N-Tuples and compact matrix element representations

Daniel Maître, IPPP Durham

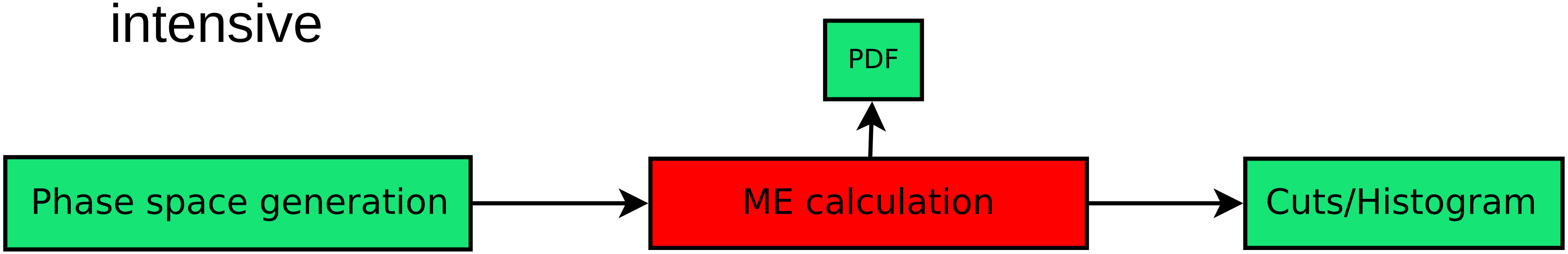
ACAT 19, 13 March 2019

Plan

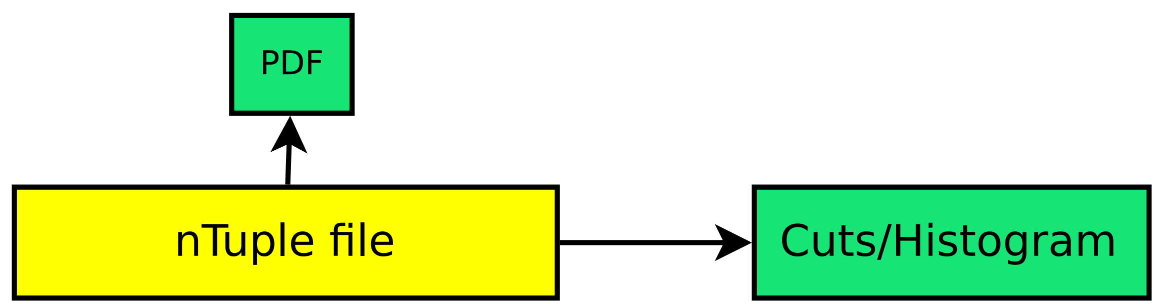
- nTuples
- NNLO ntuples
- Orthogonal functions for phasespace

n-Tuple files [arXiv:1310.7439]

- High multiplicity NLO calculations are computationally intensive



- Matrix elements are expensive
- Jet clustering, observables, PDF evaluation are relatively cheap
- Store matrix element, PS point and the information necessary to change scales

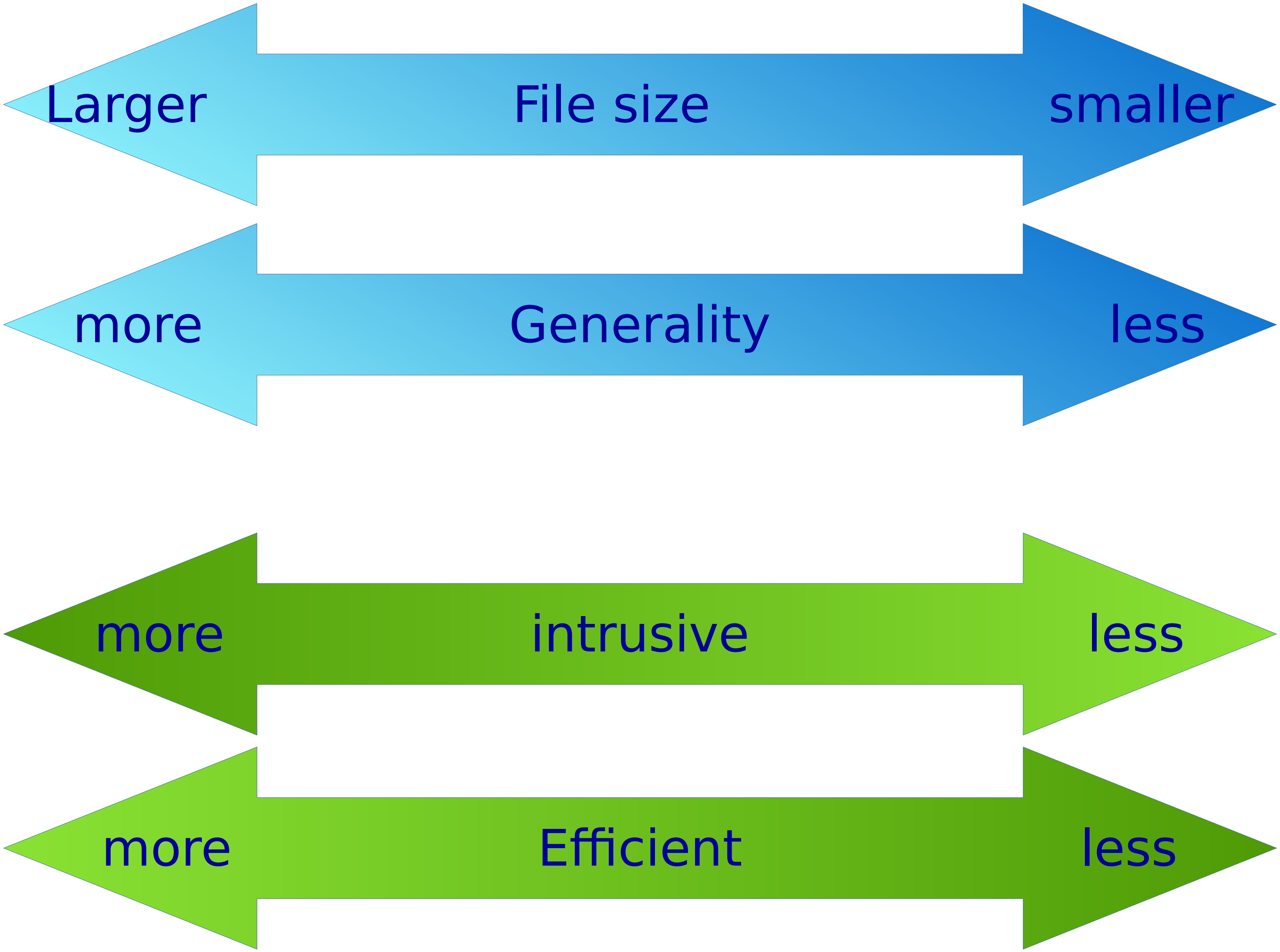


NLO nTuple files [arXiv:1310.7439]

- Advantages
  - One can change the analysis cuts, add observables
  - Cheap scale variation and PDF errors (otherwise extremely expensive)
  - Easy communication between theorists and experimenters
  - No need for specific know-how of the tool which produced the NLO calculation
  - Easier to “endorse” an event file than a program
- Disadvantage
  - Large files
  - Generation cuts need to be loose enough to accommodate many analysis $\rightarrow$ efficiency cost

NNLO nTuple files

- Trade offs



Compact Matrix element representations

nTuple file

- A nTuple file is a weighted sample to represent the integral

$$\sigma = \int d\phi_n \frac{d\sigma}{d\phi} C(\phi)$$

- Where *C* is a set of cuts designed to be as inclusive as possible
- ϕ is the phasespace, and also include the integration over the PDFs for hadronic initial states
- For hadronic initial states we need to create new nTuples
  - For new collider energies
  - For different jet pt cuts

Compact representation

- Question: if calculating the differential cross section is the difficult part, how can I store as much information as possible about it in a way that I can reuse it more?
- I can try to represent the underlying probability density
  - Is it a probability density?
    - Yes at LO
    - Not at NLO
    - It should be for an infrared-safe observable
- Goal is to build a basis of phasespace functions that can build any infrared safe observable
- Consider 1 $\rightarrow$ n process (e+e-, Higgs decay)

Orthogonal PS function

- Let's introduce as set of orthonormal functions of phasespace
- The exact form of the parametrisation is not very relevant
  - Map coordinates to either [0,1] or [-1,1]
  - Arrange for flat Jackobian:

$$\int d\phi f(\phi) = \int dx_1 \cdots dx_k f(\phi(x_1, ..., x_k))$$

  - Use polynomial orthogonal basis

$$\int_{x_{min}}^{x_{max}} \Phi_i(x)\Phi_j(x)\, dx = \delta_{ij}$$

- The basis function for the phasespace

$$\Phi_{i_1,...,i_k}(x_1, ..., x_k) = \Phi_{i_1}^{(1)}(x_1)...\Phi_{i_k}^{(k)}(x_k)$$

Orthogonal PS function

- So we can now write any (reasonable) function of phase-space as

$$f(\phi) \approx \sum_{i_1,\ldots,i_k} c_{i_1,\ldots,i_k} \Phi_{i_1,\ldots,i_k}(\phi)$$

- The number of indices corresponds to the number of free parameters (including azimuthal symmetry)
  - For 2 particle PS: 1
  - For 3 particle PS: 4
  - $3n$-5 for n particle PS
- Becomes quickly too many dimensions (curse of dimensionality)

