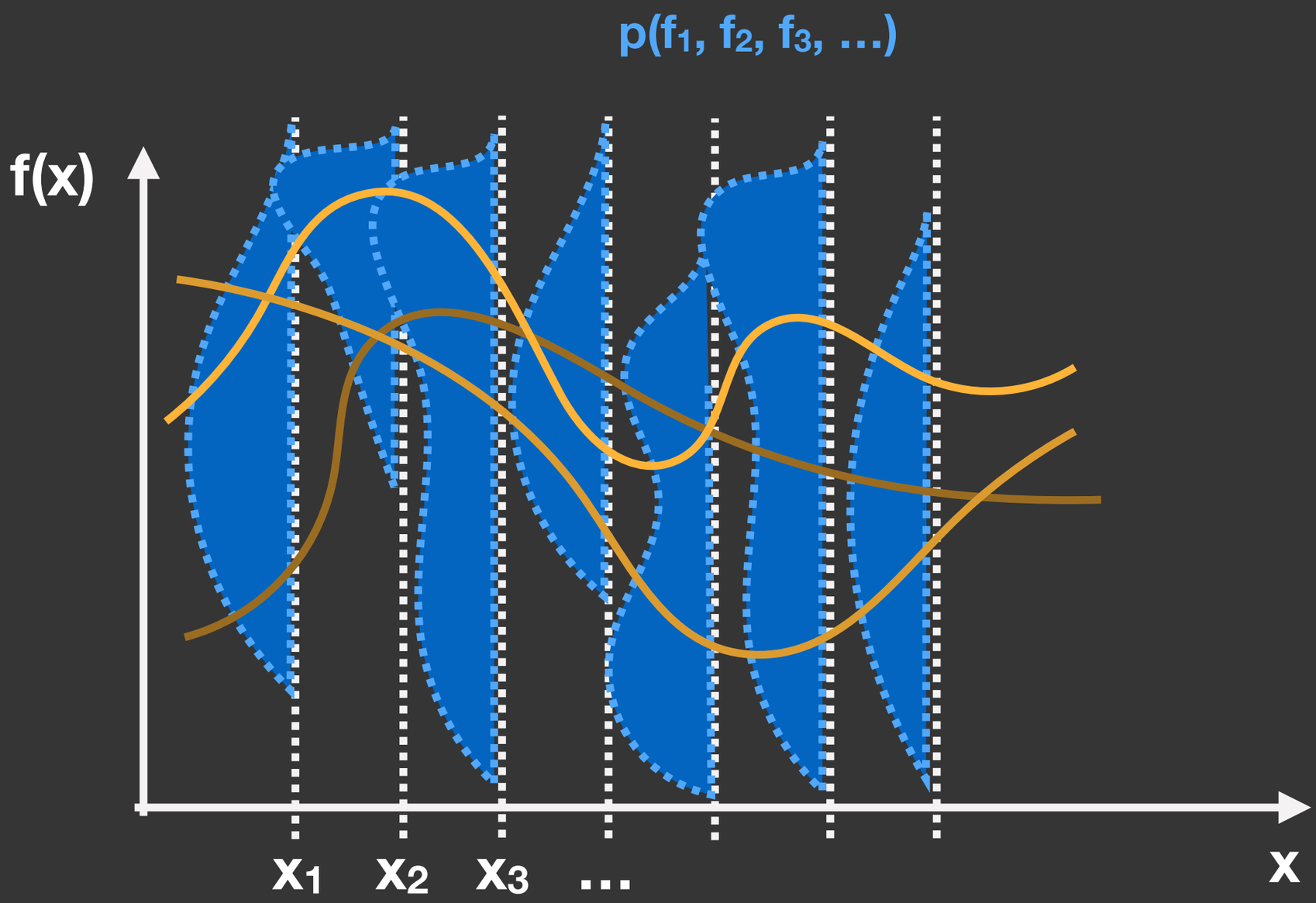
Now learning $\mathbf{f_1}$ tells us something about probable values for $\mathbf{f_2}$ and $\mathbf{f_3}$



Limit of **Δx → 0**: joint prior **p(f_1, f_2, f_3, ...)** → a **prior on function space**



