Road to “differentiable” Pythia?

+

Half-baked thoughts on energy conservation in hadronization

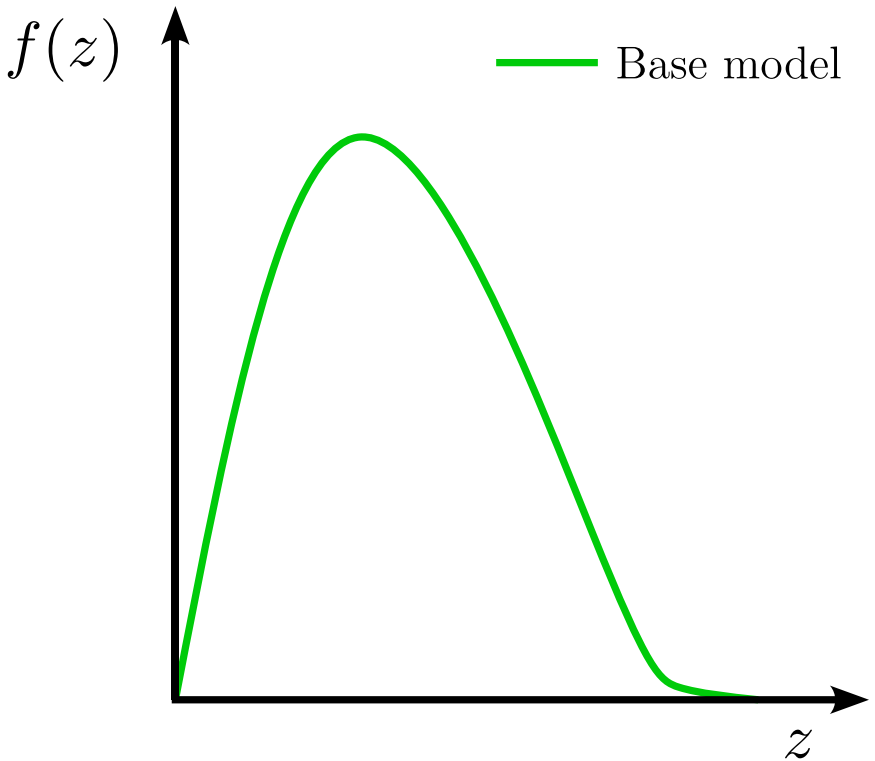
Pythia week @ Lund University
May 14th, 2025

Tony Menzo

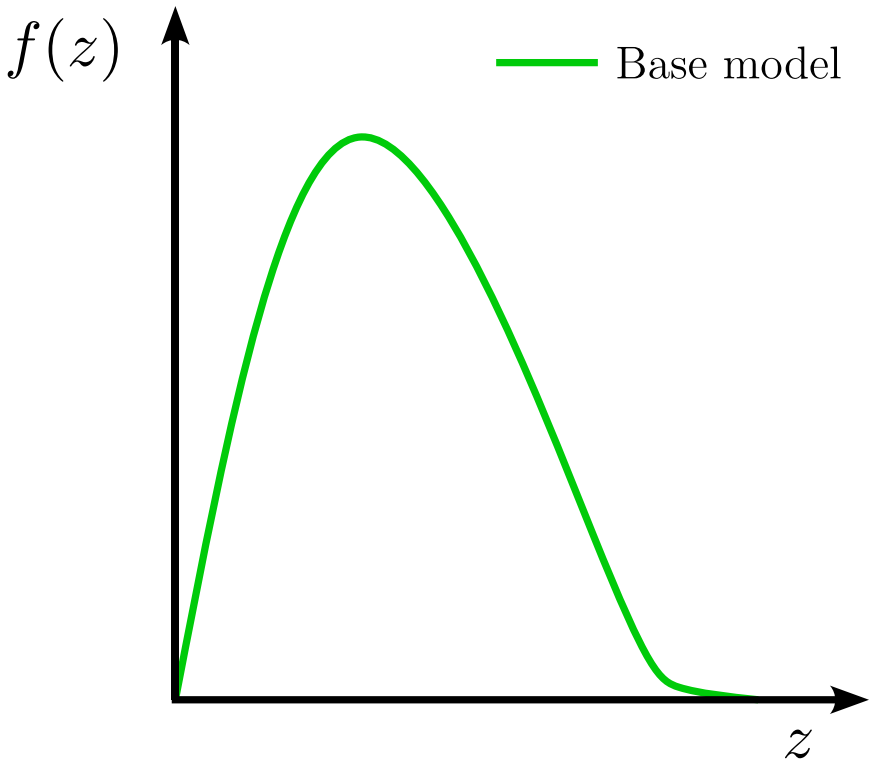
PhD candidate, University of Cincinnati

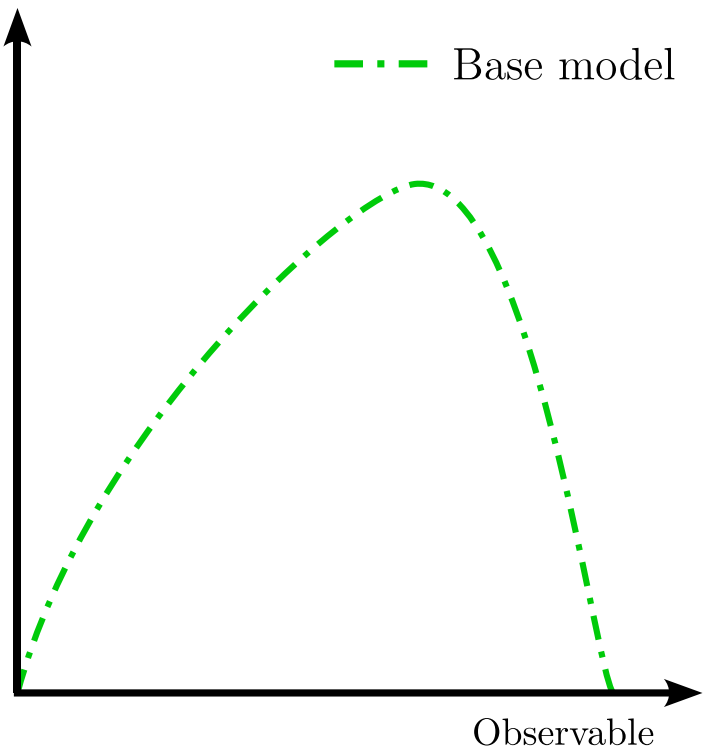
Based on work with Steve, Phil, Manuel, Christian + MLHAD + MLHADML