Coefficients

- The coefficients are determined through integration over phasespace

$$c_{i_1,\ldots,i_k} = \int d\phi\, \Phi_{i_1,\ldots,i_k} f(\phi)$$

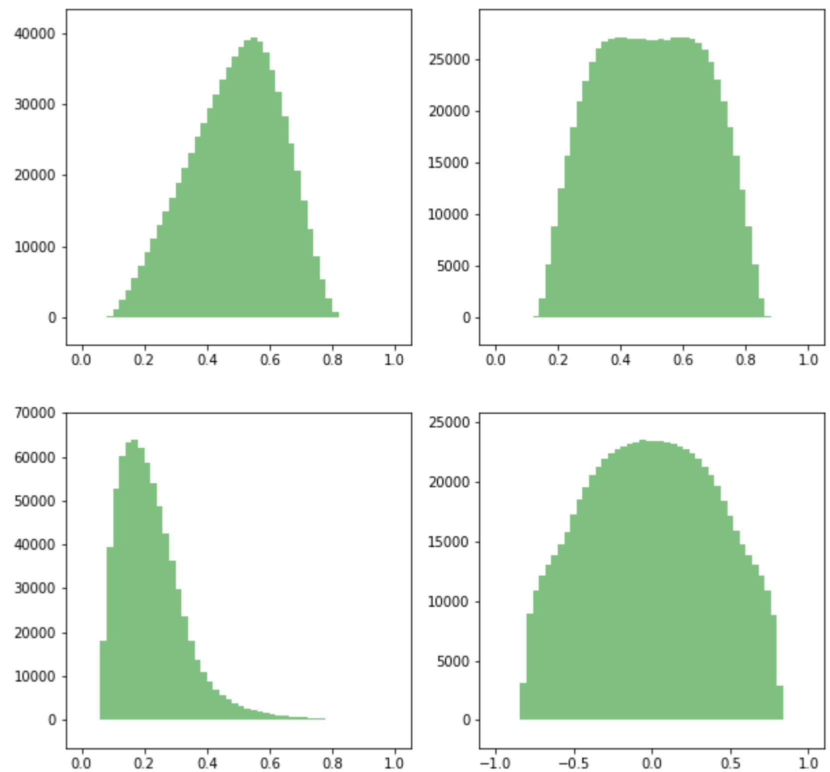
- Using this representation we have

$$\begin{aligned} \sigma &= \int d\phi_n \frac{d\sigma}{d\phi} C(\phi) \\ &= \sum_{i_1,\ldots,i_k} c_{i_1,\ldots,i_k} \int d\phi_n \Phi_{i_1,\ldots,i_k}(\phi) C(\phi) \\ &= \sum_{i_1,\ldots,i_k} c_{i_1,\ldots,i_k} d_{i_1,\ldots,i_k} \end{aligned}$$

- Phase-space integration for matrix elements and cuts can be separated

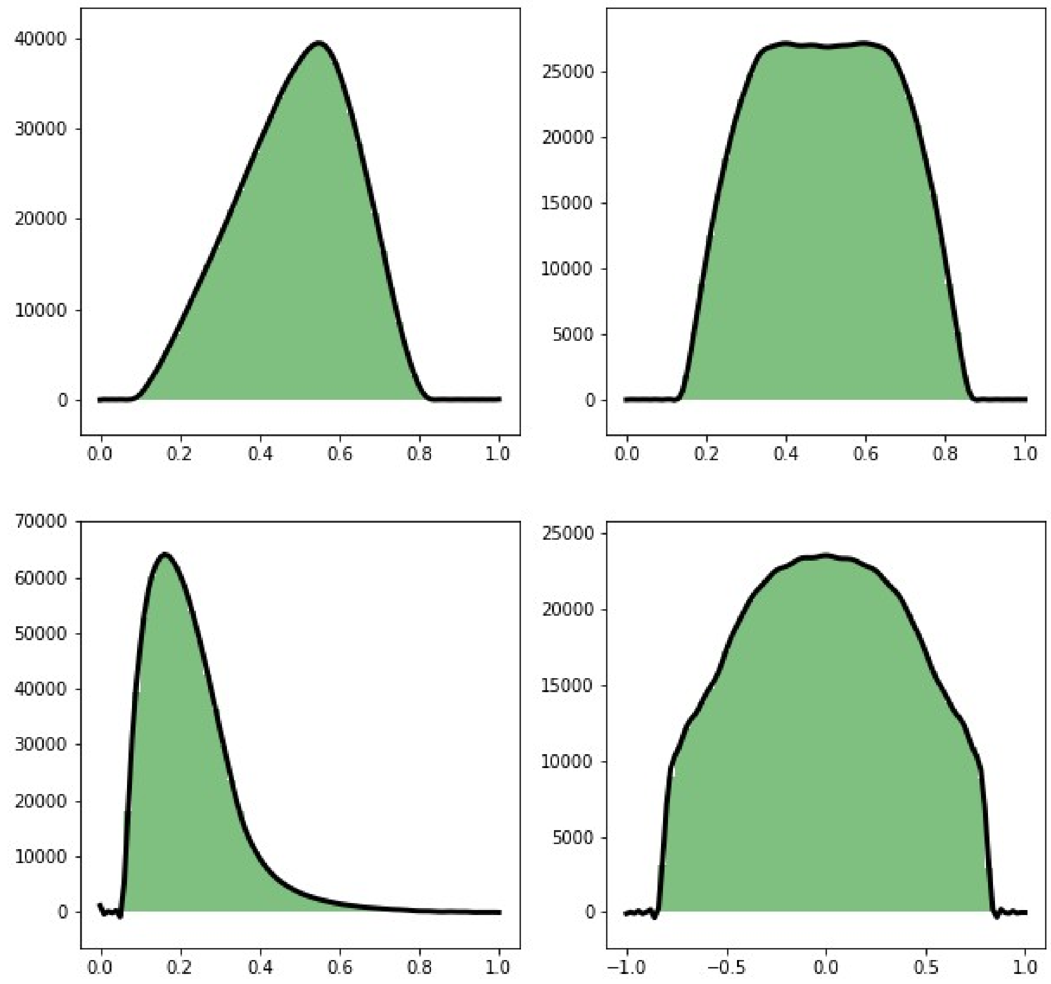
Example

- e+e- → 3j LO
  - Fix centre of mass energy to 1
  - Require 3 jets using R=0.1 and transverse momentum 0.25
- The densities $\frac{d\sigma}{dx_i}$ in phase space look like this:



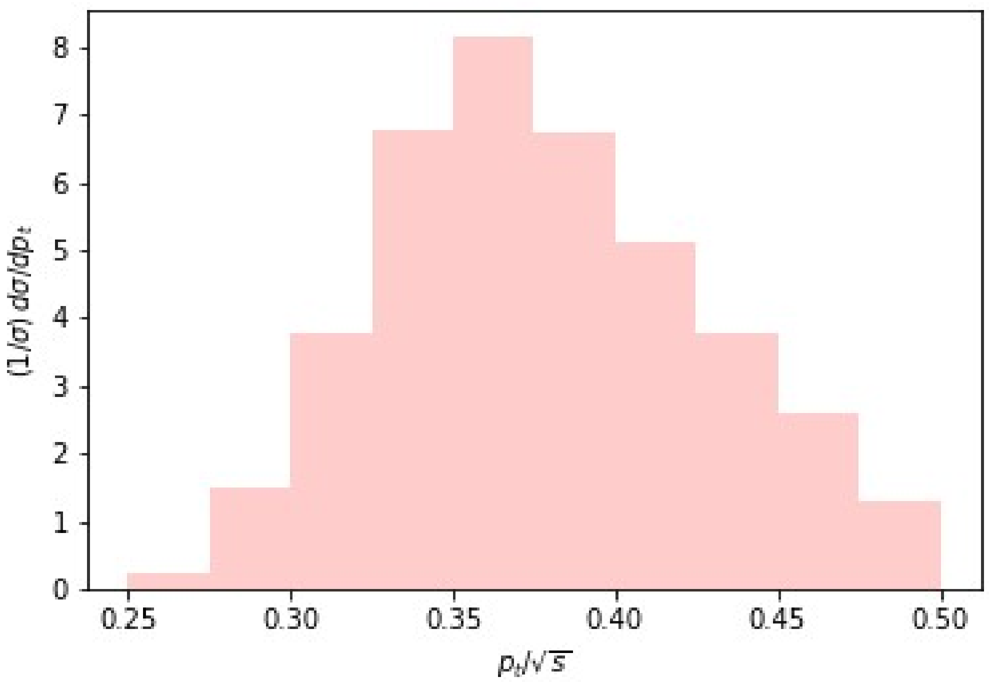
Estimate of the densities

- Calculate the coefficients for the density with 50 coefficients per dimension
- Reconstruct the density and compare with the histograms:



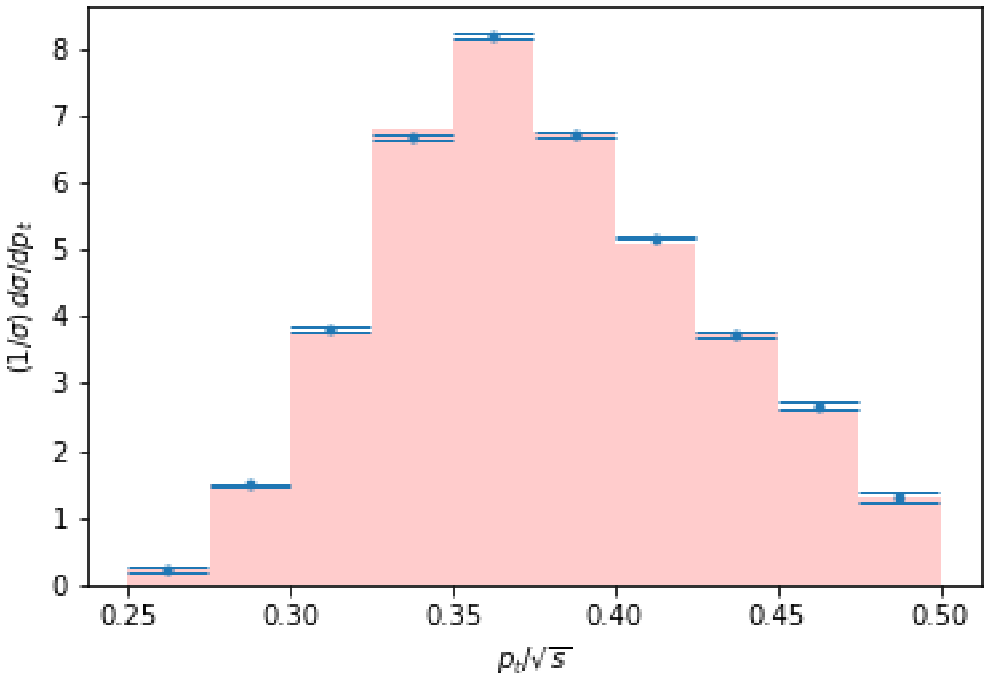
Observable

- Observables: bins in first jet transverse momentum



Reconstructed distribution

- Here are the reconstructed histograms

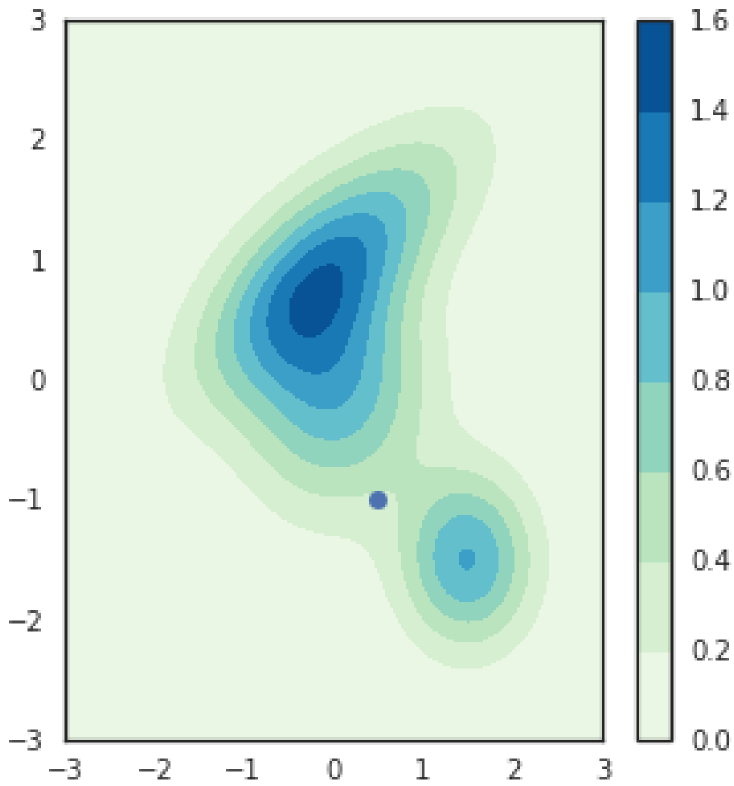


Generating samples

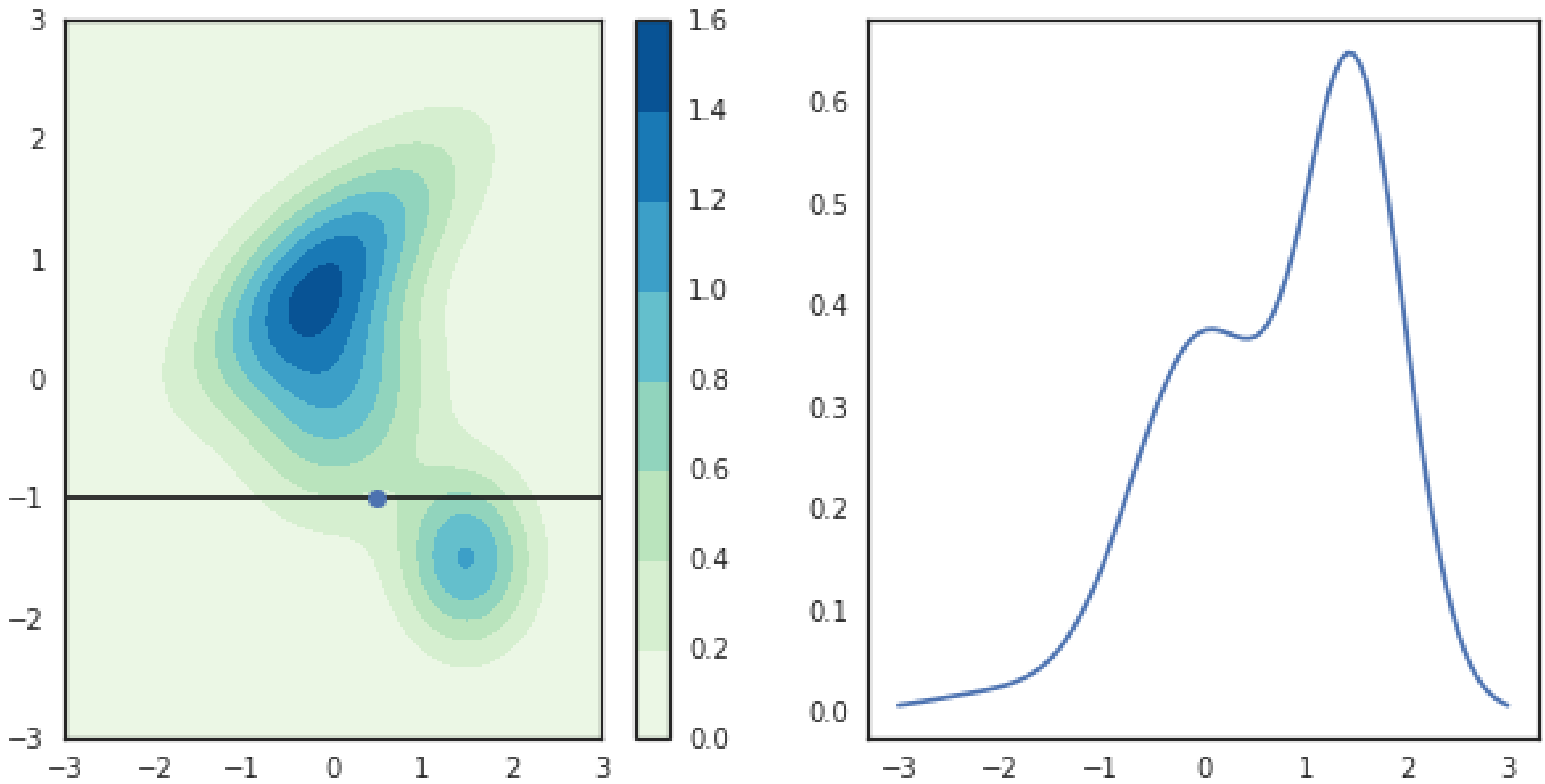
- One advantage of having an approximation for the density is that we can draw samples from it
- We can use Gibbs sampling
  - Start somewhere and repeat the following algorithm
    - Given a point $$X^i = (x_1^{(i)}, ...x_k^{(i)})$$
    - Generate the next values for each coordinate x_j successively according to the conditional probability

$$x_j^{(i+1)} \simeq P(x_j | x_1^{(i+1)}, .., x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, ..., x_k^{(i)})$$