Limit of **Δx → 0**: joint prior **p(f_1, f_2, f_3, ...)** → a **prior on function space**



The ideal solution:

- Incorporate all our prior knowledge about the problem in a joint prior **$p(f_1, f_2, f_3, \ldots)$** of **any form we choose**
- Perform the **full, expensive calculation of f(x)** for **some of the x-values**
- For any other x-value **x'**: obtain our best guess for the corresponding **f'** in the form of the posterior **$p(f' \mid f_1, f_2, \ldots)$**

But doing this with arbitrary prior pdfs is not practically feasible…

Gaussian processes

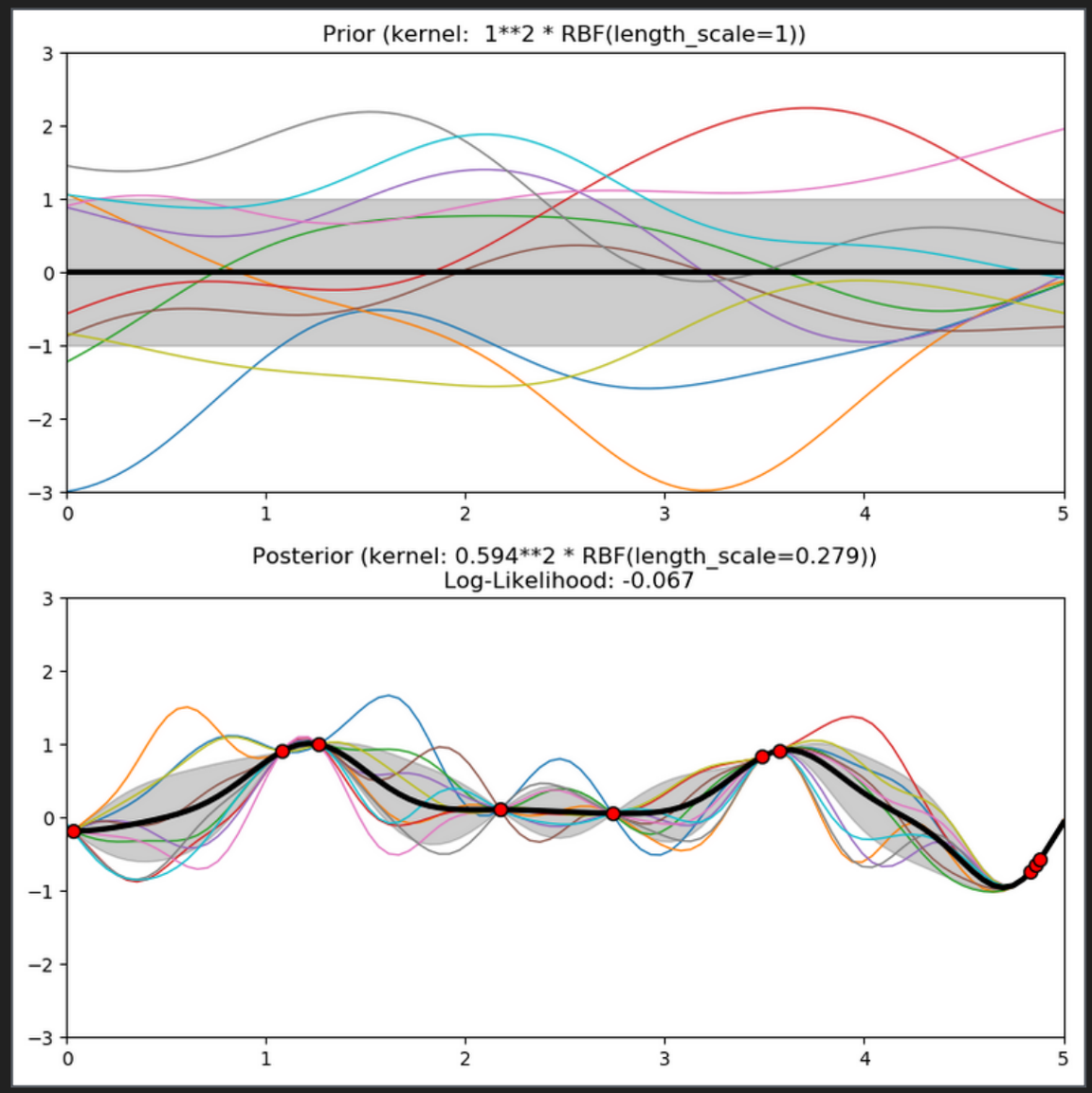
- **Gaussian process:** an infinite set of variables such that for any finite collection of these variables, the **joint pdf is a multivariate Gaussian distribution**
- Defined by **a mean** and **a covariance matrix**
- The multivariate Gaussian has some very important properties:
  - **marginalising** (intergrating) out some variables **gives another Gaussian**
  - **conditioning** on some variables **gives another Gaussian**