MLHAD efforts: big picture

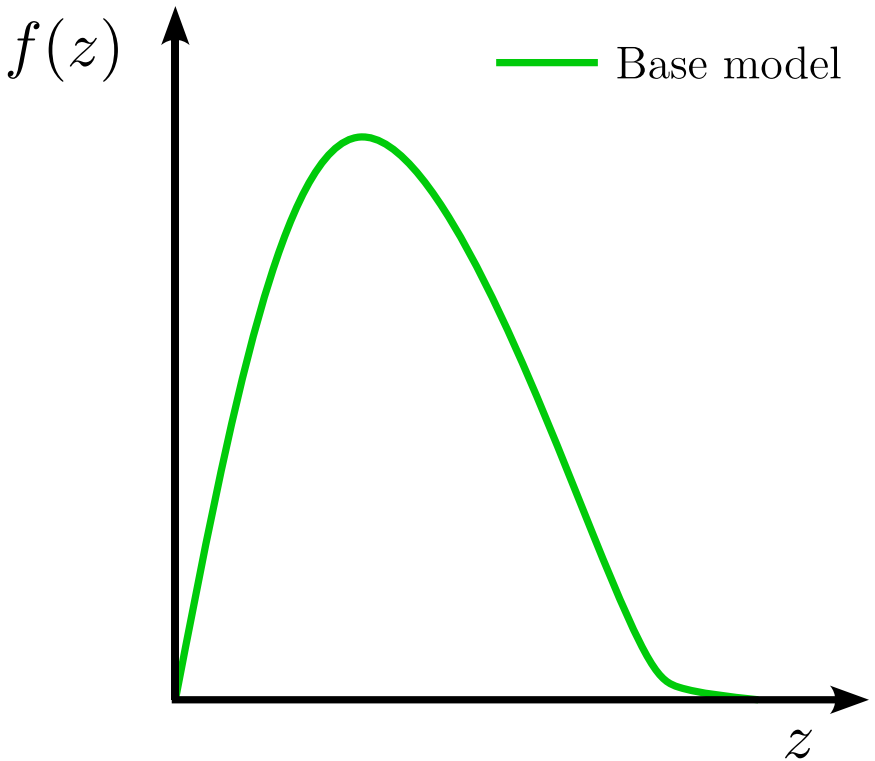


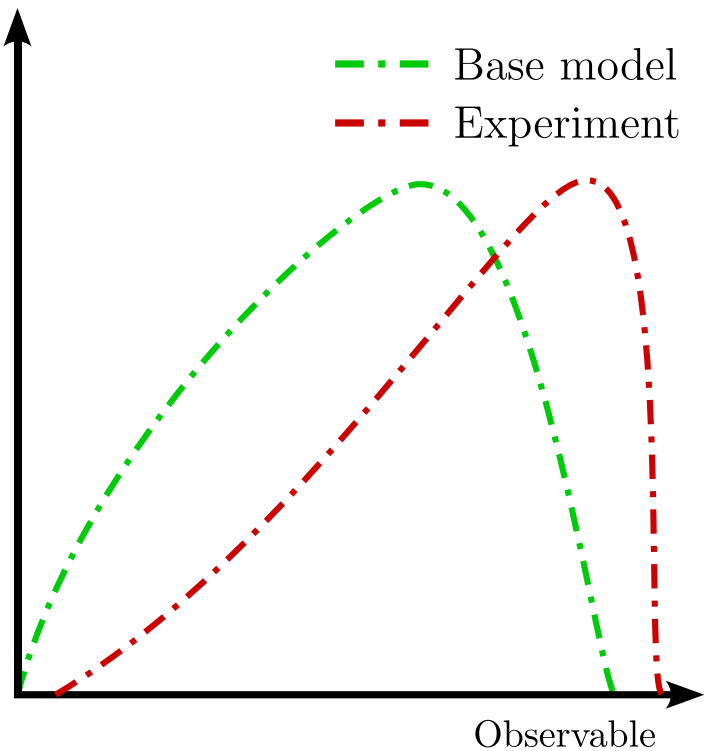
MLHAD efforts: big picture



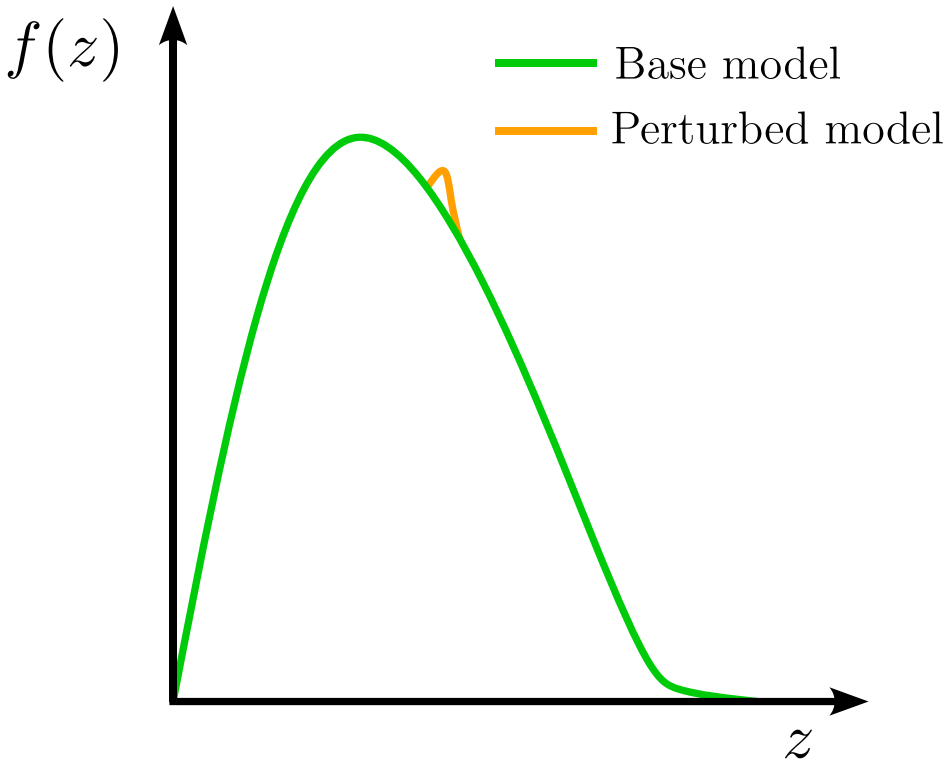


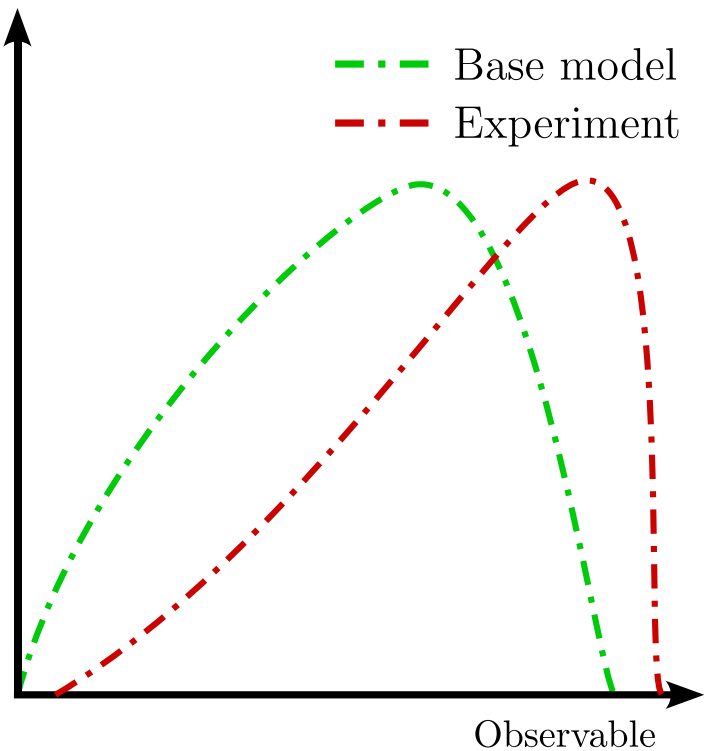
MLHAD efforts: big picture



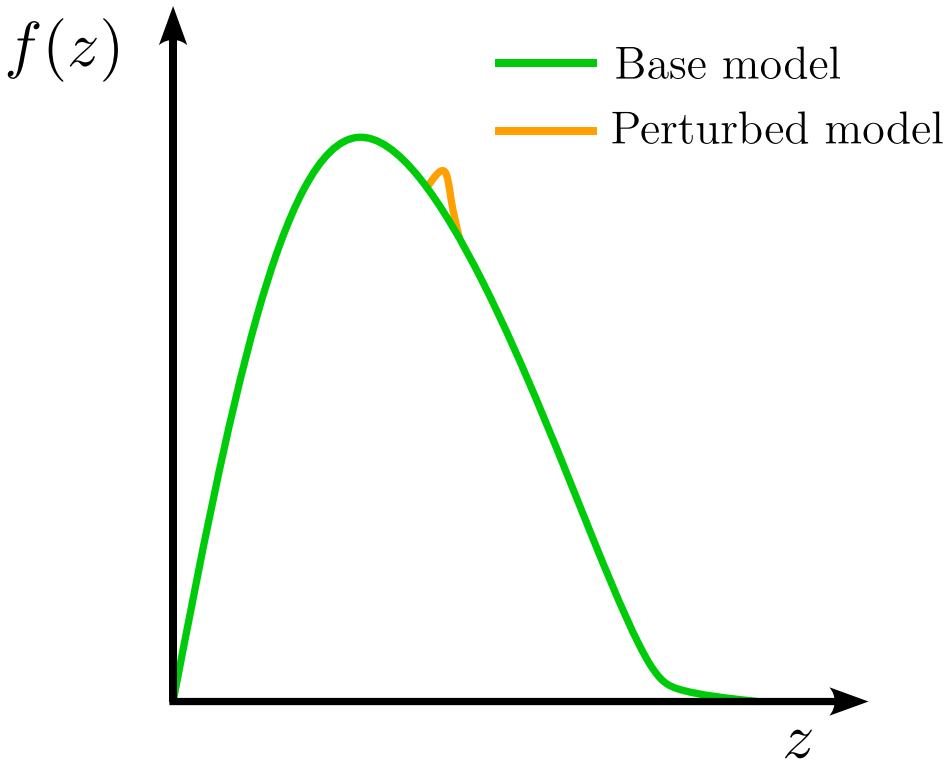


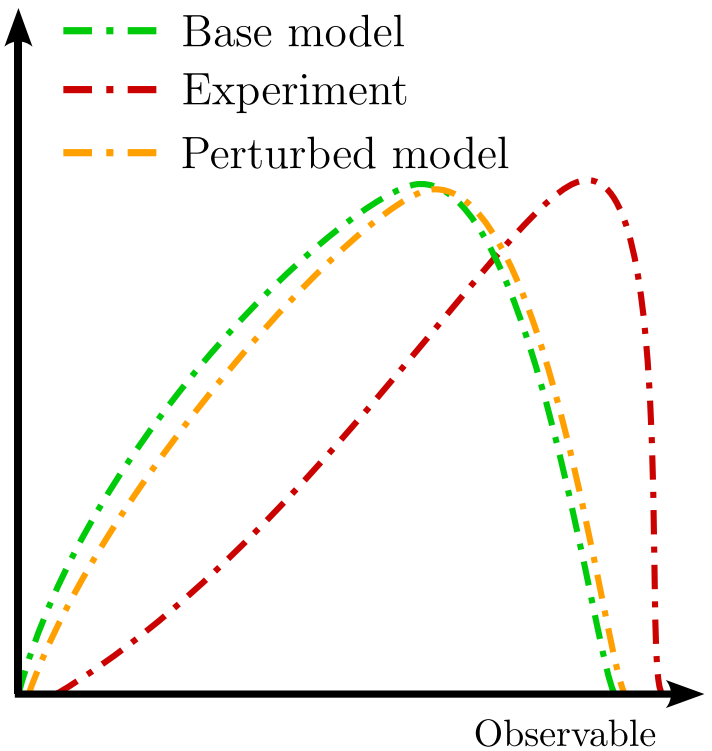
MLHAD efforts: big picture



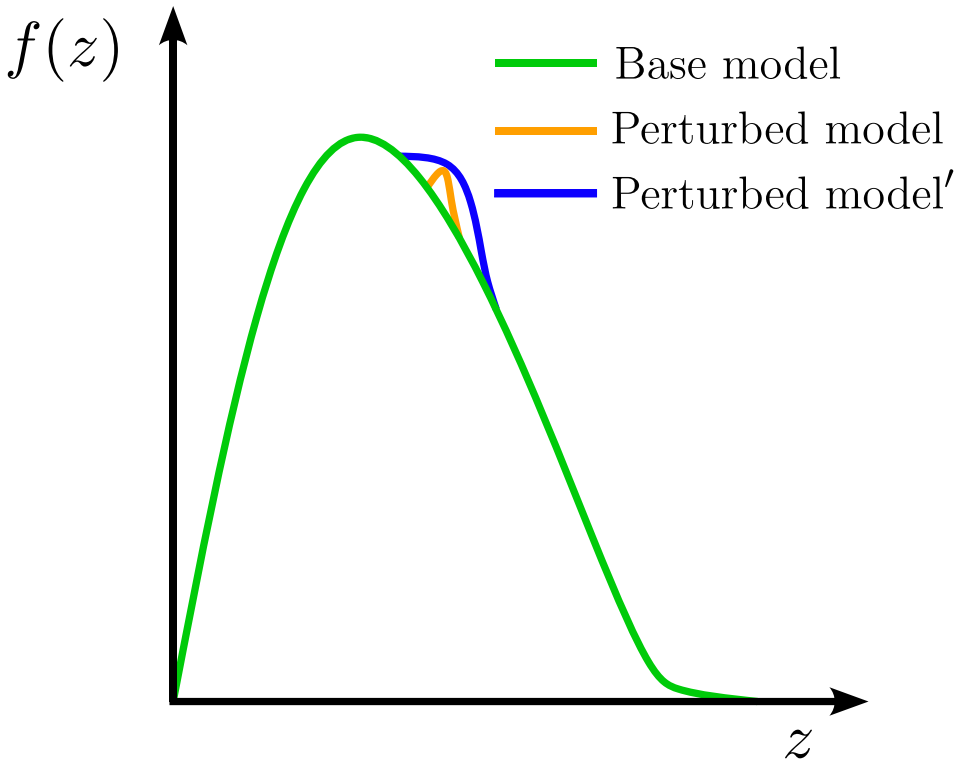