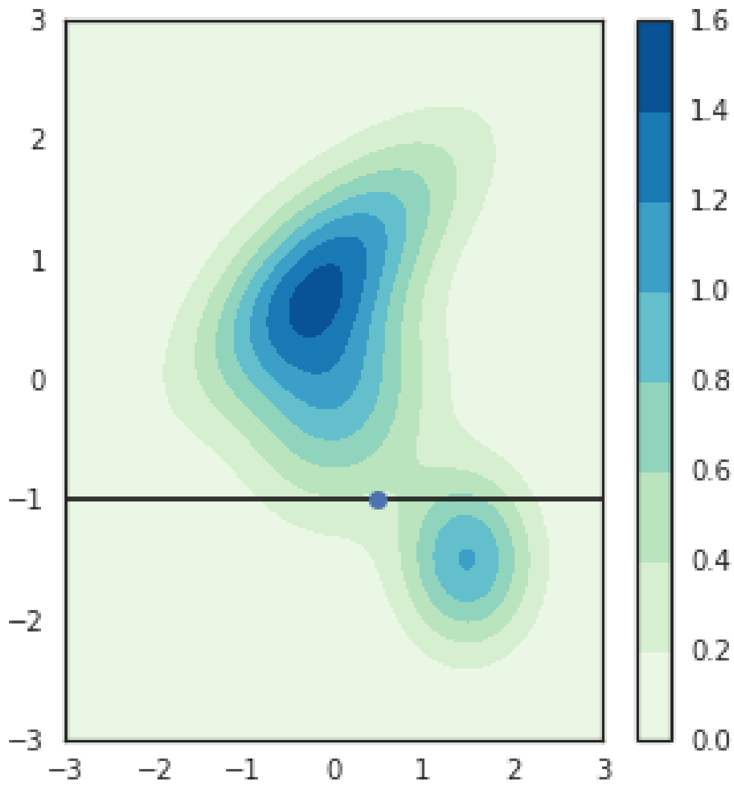
Gibbs sampling

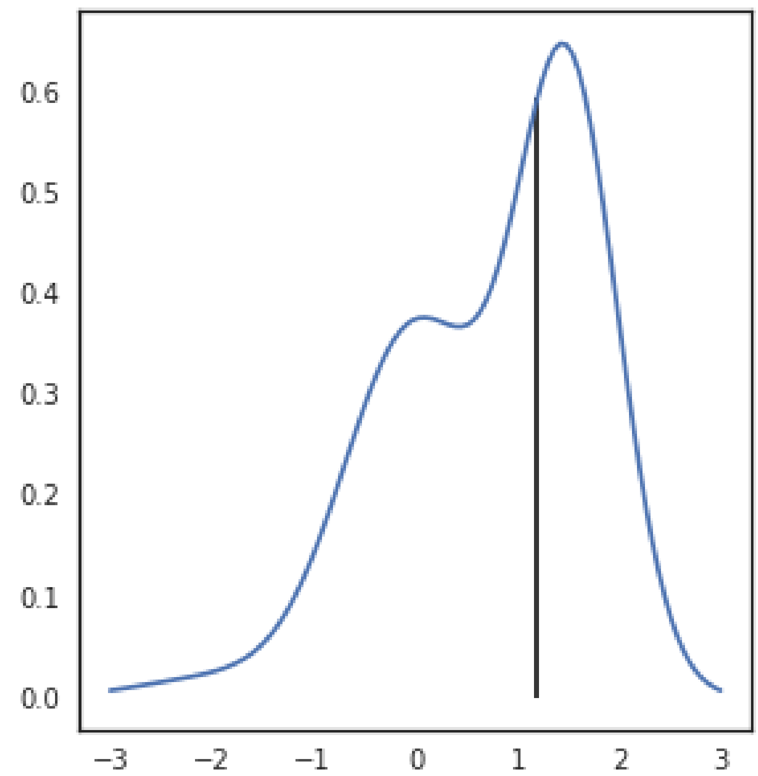


Gibbs sampling

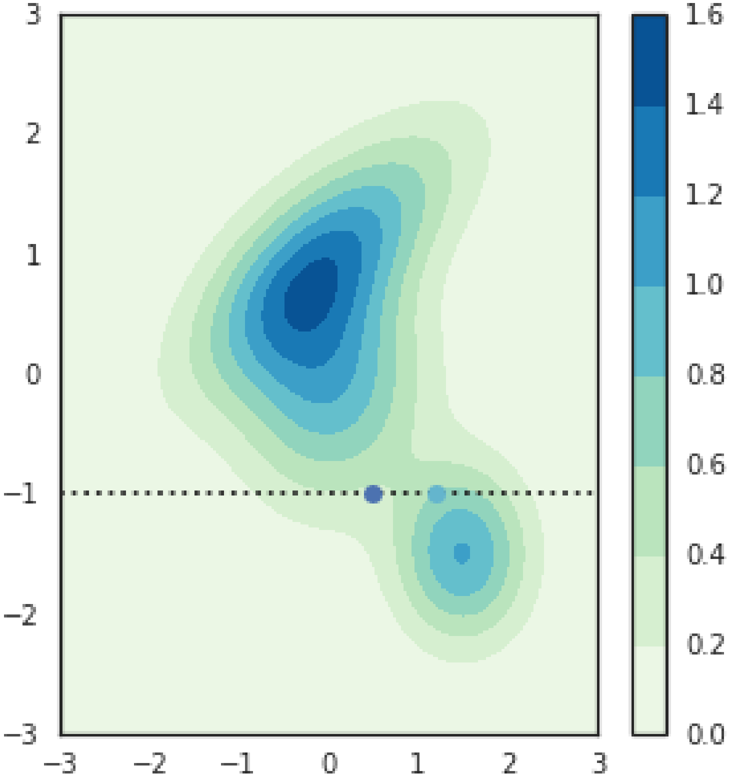


Gibbs sampling

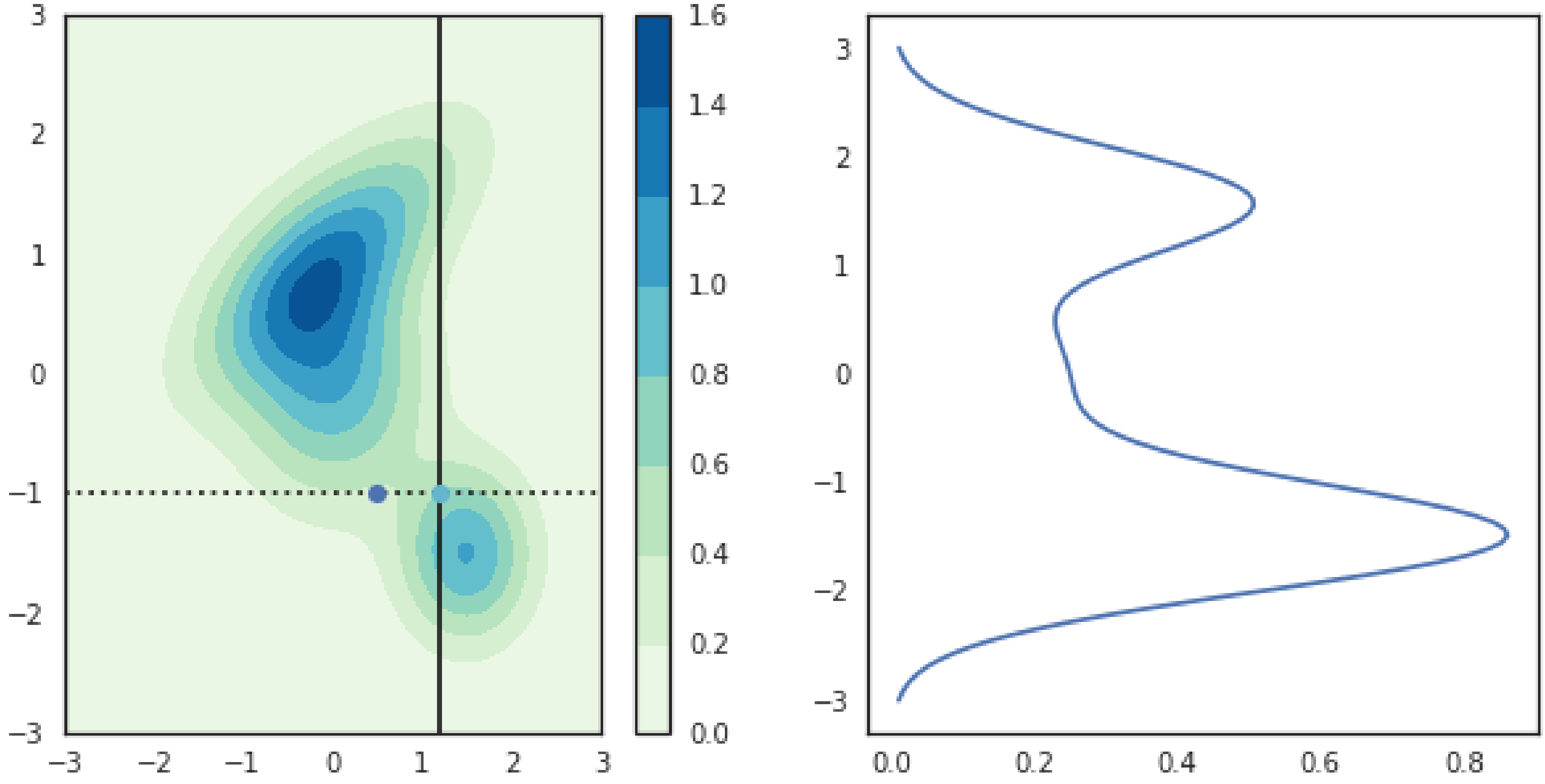




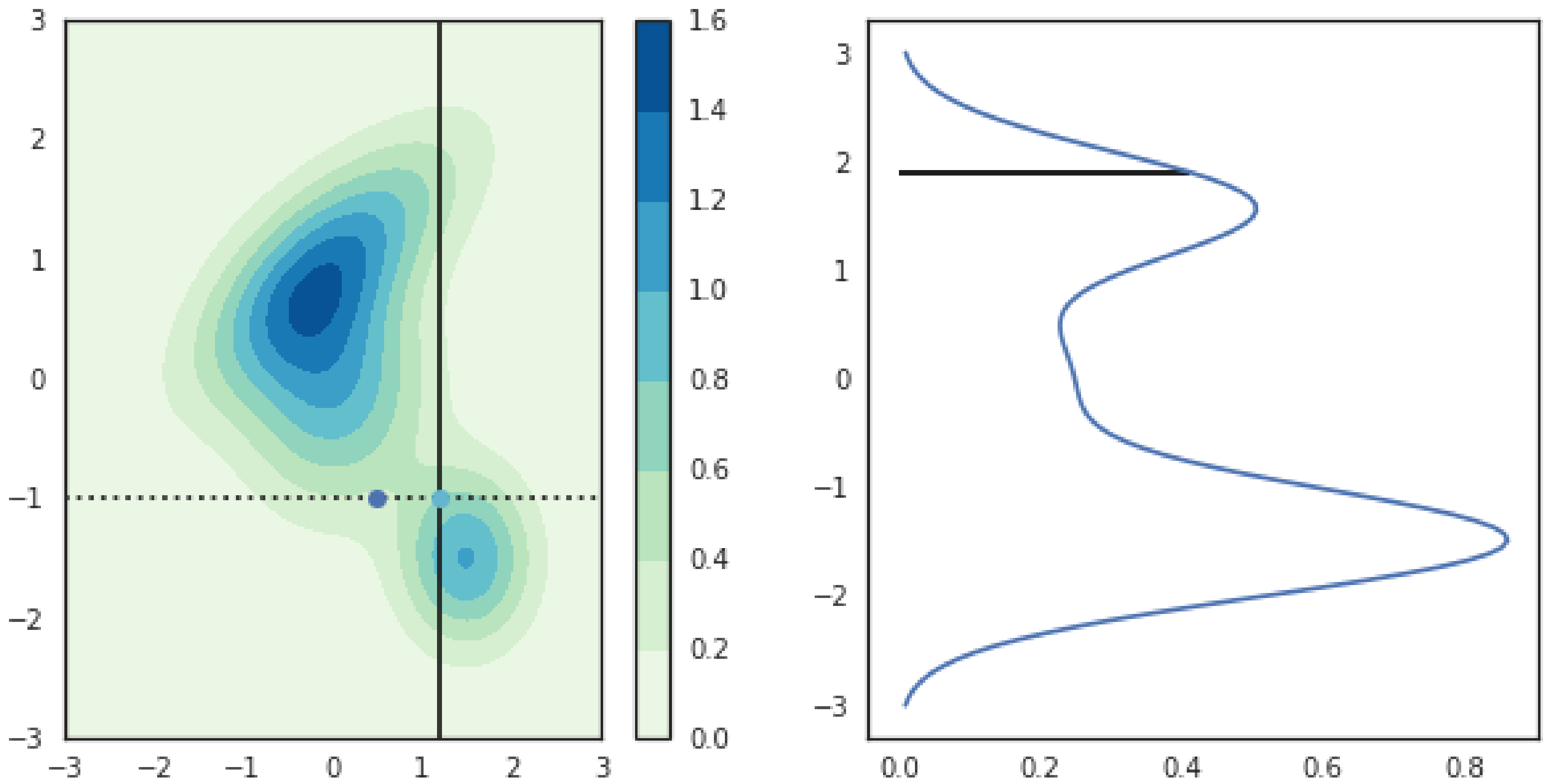
Gibbs sampling



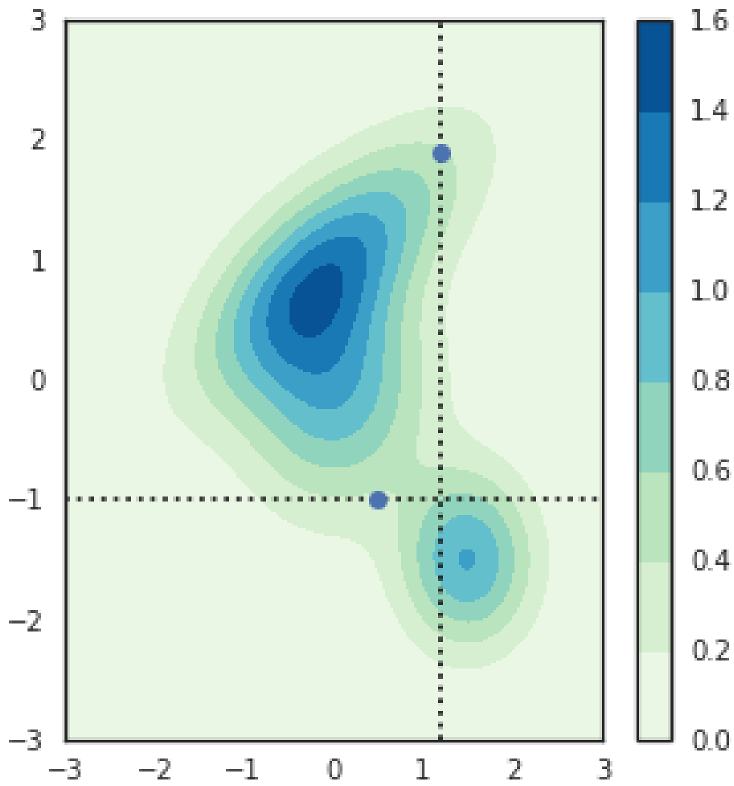
Gibbs sampling



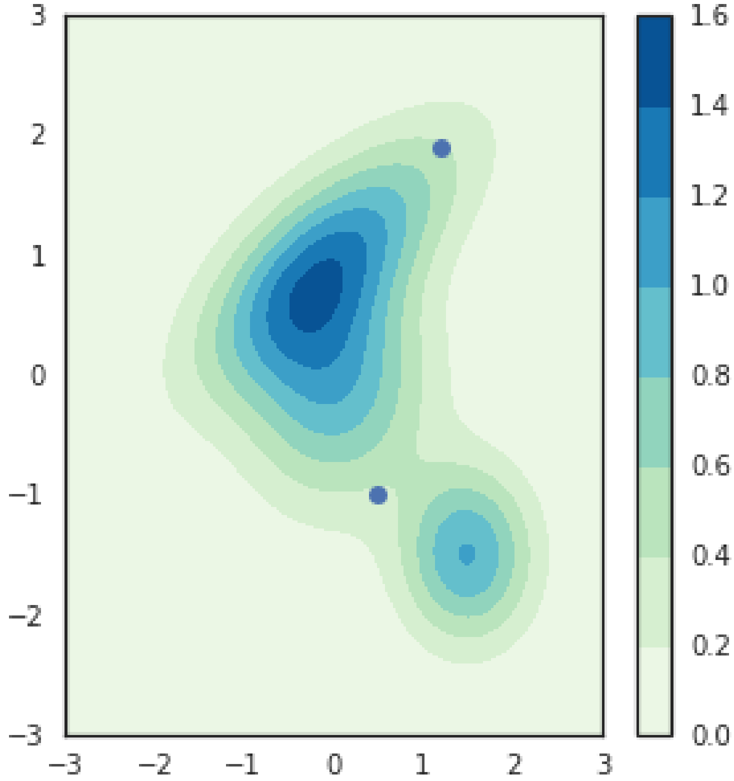
Gibbs sampling



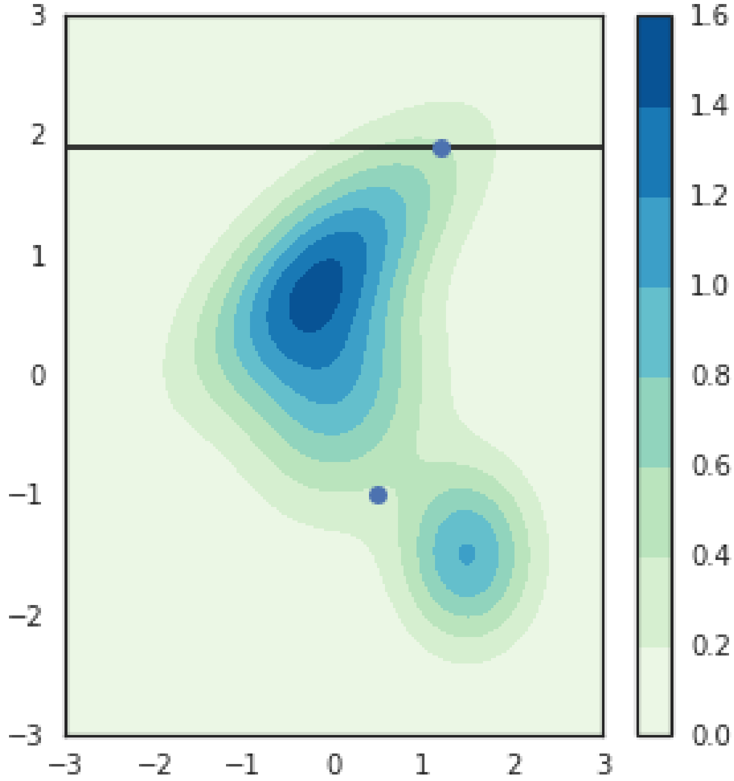
Gibbs sampling

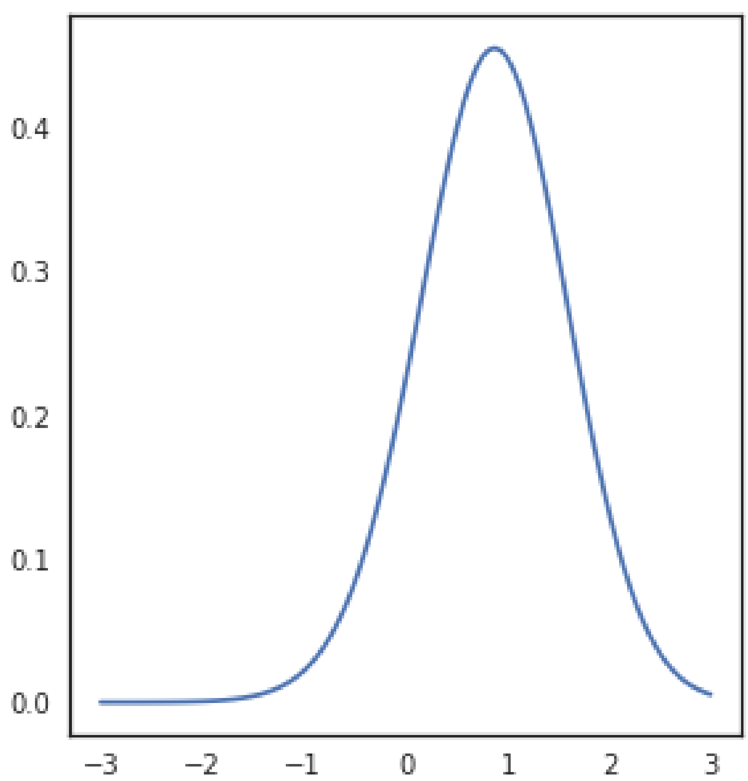


Gibbs sampling

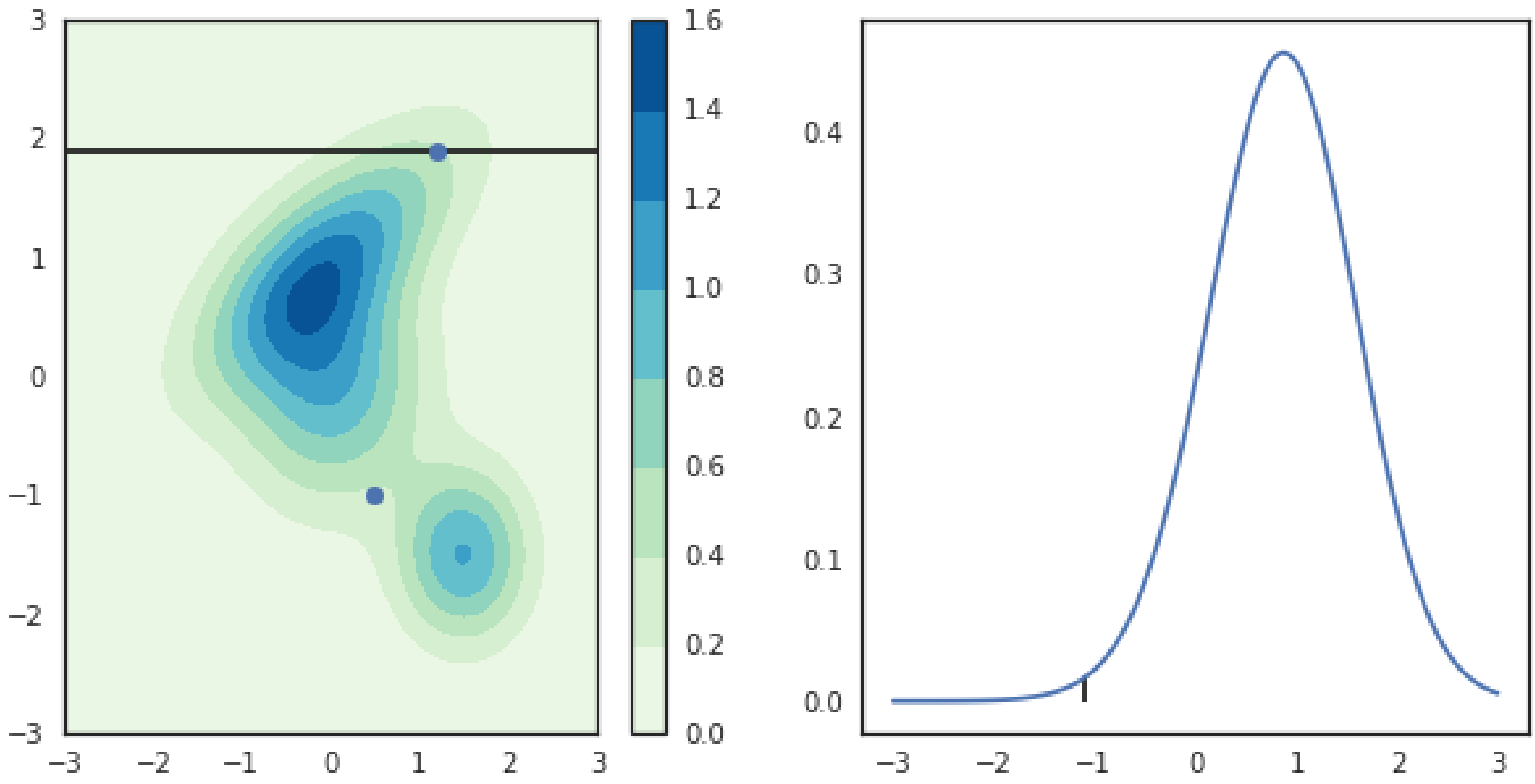