What this means for regression

- Start from joint Gaussian prior
- Generally intractable pdf calculations now reduce to **analytical expressions** for a **new mean** and **a new covariance matrix**
- *In particular:* simple, closed-form expression to get the **posterior from the prior**, **$p(f_1, f_2, \ldots, f') \rightarrow p(f' \mid f_1, f_2, \ldots)$**

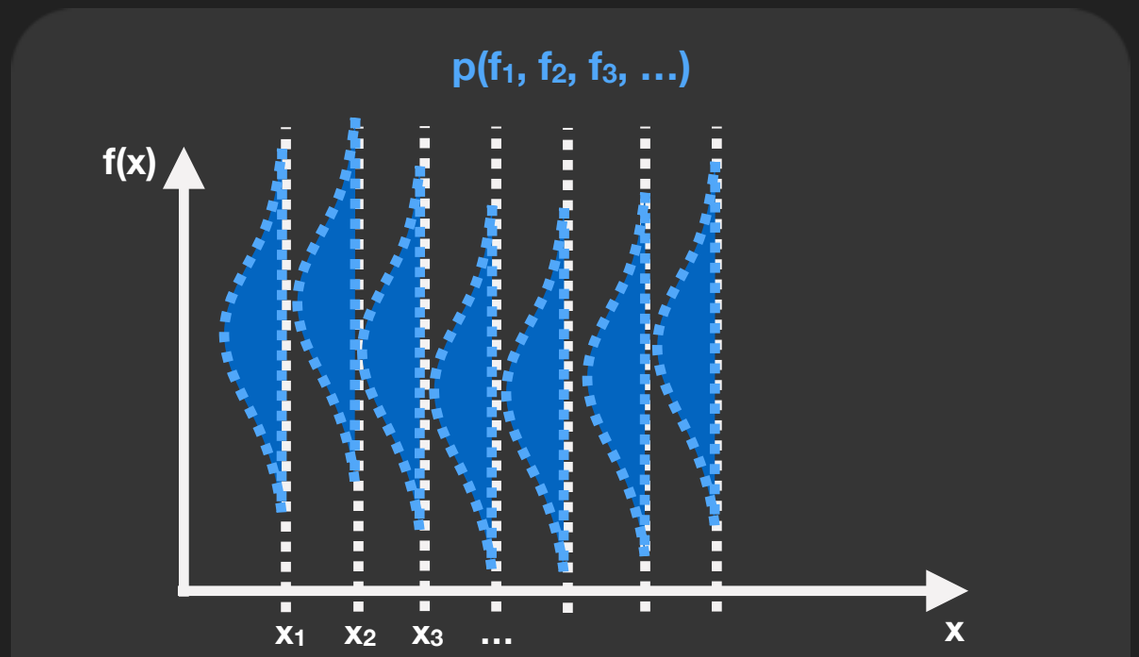
Gaussian processes regression

- Use probabilistic inference to **learn a function from data** in an **interpretable**, yet **non-parametric** framework
- *A major advantage*: the probabilistic framework **automatically provides regression uncertainty**
- *Main drawback*: given N data points, must invert the N x N covariance matrix
  → **can't use too large datasets**
  (but there are ways to alleviate the problem)
- Standard GP reference: Rasmussen & Williams, *Gaussian Processes for Machine Learning*