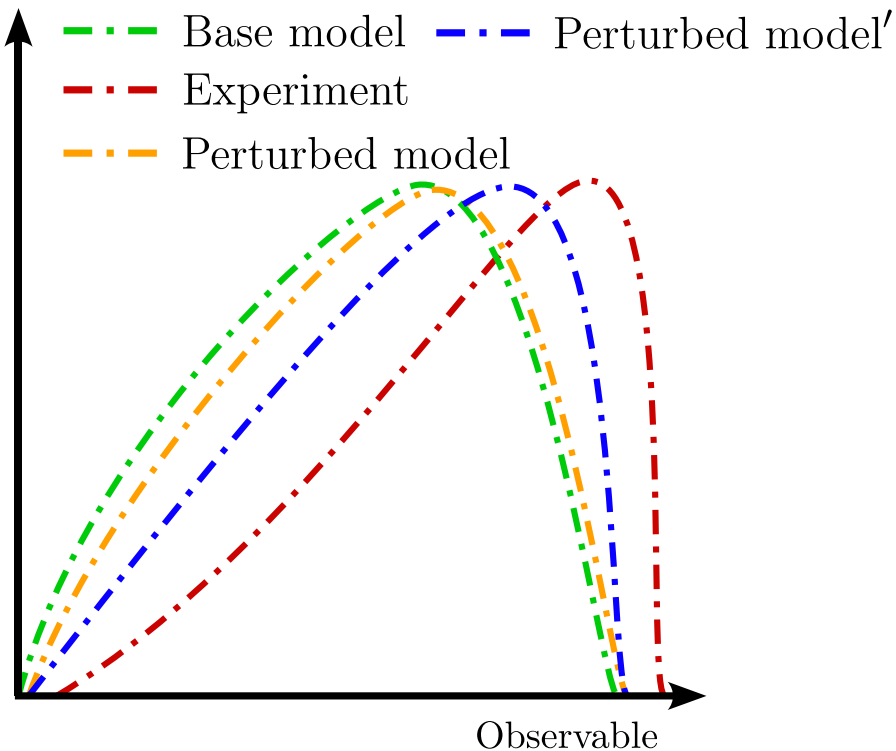
MLHAD efforts: big picture



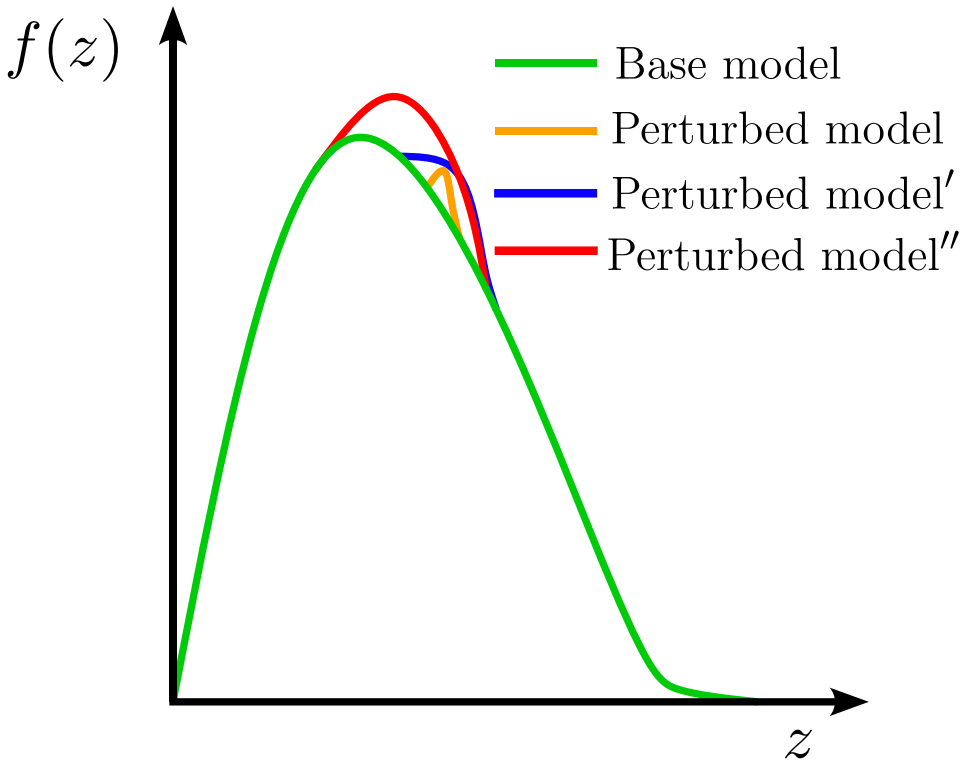


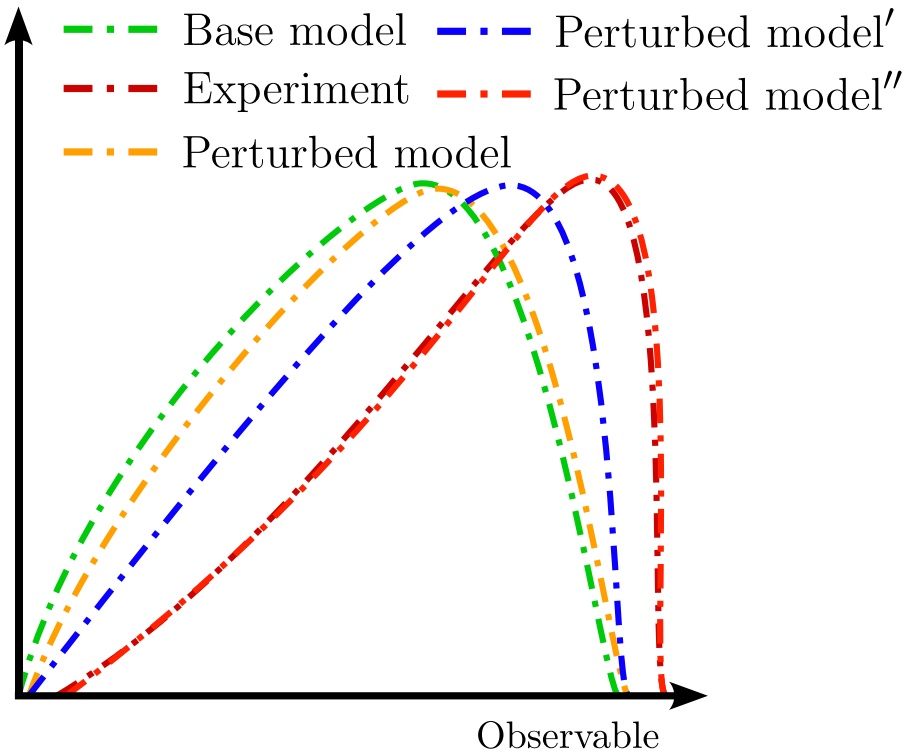
MLHAD efforts: big picture



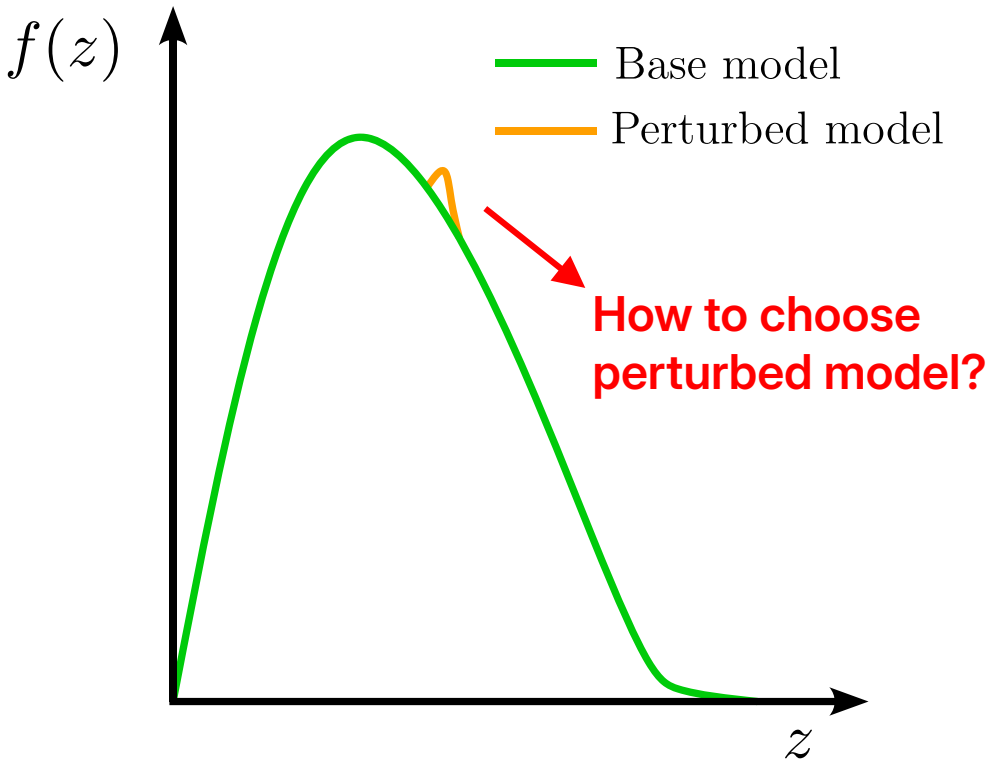


MLHAD efforts: big picture

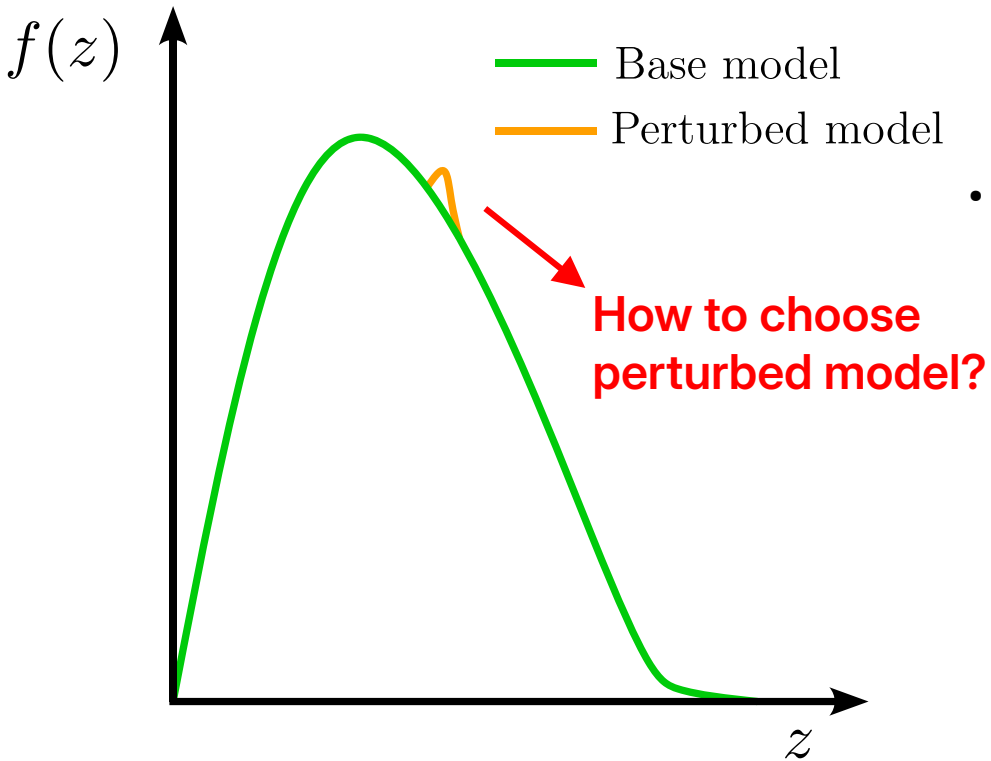




MLHAD efforts: big picture



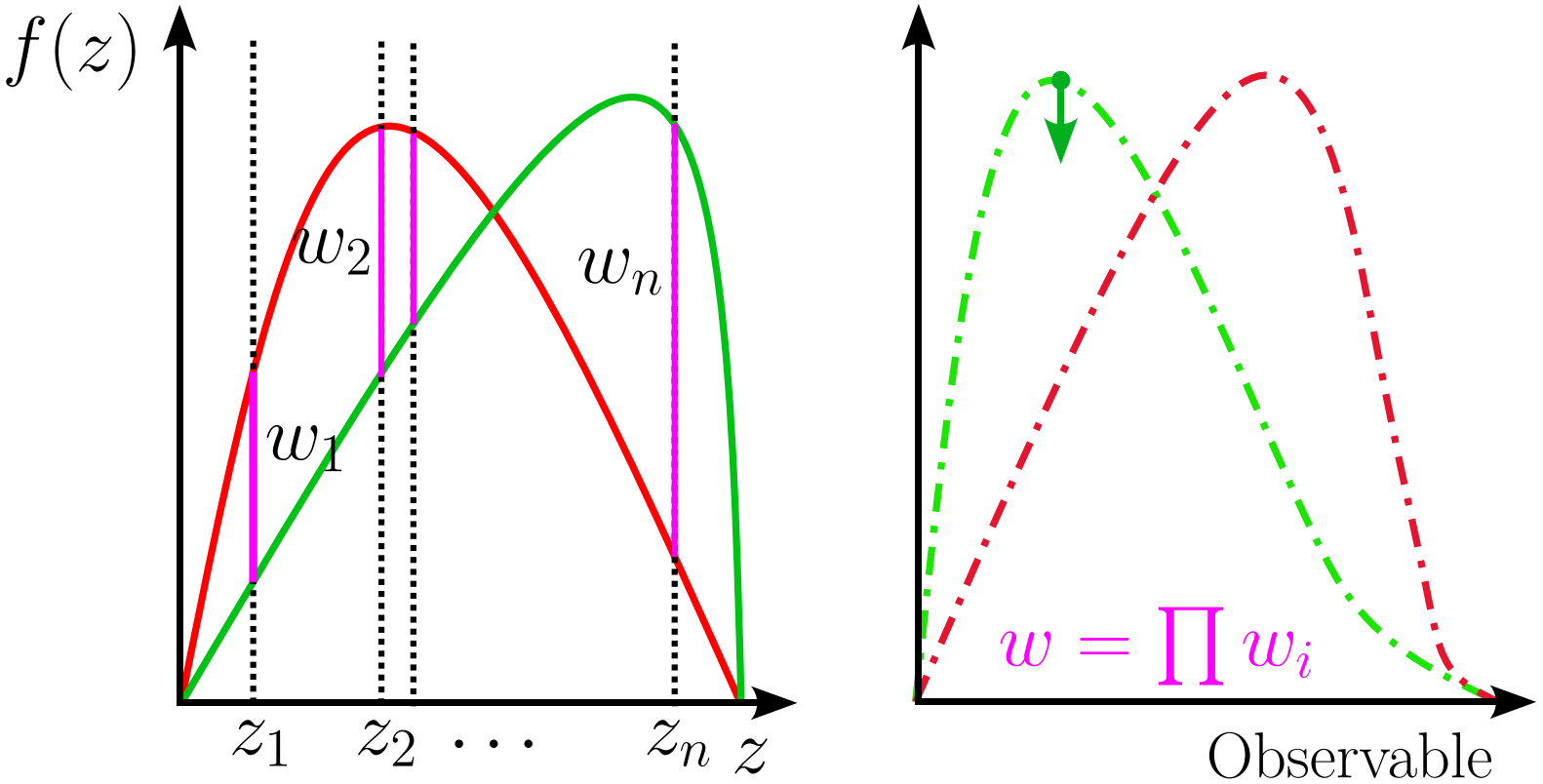