Gibbs sampling

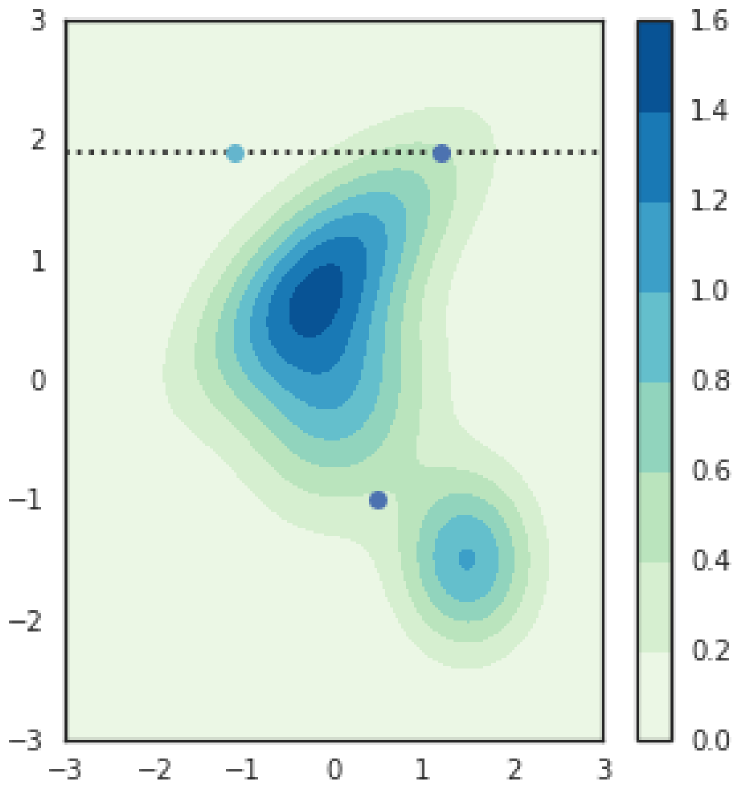




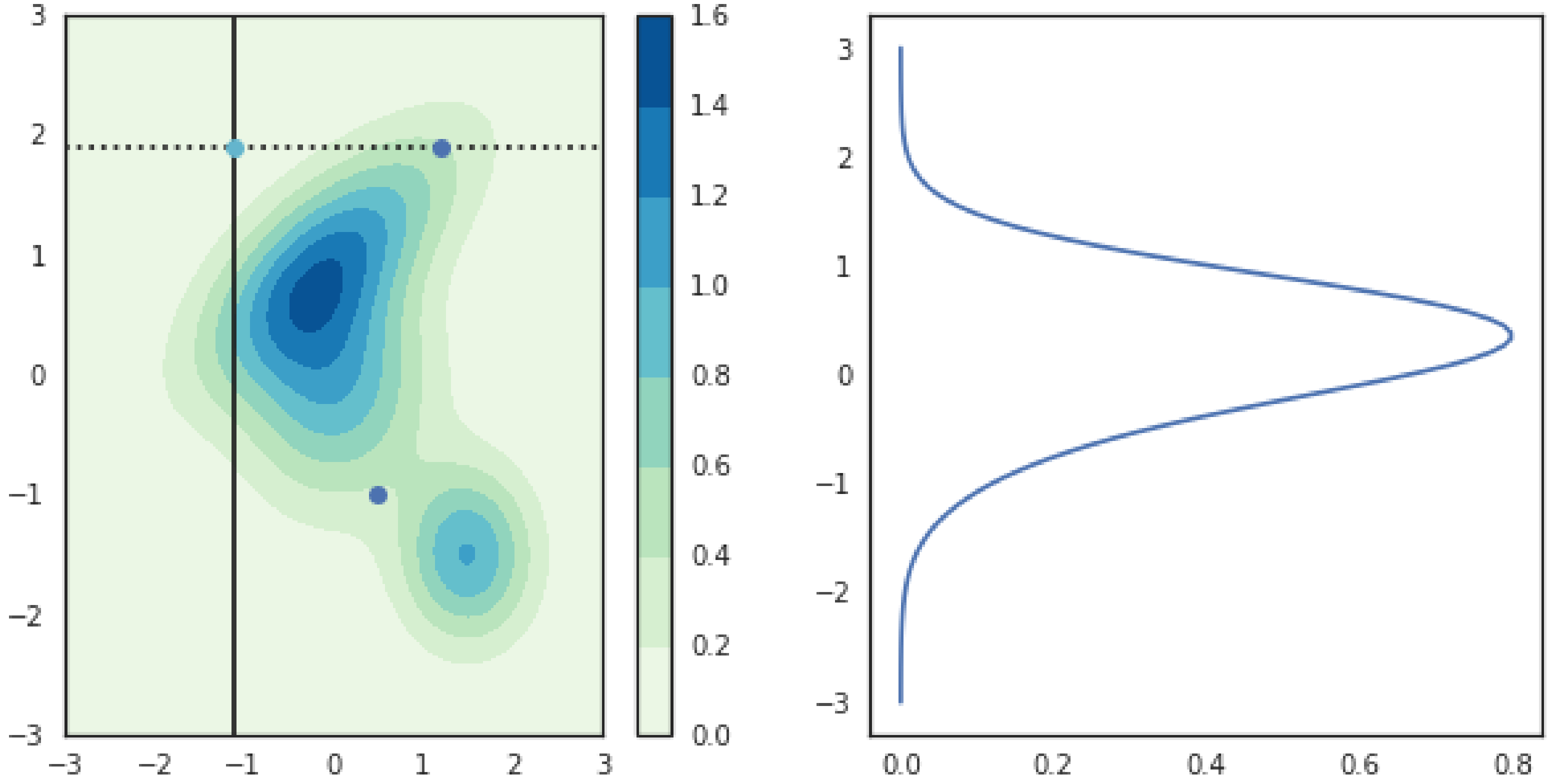
Gibbs sampling



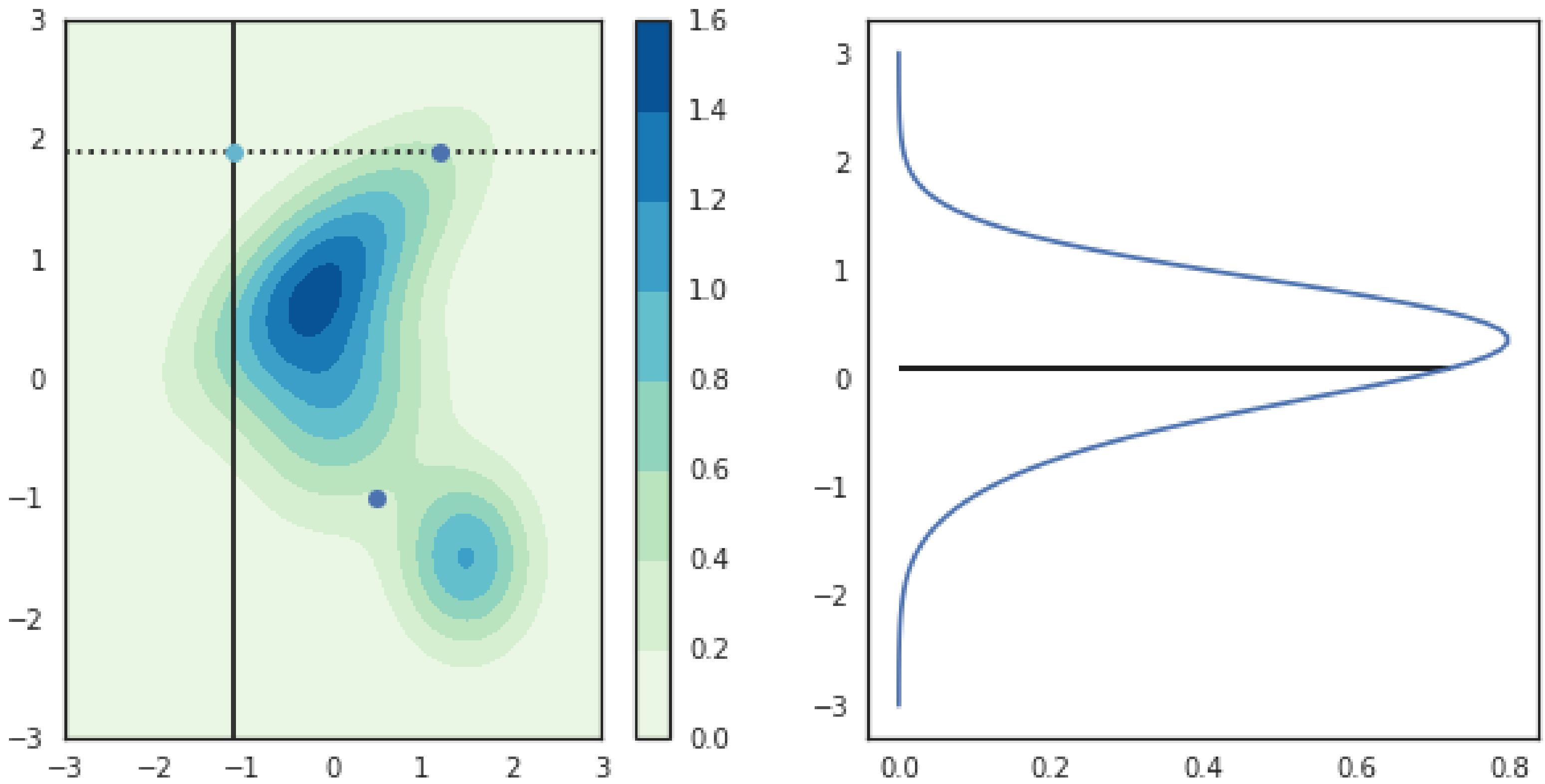
Gibbs sampling



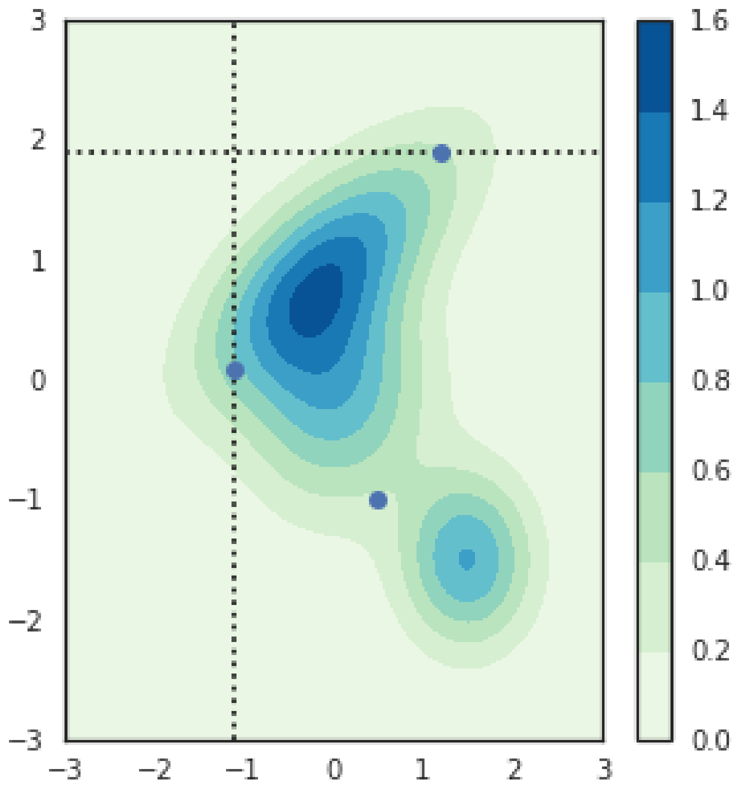
Gibbs sampling



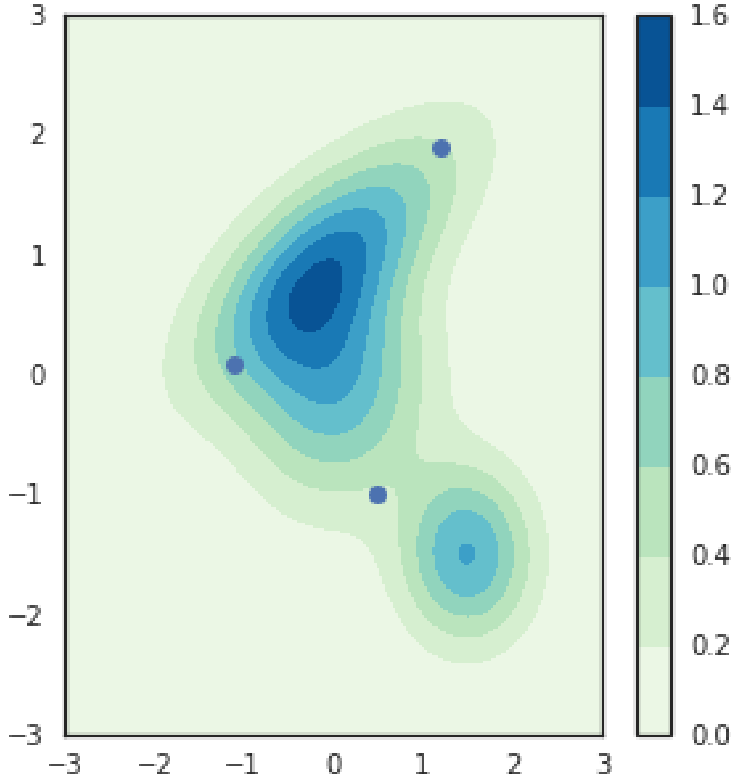
Gibbs sampling



Gibbs sampling



Gibbs sampling



Conditional probability

- With our expression for the probability density

$$f(x_1, ..., x_k) = \sum_{i_1,...,i_k} c_{i_1,...,i_k} \Phi_{i_1}(x_1) \cdots \Phi_{i_k}(x_k)$$