Choosing our Gaussian prior

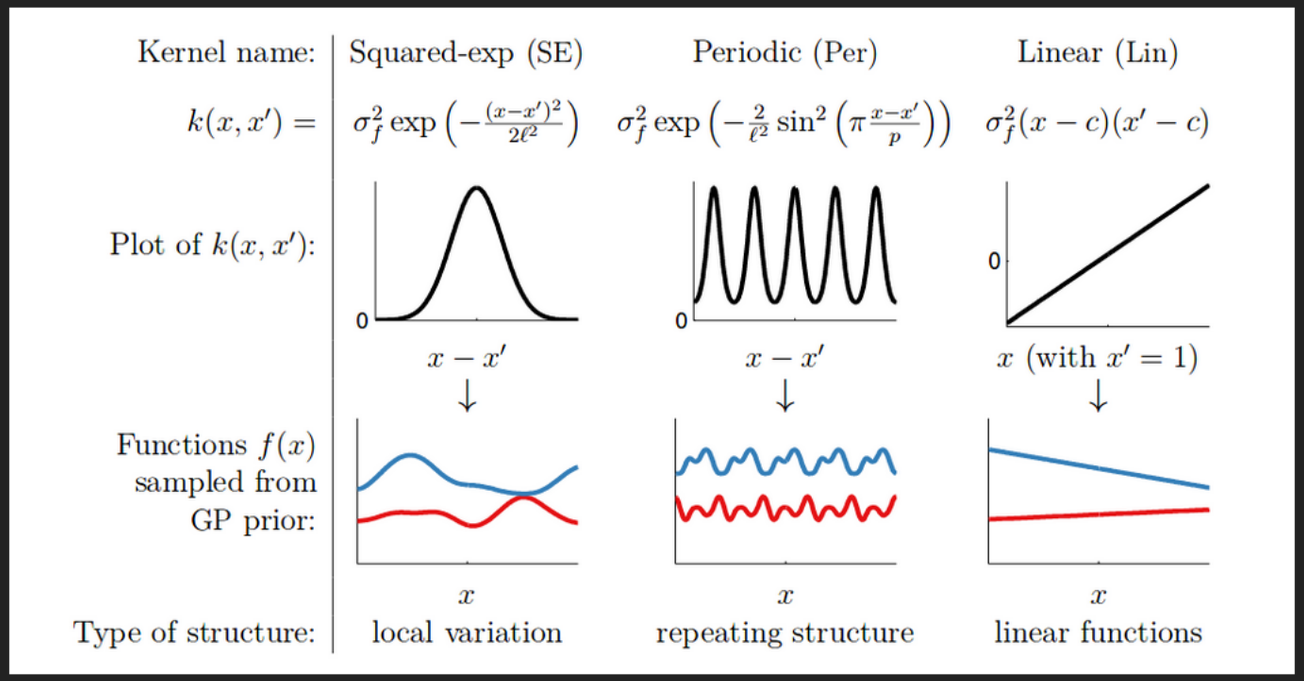
- Need to specify:
  - the means $\mathbf{E[f_i]}$
  - the covariance $\mathbf{cov(f_i,f_j)}$ for all pairs $\mathbf{(f_i,f_j)}$
- Do this indirectly by choosing
  - a mean function $\mathbf{m(x)}$, such that $\mathbf{E[f_i] = m(x_i)}$
  - a covariance (kernel) function $\mathbf{k(x,x')}$ such that $\mathbf{cov(f_i,f_j) = k(x_i,x_j)}$
- The art of GP regression:
  - Choose/design $\mathbf{k(x,x')}$ to match what you expect about the unknown $\mathbf{f(x)}$



MY KINGDOM FOR A GOOD KERNEL ...

The choice of kernel allows for great **flexibility**. But once chosen, it fixes the type of functions likely under the GP prior and determines the kind of structure captured by the model, e.g., periodicity and differentiability.