MLHAD efforts: big picture



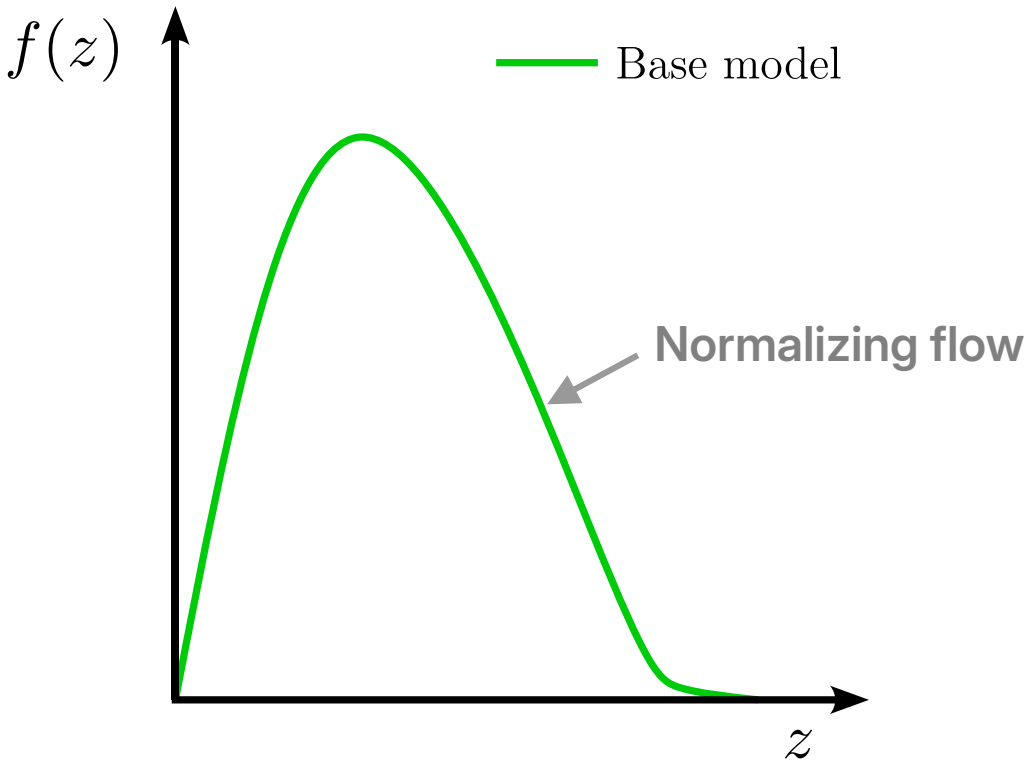
~~ML~~ vs ML

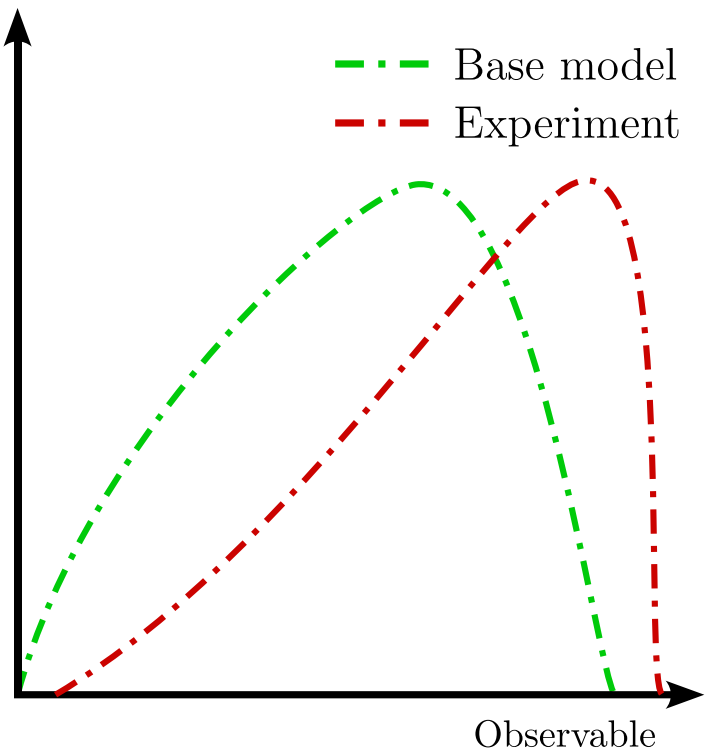
- Vary Lund parameters (traditional "manual"/semi-"manual" tuning)
- ML-based (data-driven) fragmentation function
  - MAGIC, HOMER
- Hybrid – keep Lund fragmentation function, promote Lund parameters to differentiable objects
  - Rejection sampling with autodifferentiation (RSA)