  it is easy to calculate the conditional probabilities

- They are naturally expressed in terms of the basis functions

$$P(x_j | x_1, ..., x_{j-1}, x_{j+1}, ..., x_k) = \sum_j d_j \Phi_j(x_j)$$

  with

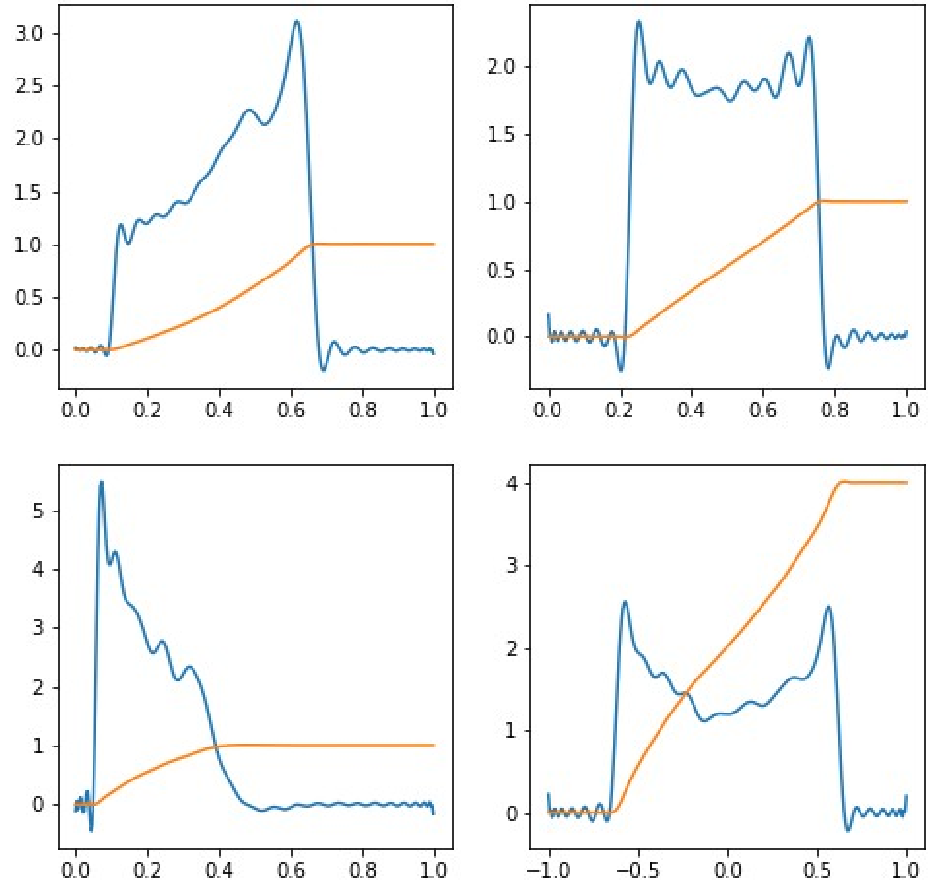
$$\begin{aligned} d_j &= \sum c_{i_1,...,i_{j-1},j,i_{j+1},...,i_k} \Phi_{i_1}(x_1) \cdots \Phi_{i_{j-1}}(x_{j-1}) \\ &\quad \times \Phi_{i_{j+1}}(x_{j+1}) \cdots \Phi_{i_k}(x_k) \end{aligned}$$

Conditional probability

- In order to draw a new value for x_j we need the cumulative distribution of the conditional probability