[D. Duvenaud, PhD thesis]



13

[Slide from Jeriek Van den Abeele]

The covariance function hyperparameters

- Typically don't choose a fully specified **k(x, x')**, but rather some parameterised function **k(x, x' ; θ)**

- **To be fully consistent (Bayesian):** introduce prior **p(θ)** and marginalise over **θ** to obtain the posterior

$$p(f'|f_1, f_2, \ldots) \propto \int p(f'|f_1, f_2, \ldots, \theta) p(f_1, f_2, \ldots|\theta) p(\theta)\, d\theta$$

…but this is computationally very expensive

- **Common approach:** Fix hyperparameters by **maximising the log-likelihood** (training step)

$$\log p(f_1, f_2, \ldots|\theta)$$

Conceptually: adjust your prior until the belief it assigned to the known function values (training points) is maximised.

xsec: the cross-section evaluation code

- A new **Python tool**
- **Pre-trained GPs** that evaluate BSM production cross-sections at **NLO**
  (hours → fractions of a second)
- For v1.0: BSM = supersymmetry (SUSY)
- All **strong SUSY** cross-sections in the **MSSM-24** for LHC @ 13 TeV
- Provides **pdf, scale** and α_s uncertainties, in addition to the subdominant GP regression uncertainty
- pip installable, source code on github

README.md

xsec: the cross-section evaluation code

arxiv 2006.16273 | release v1.0.2 | pypi v1.0.2 | build passing | python 2.7 | 3 | code style black | license GPL-3.0

`xsec` is a tool for fast evaluation of cross-sections, taking advantage of the power and flexibility of Gaussian process regression.

For a detailed description of the methodology, validation and instructions for usage, see https://arxiv.org/abs/2006.16273.

Installation

Requirements

`xsec` is compatible with both Python 2.7 and 3. The following external Python libraries are required to run `xsec`:

- setuptools 20.7.0 or later
- numpy 1.16 or later
- scipy 1.0.0 or later
- joblib 0.12.2 or later
- dill 0.3.2 or later
- pyslha 3.2.3 or later

`pip` installation

`xsec` can be installed from PyPI using

```
pip install xsec
```

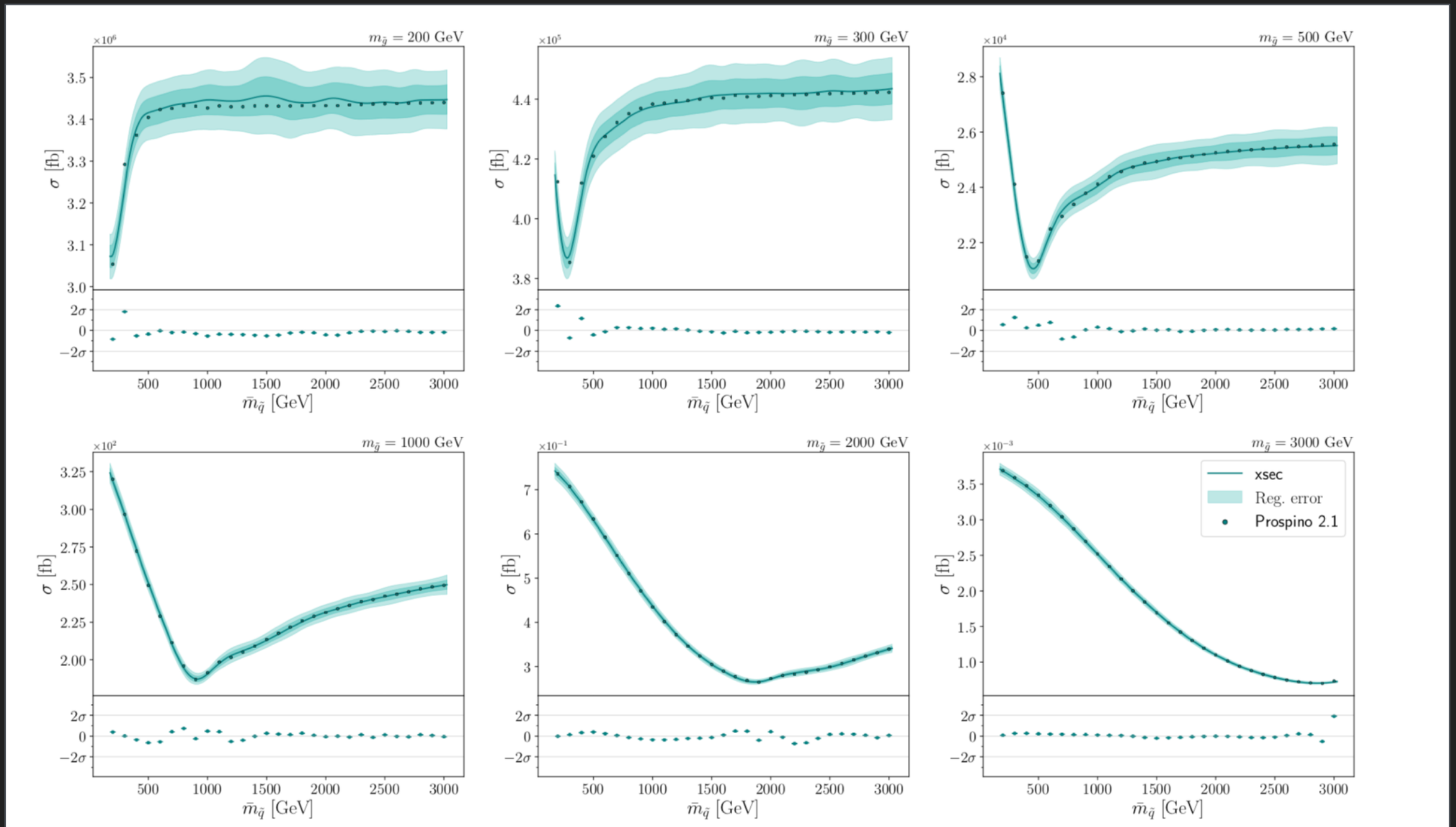
Optionally with the `--user` flag if your IT department is mean.