Aside - Reweighting

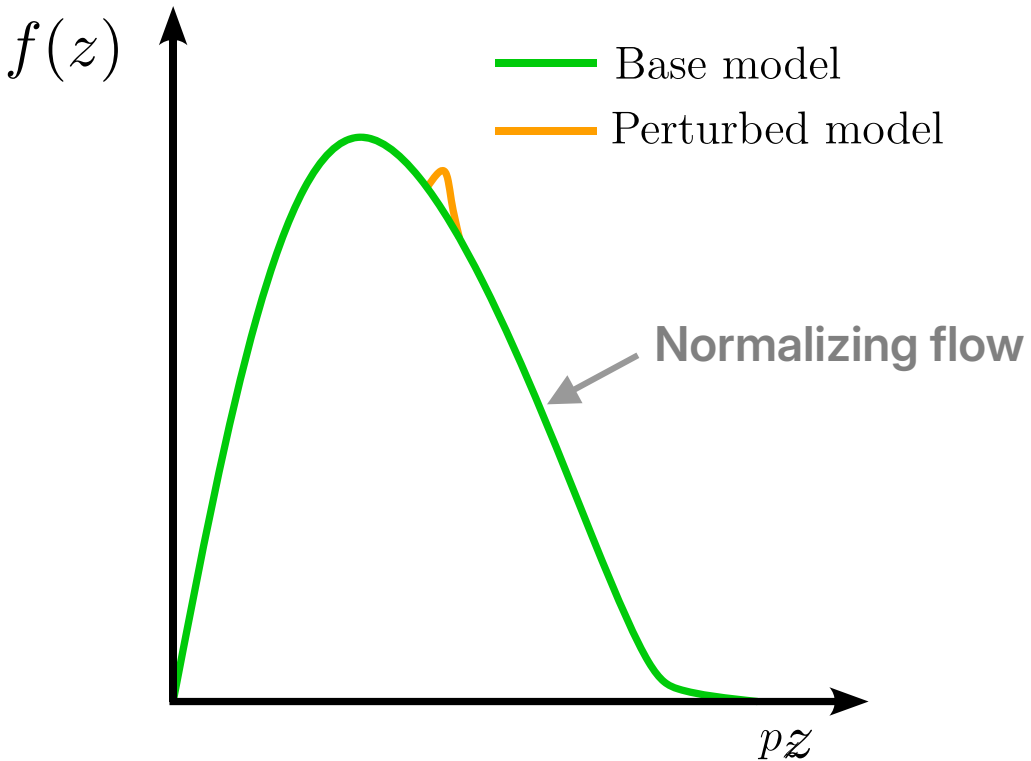


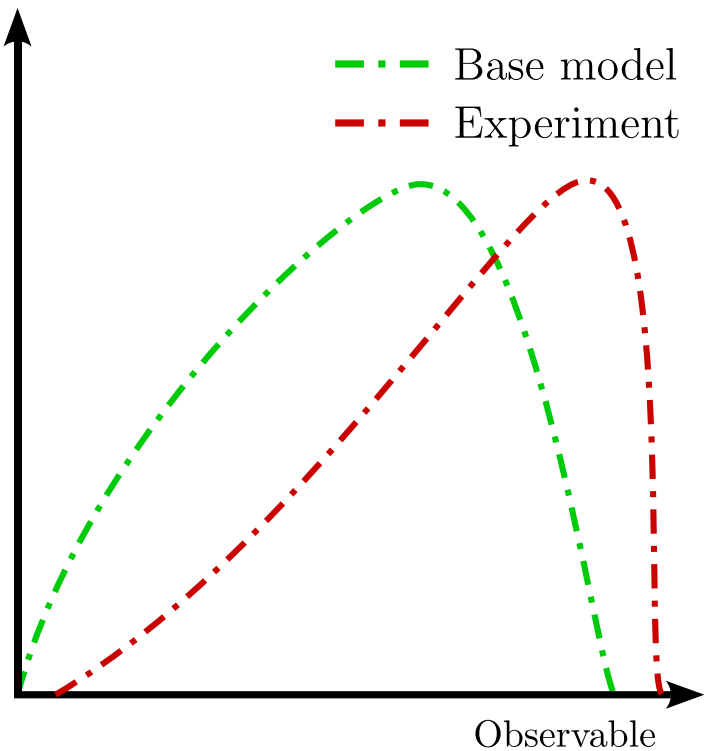
MLHAD efforts: MAGIC



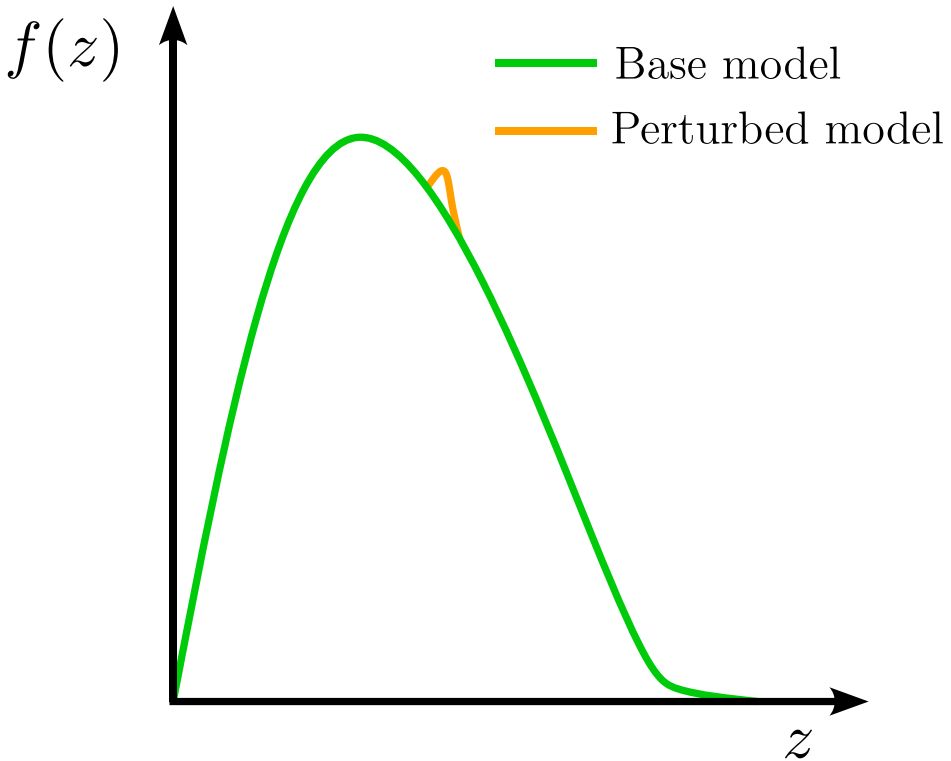


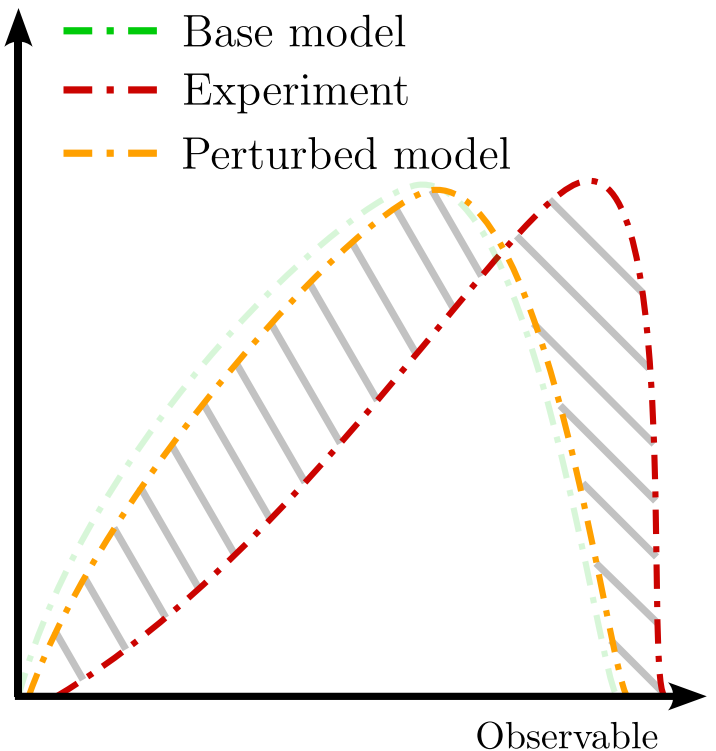
MLHAD efforts: MAGIC



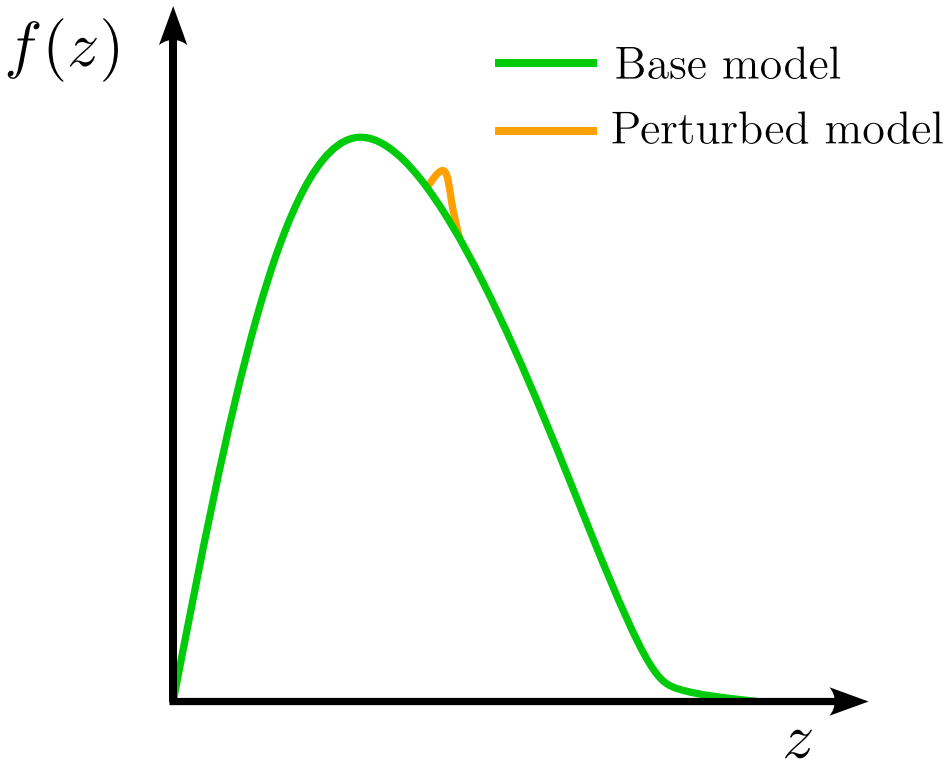


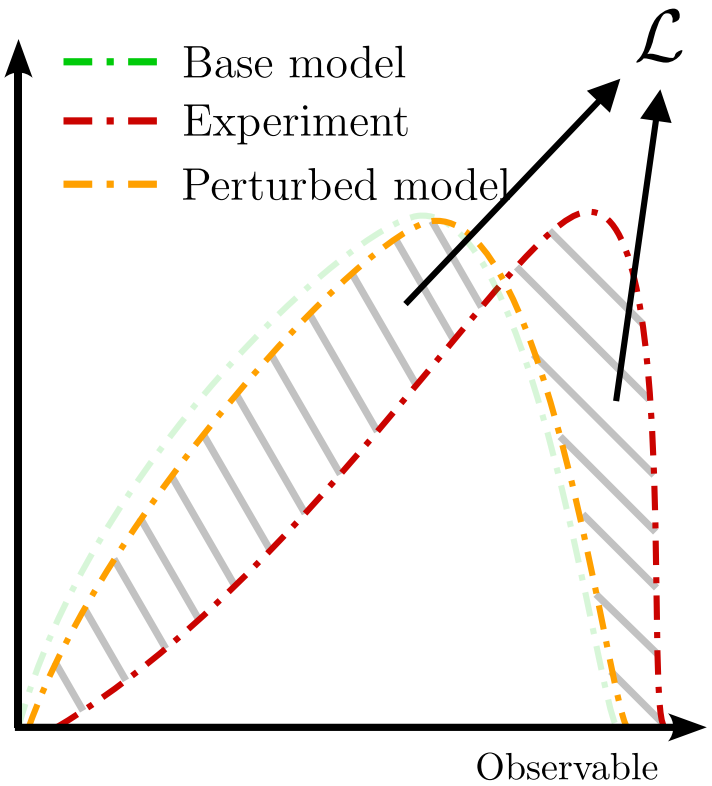
MLHAD efforts: MAGIC



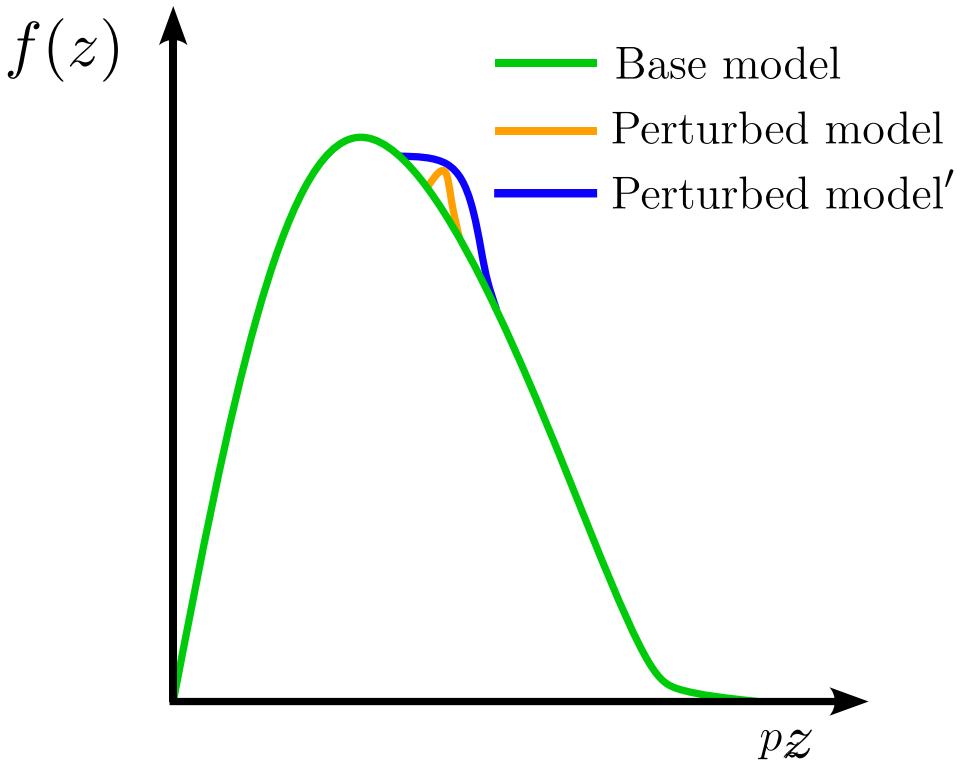


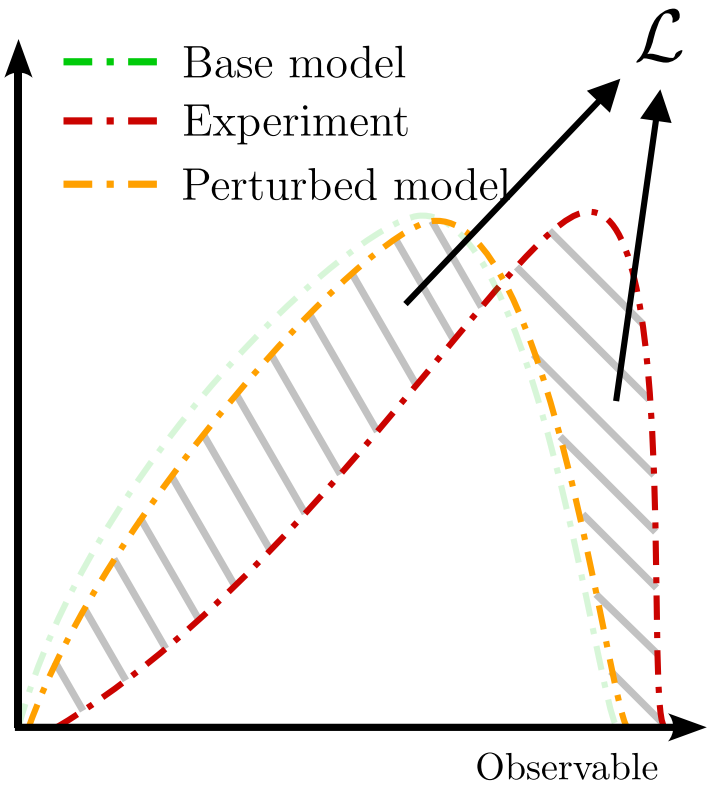
MLHAD efforts: MAGIC



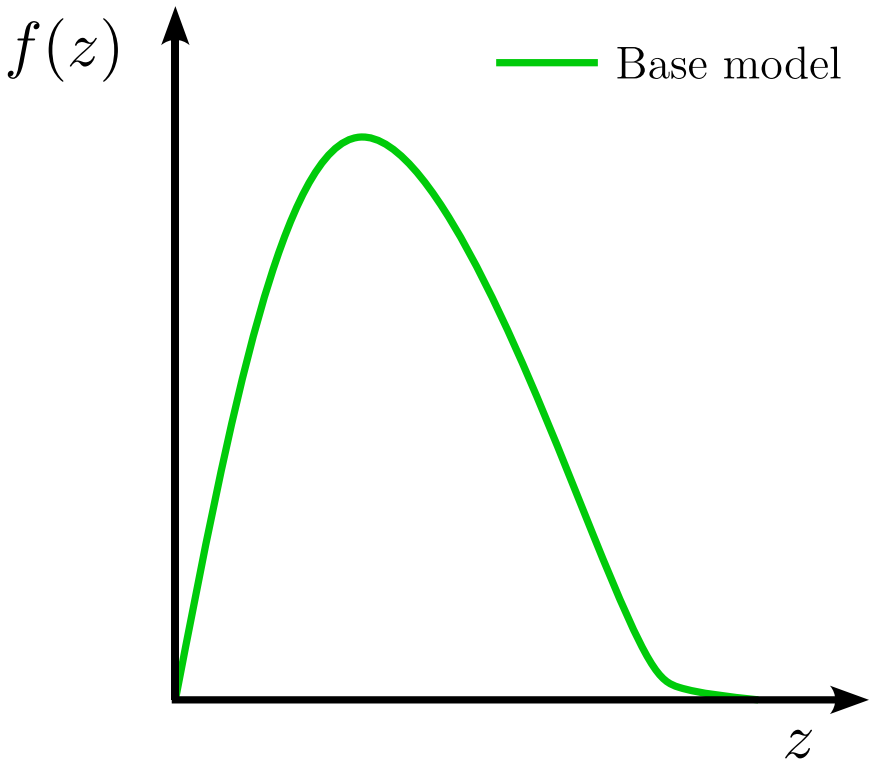


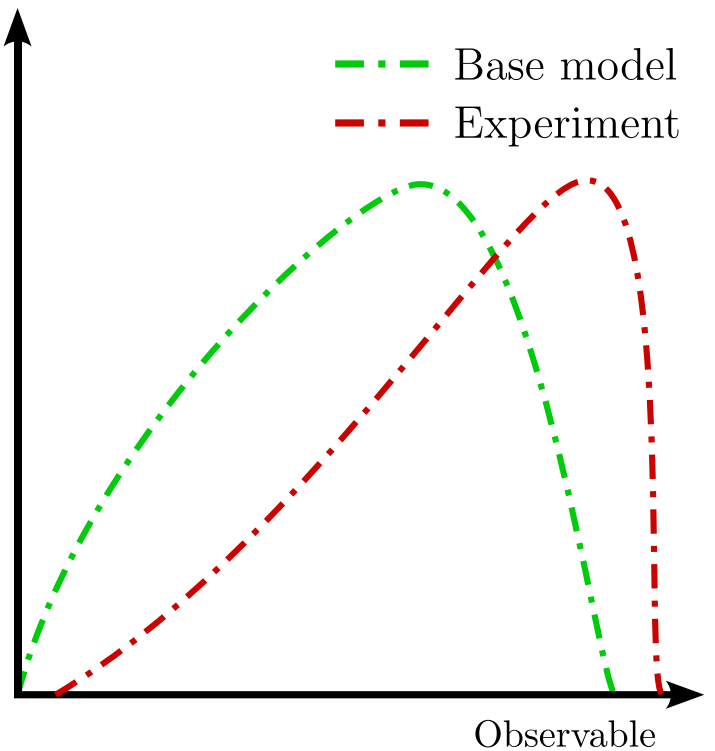