- The good news is that it is very easy to perform integration in the orthonormal basis:
  - The primitive of the basis functions can be written in terms of the basis functions
  - Integration is simply a multiplication with a matrix than only needs to be calculated once.
  - For the polynomial basis used here the matrix is very sparse

Example

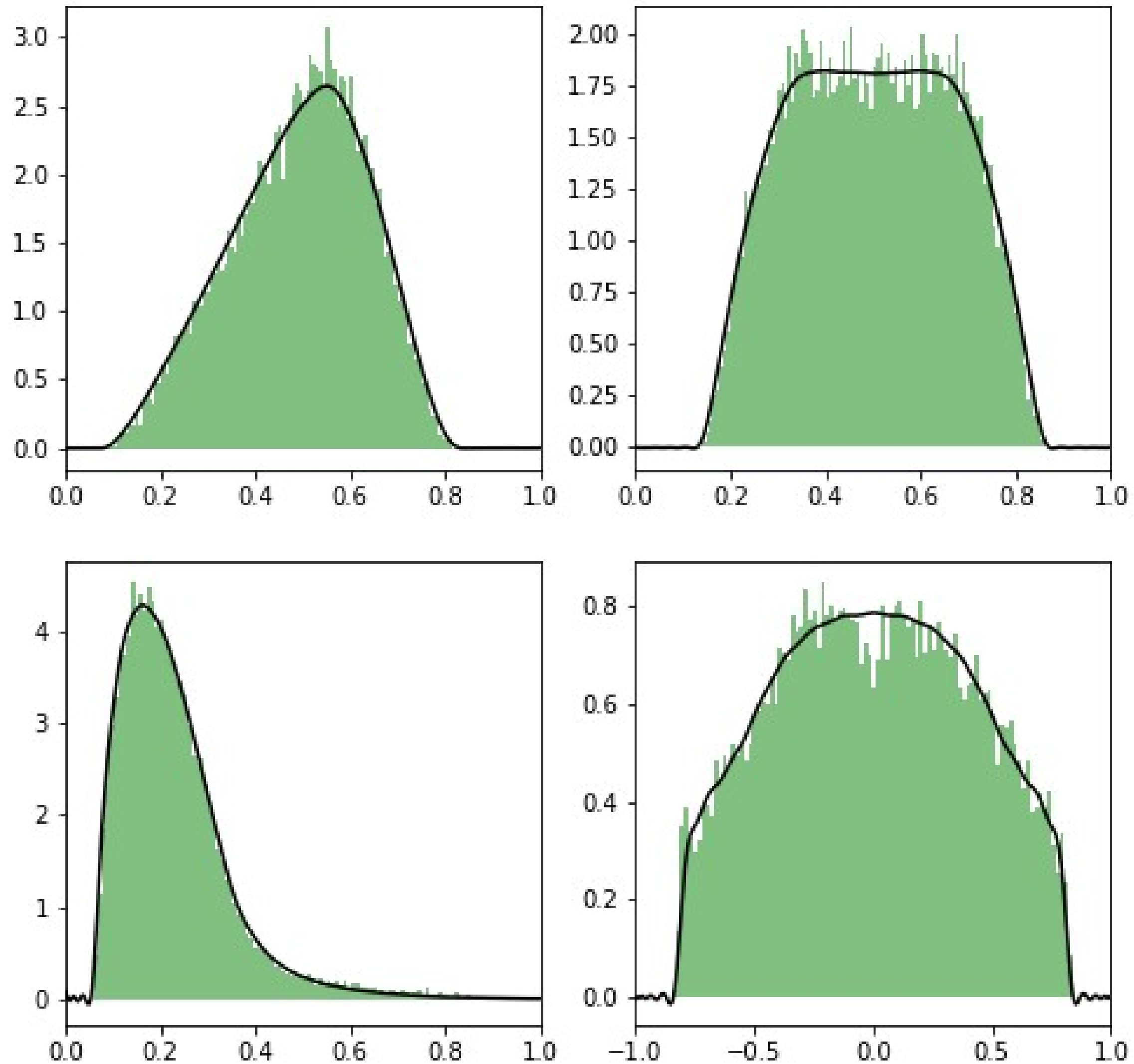
- Conditional distribution



- Fluctuations come from:
  - Truncation of the basis function
  - Limited statistics in the matrix element integration