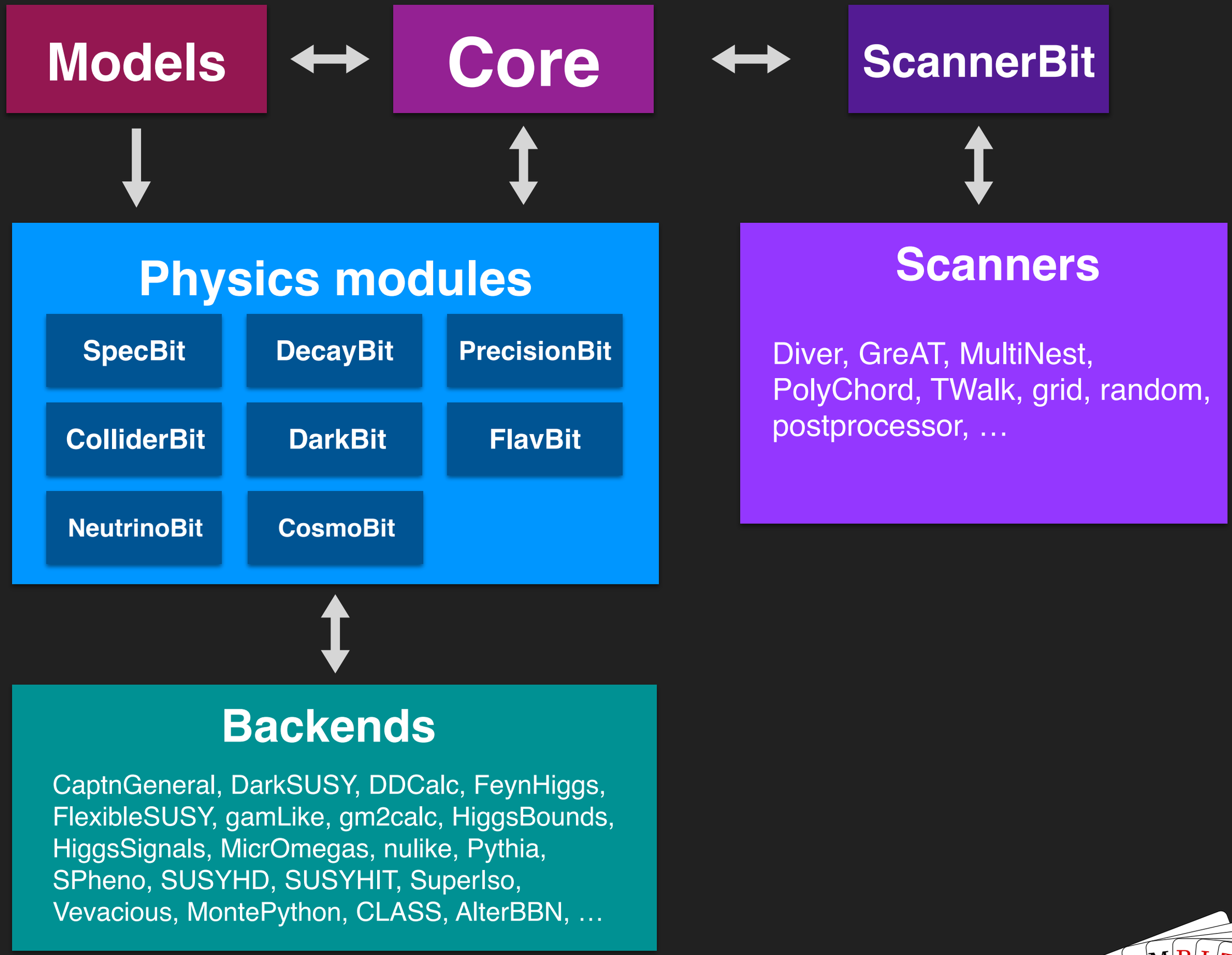
Alternatively, you can clone the repo from GitHub and install from there:

```
git clone https://github.com/jeriek/xsec.git
pip install ./xsec
```