MLHAD efforts: MAGIC



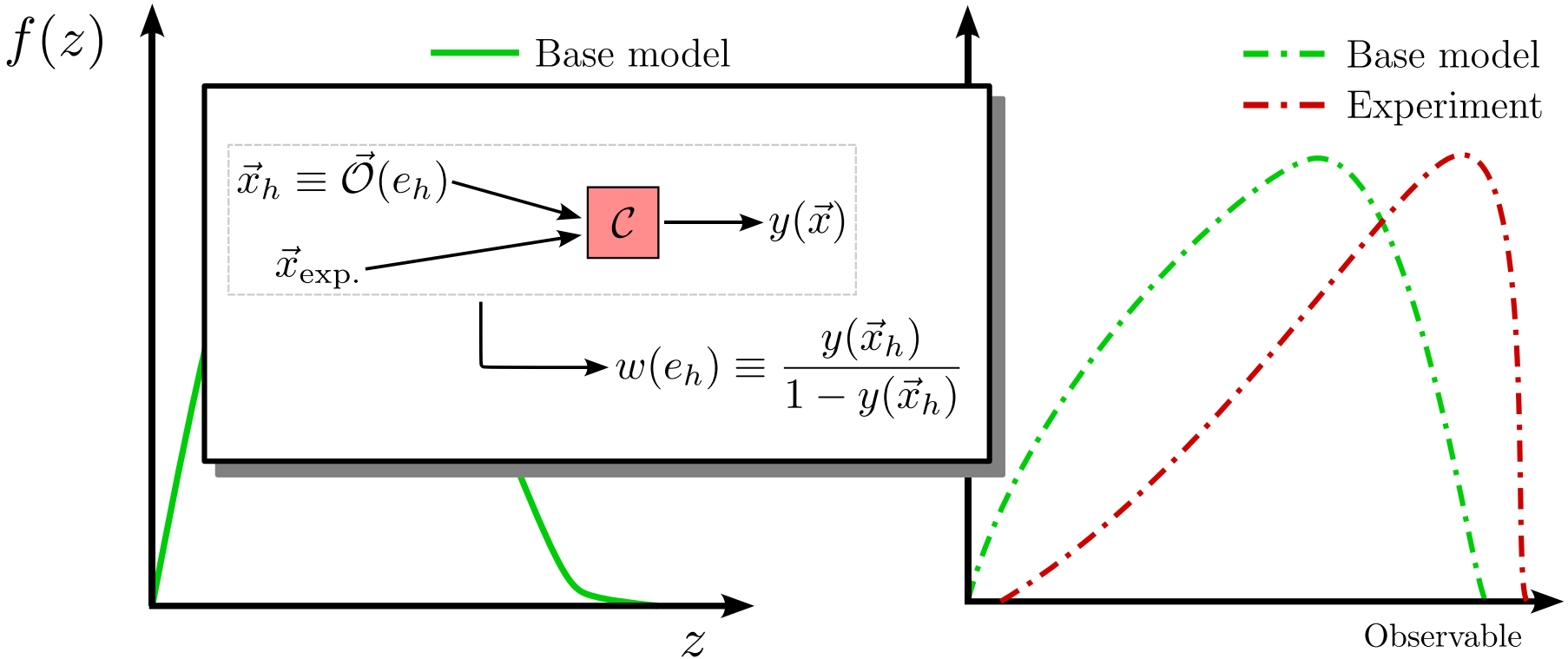


MLHAD efforts: HOMER

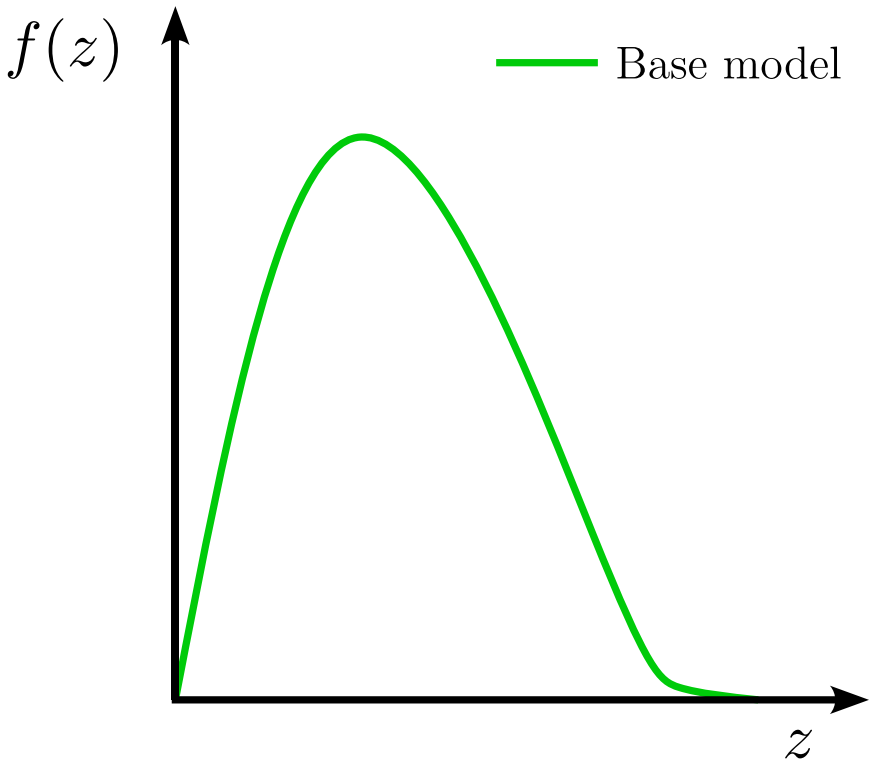


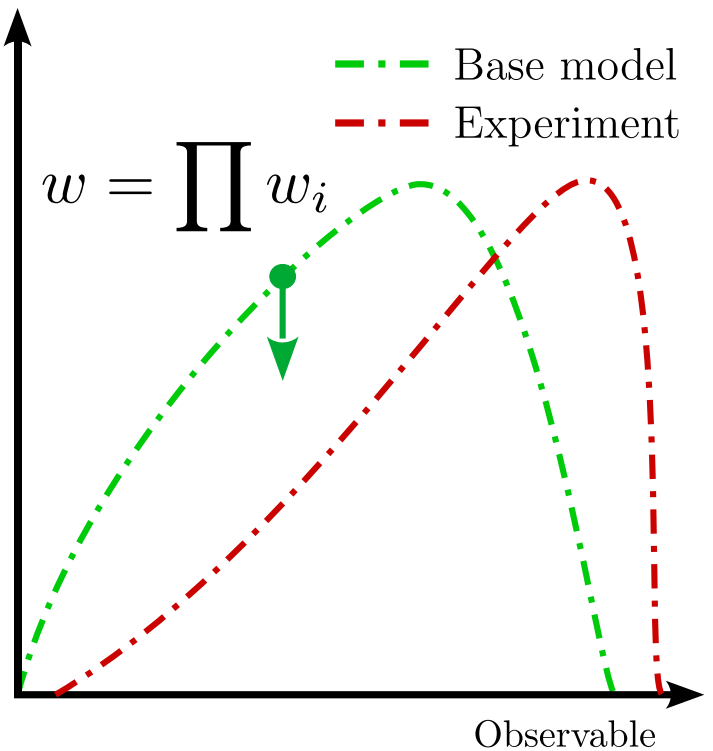


MLHAD efforts: HOMER

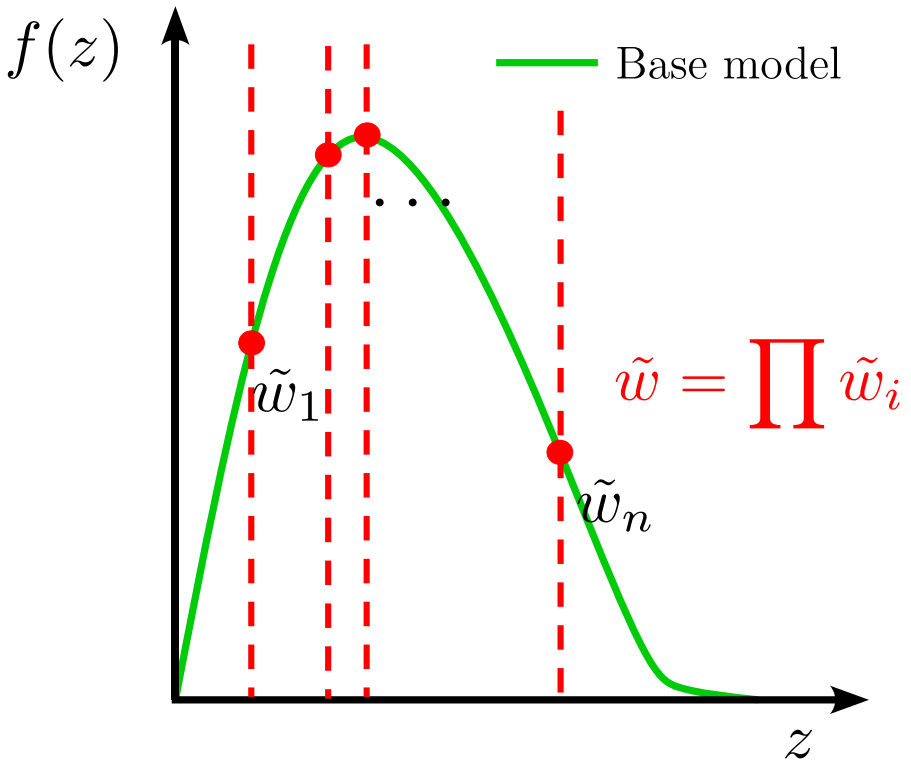


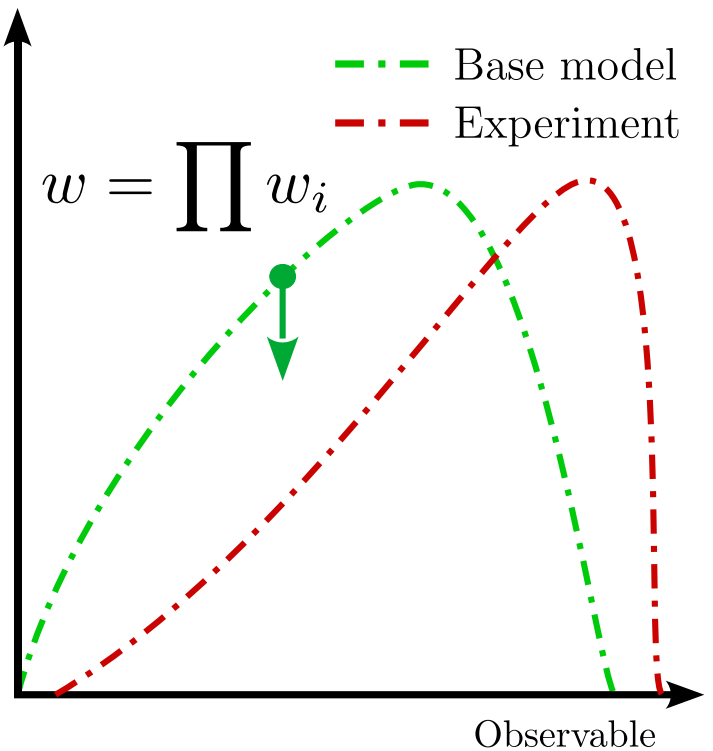
MLHAD efforts: HOMER



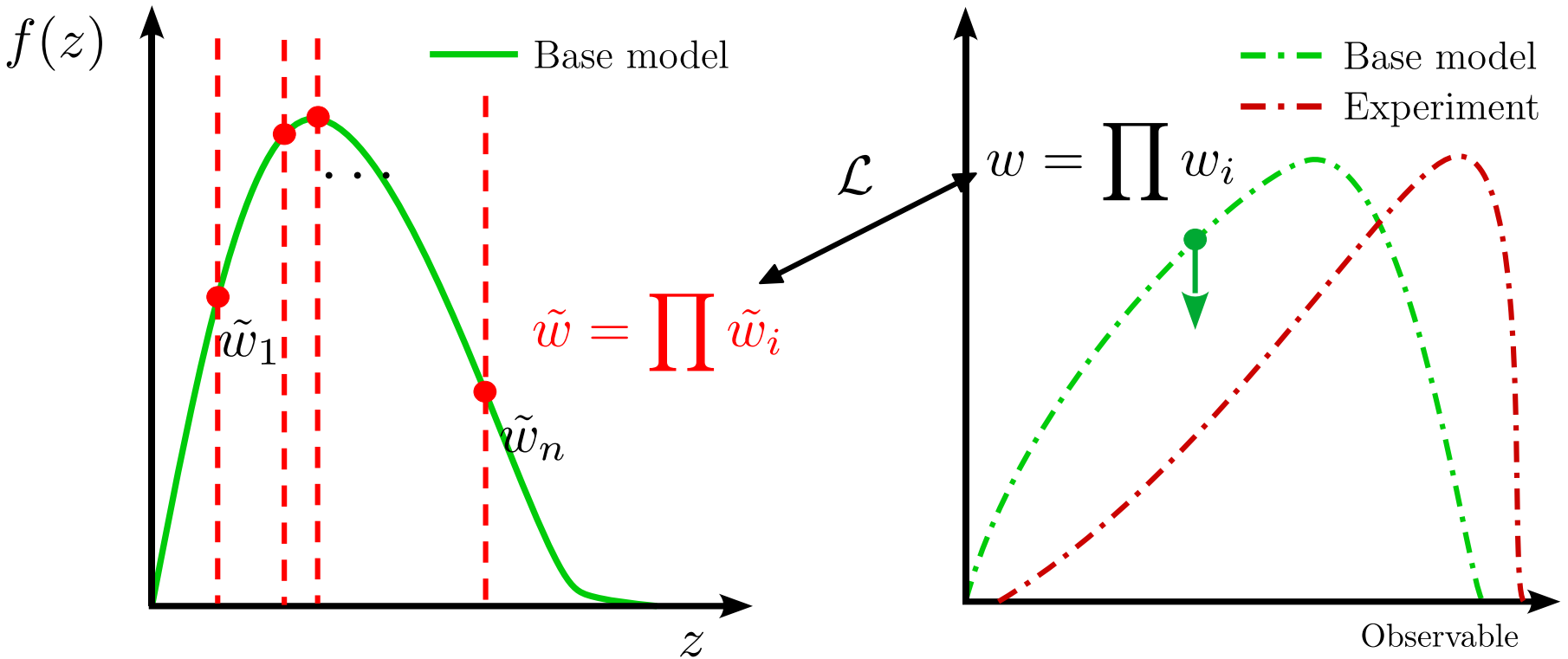


MLHAD efforts: HOMER

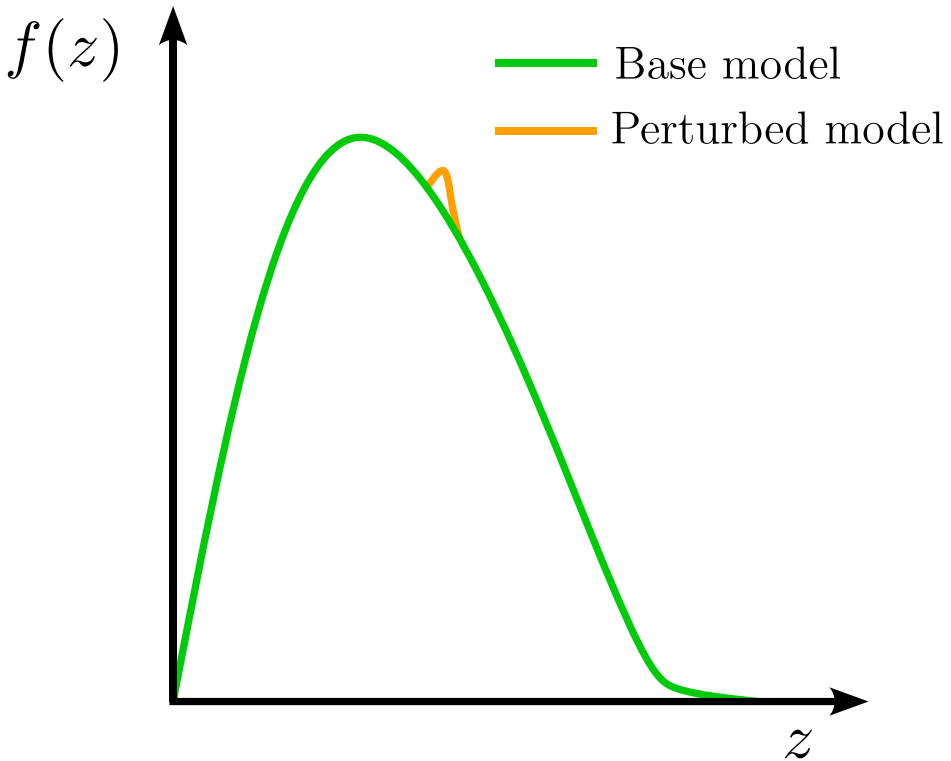


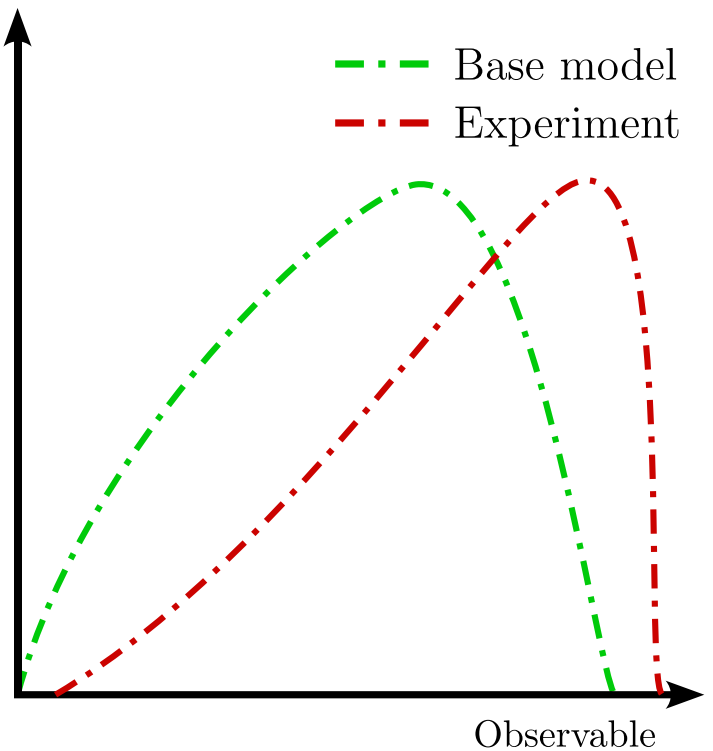


MLHAD efforts: HOMER

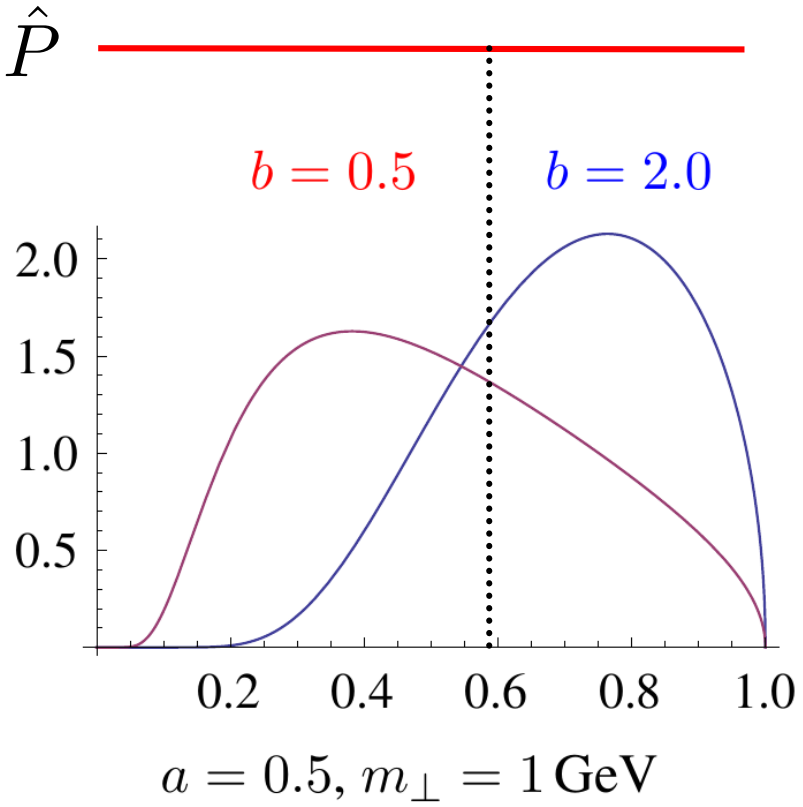


MLHAD efforts: HOMER