Unweighted events

- Using the Gibbs sampling method we generate an unweighted sample:



Further work

- Lots to do:
  - Use more GPU
  - NLO/NNLO working example
  - Fix numerical issues
    - Polynomial basis chosen is not good with phase-space theta functions
    - Conditional probabilities not well constrained on phasespace boundaries
  - Hadronic example

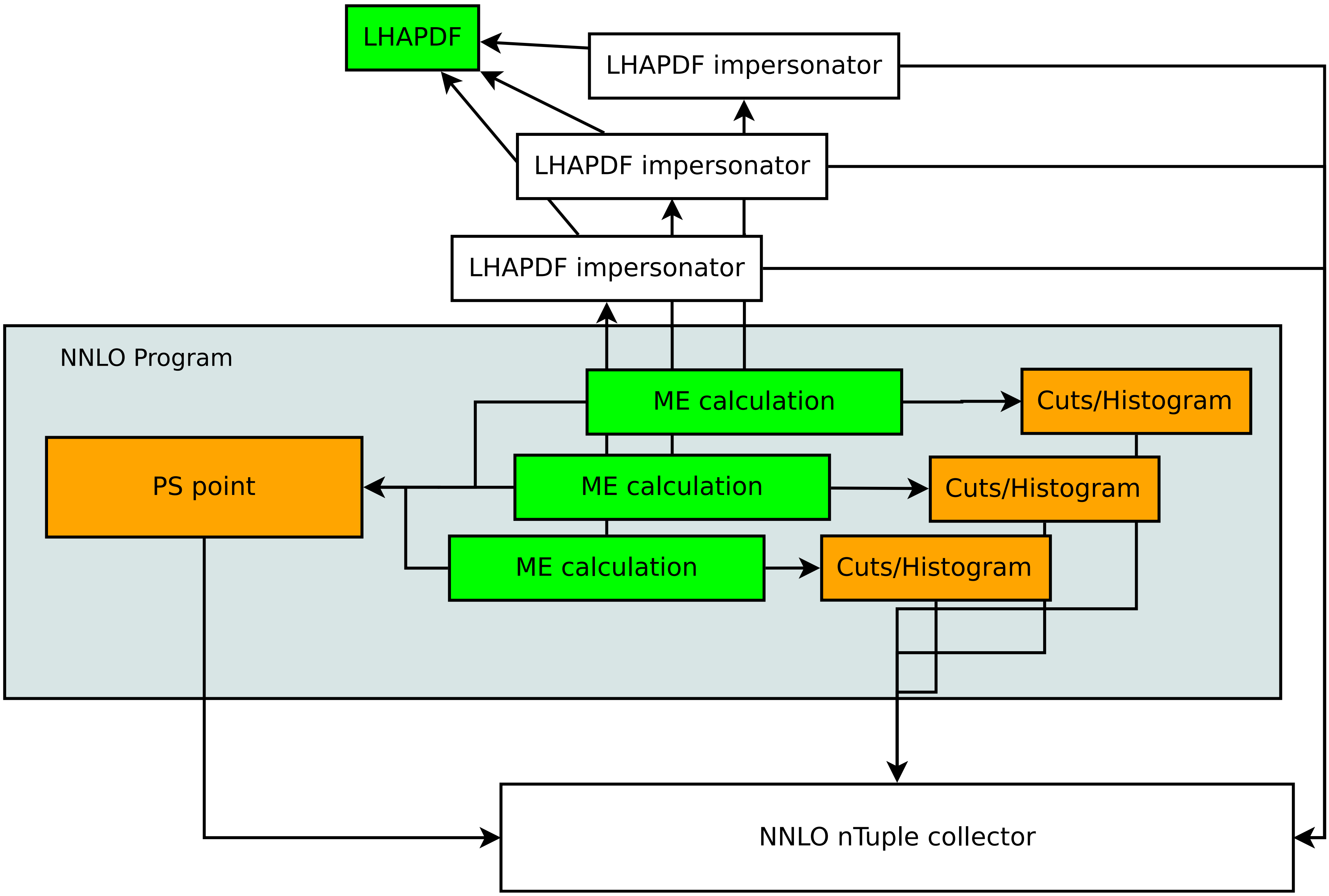
Backup

nTuples for NNLO

nTuples for NNLO

- nTuples have proven useful for NLO
- Can they be as useful for NNLO?
- Same advantages and same disadvantages but amplified:
  - Programs are more complex
  - Larger files:
    - Many more pieces in the calculation
    - More logarithm coefficients
- Main question: is the size reasonable?

Implementation



LHAPDF impersonator

- Allows to alternate vanishing and real pdf to disentangle different pdf terms
- Allows to filter specific initial state
- Reports pdf arguments to the nTuple collector
- Allows to set coupling constant values to one
- Is implemented using a hacking technique so there is no need to modify either
  - the NNLO program
  - the LHAPDF code

Additional difficulties

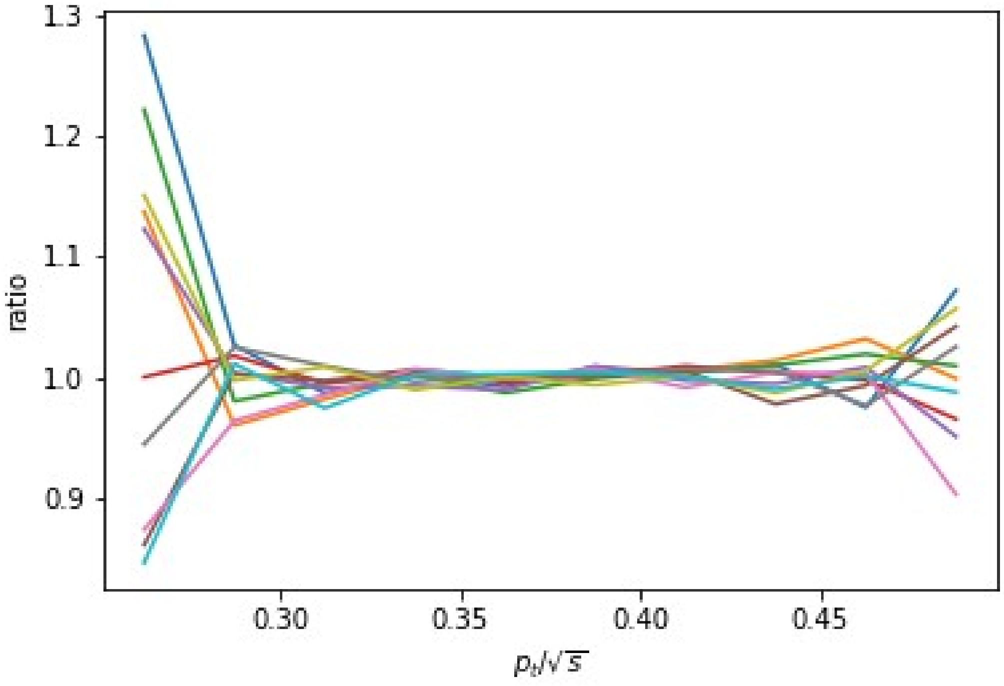
- Caching can cause confusion when the order of pdf, alphas, cuts and histograms are perturbed.
- Need to avoid grid adaptation to ensure synchronisation of the threads

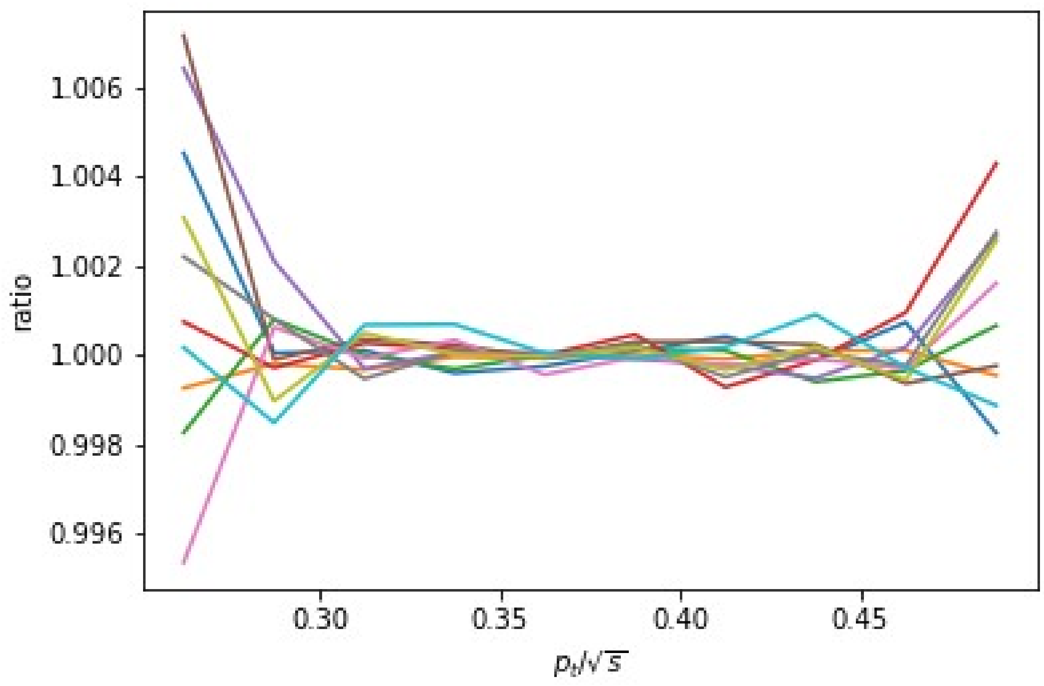
Error estimate

- Two sources of error:
  - Statistical uncertainty on the true value of the coefficients
    - Uncertainty in the matrix element and in the observable
    - This uncertainty can be estimated either
      - during the coefficient determination as a Monte-Carlo error
      - using sub-sampling techniques, with the advantage that correlations between errors are taken into account
  - Truncation error
    - Can be estimated by studying the stability of the prediction as a function of the depth of the expansion.

Coefficient uncertainties

- Observable coefficients
- Matrix elements coefficients





Truncation error

- We can estimate the truncation uncertainty by looking at the convergence as a function of the number of basis function