A. Buckley, AK, A. Raklev, P Scott, J.V. Sparre,
J. Van den Abeele, I. A. Vazquez-Holm
[arXiv:2006.16273]
github.com/jeriek/xsec

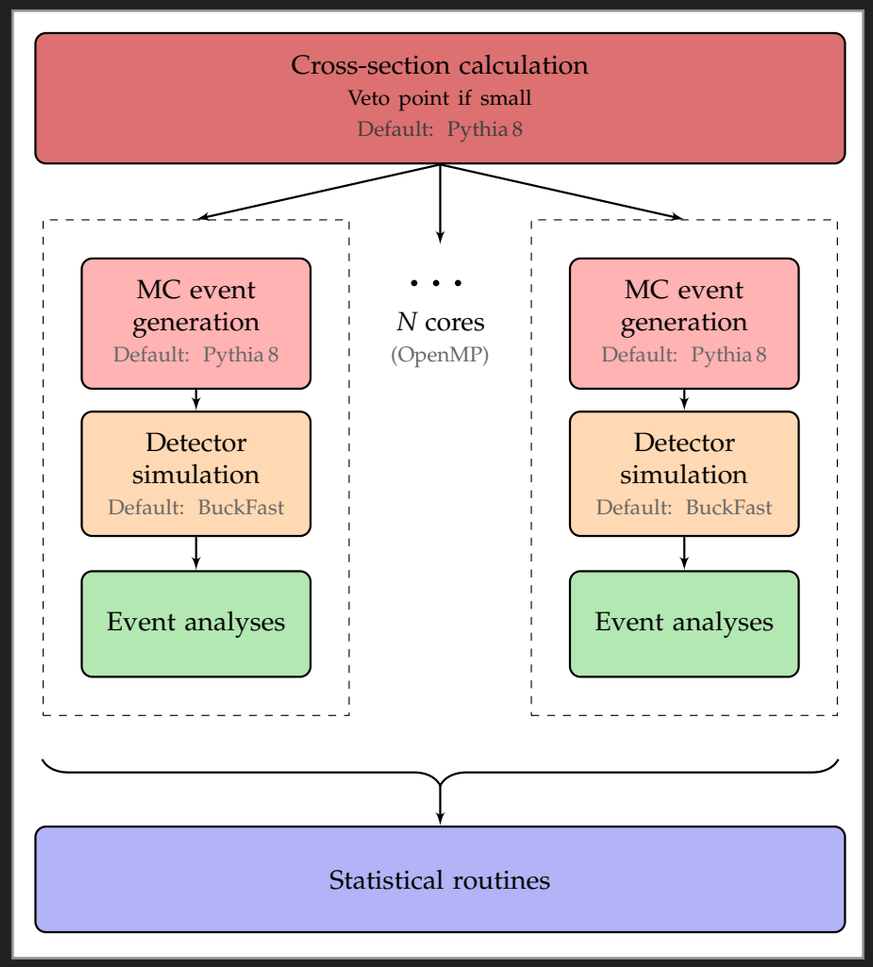
gluino—gluino

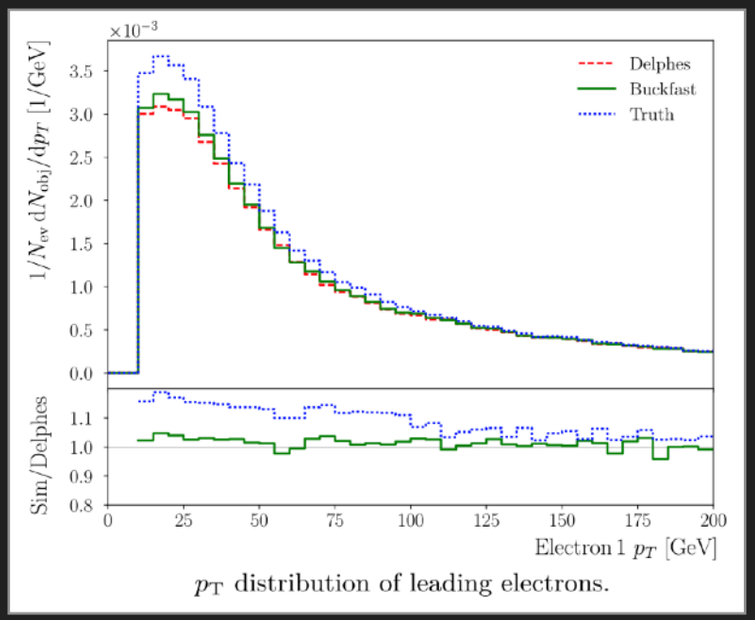



Models
Core
ScannerBit
Physics modules
SpecBit
DecayBit
PrecisionBit
ColliderBit
DarkBit
FlavBit
NeutrinoBit
CosmoBit
Scanners
Diver, GreAT, MultiNest, PolyChord, TWalk, grid, random, postprocessor, …
Backends
CaptnGeneral, DarkSUSY, DDCalc, FeynHiggs, FlexibleSUSY, gamLike, gm2calc, HiggsBounds, HiggsSignals, MicrOmegas, nulike, Pythia, SPheno, SUSYHD, SUSYHIT, SuperIso, Vevacious, MontePython, CLASS, AlterBBN, …

ColliderBit

- **LHC particle searches:** Full Poisson likelihood from fast MC simulation of LHC searches
  - Parallellized MC event generation and analysis loop inside ColliderBit
  - Event generation with Pythia 8
  - Fast detector simulator: BuckFast (4-vector smearing)
- **Focus on speed,** as required for use in global fits
- **Extensive list ATLAS/CMS searches** and more being added…
- **LEP limits (SUSY):** Calculate $\sigma \times BR$ and check against published limits





arXiv:1705.07919

The basic steps of a BSM global fit

- Choose your **BSM model and parameterisation**
- Construct the **combined likelihood function** including observables from collider physics, dark matter, flavor physics, +++
- Use **sophisticated scanning techniques** to explore the likelihood function across the parameter space of the theory
- Test **parameter regions** in a statistically sensible way — not just single points *(parameter estimation)*
- Test **different theories the same way** *(model comparison)*

Constructing a likelihood function

How to compare the predictions of a theory against *many* experimental results?

$$\text{Theory} \rightarrow f(x;\theta)$$

$$\text{Experiment} \rightarrow \mathcal{L}(\theta) = f(x_{\text{data}};\theta)$$

$$\mathcal{L} = \mathcal{L}_{\text{Collider}}\,\mathcal{L}_{\text{Higgs}}\,\mathcal{L}_{\text{DM}}\,\mathcal{L}_{\text{EWPO}}\,\mathcal{L}_{\text{Flavor}}\,\ldots$$