Kinematic reweighting (2308.13459)



Rejection sampling:

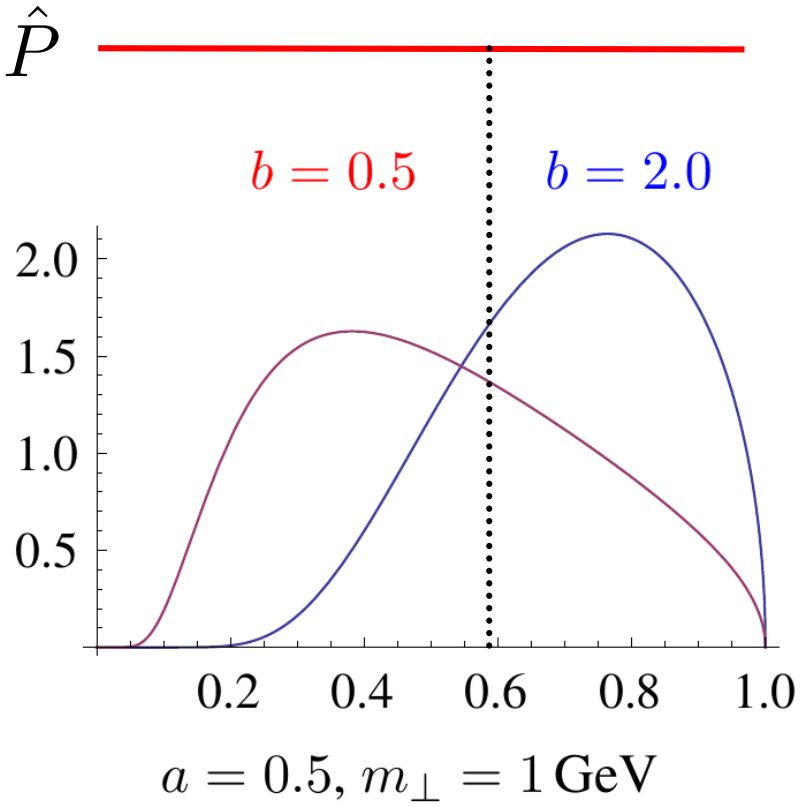
Acceptance probability: $P_{\text{accept}} = \dfrac{p(z,\theta)}{\hat{P}}$

Rejection probability: $P_{\text{reject}} = 1 - P_{\text{accept}}$

$$w_{\text{accept}} = \frac{p(z;\theta')}{p(z;\theta)}$$

$$w_{\text{reject}} = \frac{\hat{P} - p(z;\theta')}{\hat{P} - p(z;\theta)}$$

Kinematic reweighting (2308.13459)



Rejection sampling:

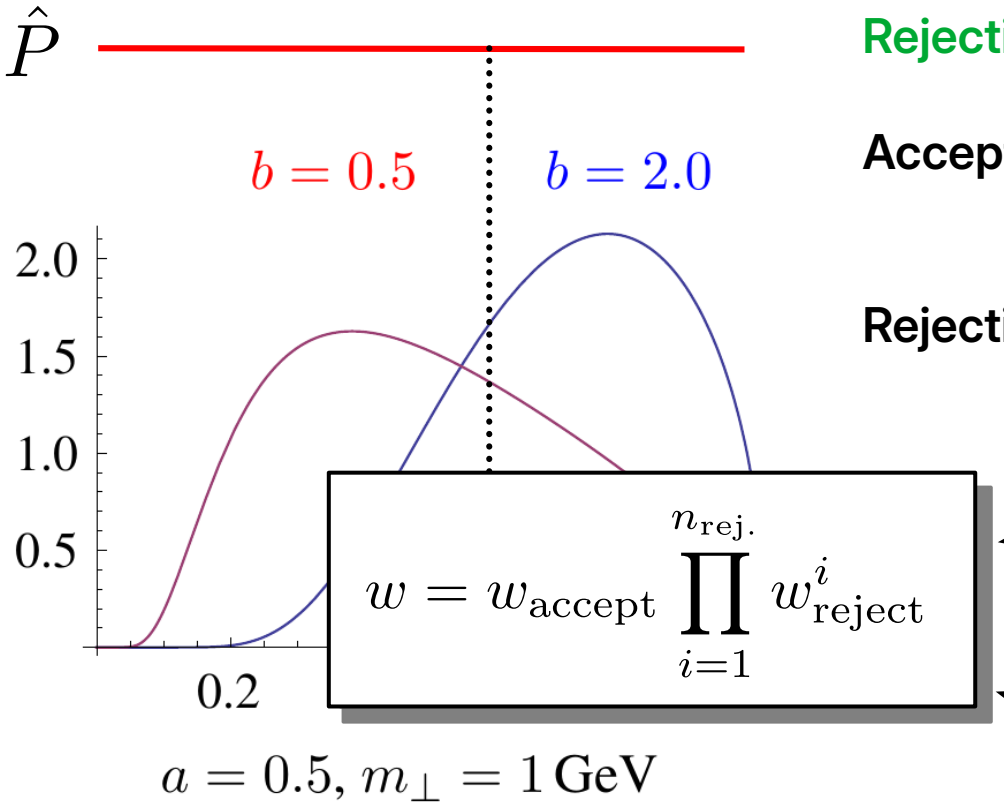
Acceptance probability: $P_{\text{accept}} = \dfrac{p(z,\theta)}{\hat{P}}$

Rejection probability: $P_{\text{reject}} = 1 - P_{\text{accept}}$

$$w_{\text{accept}} = \frac{P_{\text{accept}}(z;\theta')}{P_{\text{accept}}(z;\theta)}$$

$$w_{\text{reject}} = \frac{1 - P_{\text{accept}}(z;\theta')}{1 - P_{\text{accept}}(z;\theta)}$$

Kinematic reweighting (2308.13459)



Rejection sampling:

Acceptance probability: $P_{\mathrm{accept}} = \dfrac{p(z,\theta)}{\hat{P}}$

Rejection probability: $P_{\mathrm{reject}} = 1 - P_{\mathrm{accept}}$

$$w = w_{\mathrm{accept}} \prod_{i=1}^{n_{\mathrm{rej.}}} w^i_{\mathrm{reject}}$$

$$w_{\mathrm{accept}} = \frac{p(z;\theta')}{p(z;\theta)}$$

$$w_{\mathrm{reject}} = \frac{\hat{P} - p(z;\theta')}{\hat{P} - p(z;\theta)}$$

Kinematic reweighting (2308.13459)

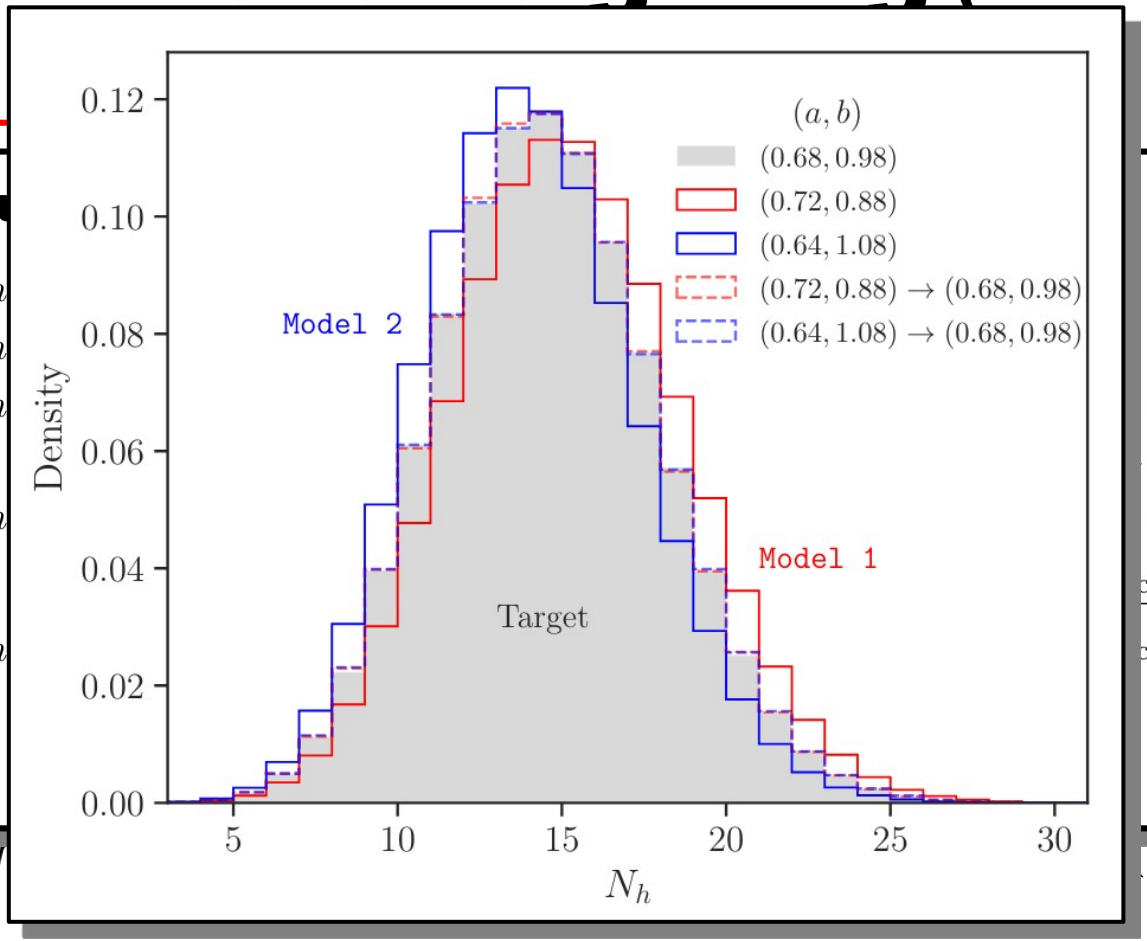
$\hat{P}$ ——————— Rejection sampling:

Data-structure:

$$z = \begin{pmatrix} z_1 = \begin{pmatrix} \{m_T^{h_1}, z_{\text{accept}}^{h_1}, z_{\text{reject}}^{1,h_1}, \cdots, z_{\text{reject}}^{n_{h_1},h_1}\} \\ \{m_T^{h_2}, z_{\text{accept}}^{h_2}, z_{\text{reject}}^{1,h_2}, \cdots, z_{\text{reject}}^{n_{h_2},h_2}\} \\ \{m_T^{h_3}, z_{\text{accept}}^{h_3}, z_{\text{reject}}^{1,h_3}, \cdots, z_{\text{reject}}^{n_{h_3},h_3}\} \end{pmatrix}_1 \\ z_2 = \begin{pmatrix} \{m_T^{h_1}, z_{\text{accept}}^{h_1}, z_{\text{reject}}^{1,h_1}, \cdots, z_{\text{reject}}^{n_{h_1},h_1}\} \\ \vdots \\ \{m_T^{h_4}, z_{\text{accept}}^{h_4}, z_{\text{reject}}^{1,h_4}, \cdots, z_{\text{reject}}^{n_{h_4},h_4}\} \end{pmatrix}_2 \\ \vdots \\ z_N = \cdots \end{pmatrix}$$

$$w_n = \prod_{i=1}^{\tilde{N}_{h,n}} \left(\frac{f(z_{\text{accept}}^{h_i}; \{a,b\}_P)}{f(z_{\text{accept}}^{h_i}; \{a,b\}_B)} \right) \times \prod_{j=1}^{n_{h_i}} \left(\frac{\hat{f} - f(z_{\text{reject}}^{j,h_i}; \{a,b\}_P)}{\hat{f} - f(z_{\text{reject}}^{j,h_i}; \{a,b\}_B)} \right)$$

Kinematic reweighting (2308.13459)



Kinematic reweighting (2308.13459)

$\hat{P}$

Rejection sampling:

Data-structure:

$$z = \begin{pmatrix} z_1 = \begin{pmatrix} \{m_T^{h_1}, z_{\text{accept}}^{h_1}, z_{\text{reject}}^{1,h_1}, \cdots, z_{\text{reject}}^{n_{h_1},h_1}\} \\ \{m_T^{h_2}, z_{\text{accept}}^{h_2}, z_{\text{reject}}^{1,h_2}, \cdots, z_{\text{reject}}^{n_{h_2},h_2}\} \\ \{m_T^{h_3}, z_{\text{accept}}^{h_3}, z_{\text{reject}}^{1,h_3}, \cdots, z_{\text{reject}}^{n_{h_3},h_3}\} \end{pmatrix}_1 \\ z_2 = \begin{pmatrix} \{m_T^{h_1}, z_{\text{accept}}^{h_1}, z_{\text{reject}}^{1,h_1}, \cdots, z_{\text{reject}}^{n_{h_1},h_1}\} \\ \vdots \\ \{m_T^{h_4}, z_{\text{accept}}^{h_4}, z_{\text{reject}}^{1,h_4}, \cdots, z_{\text{reject}}^{n_{h_4},h_4}\} \end{pmatrix}_2 \\ \vdots \\ z_N = \cdots \end{pmatrix}$$

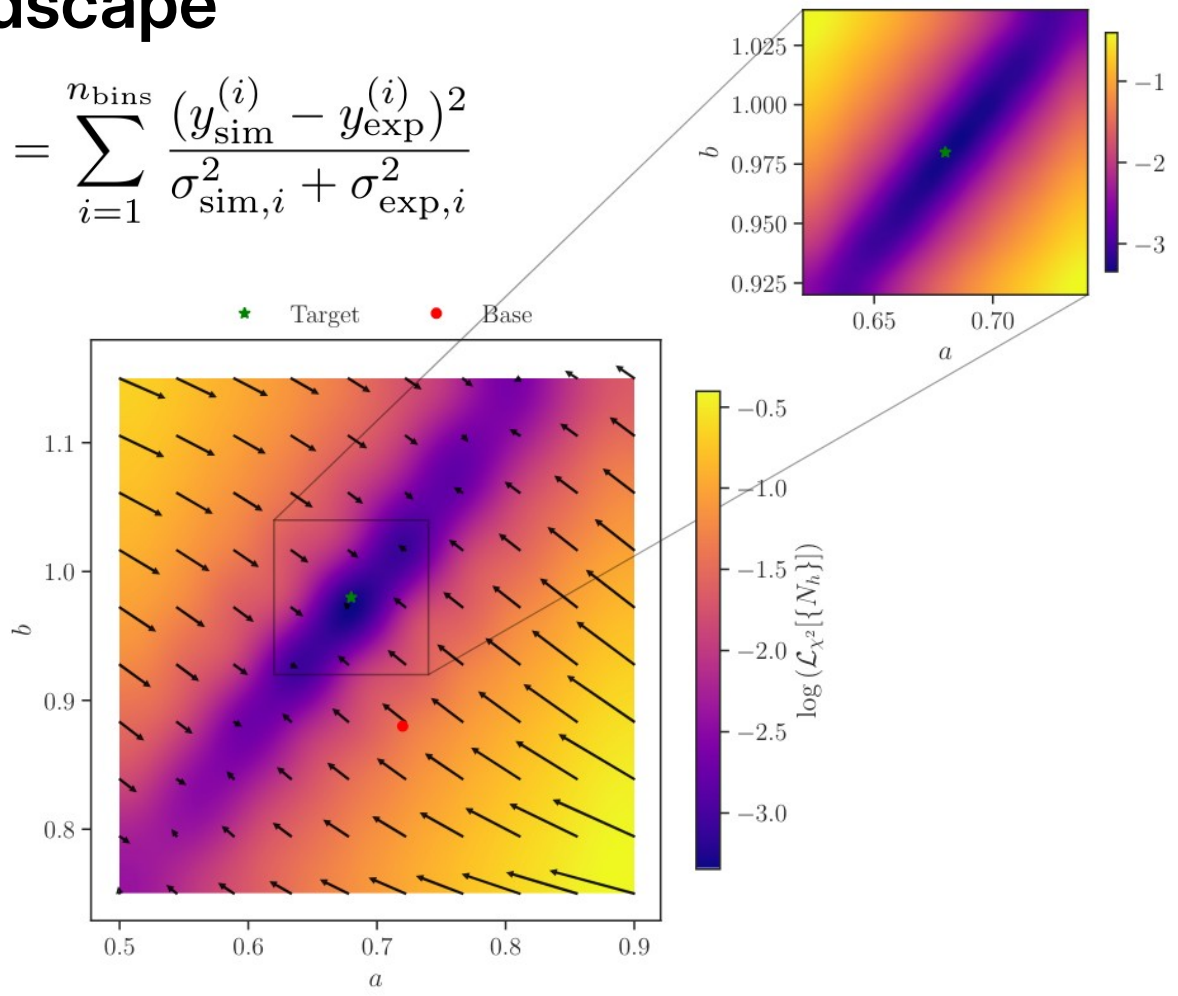
Differentiable! → RSA

$$w_n = \prod_{i=1}^{\tilde{N}_{h,n}} \left(\frac{f(z_{\text{accept}}^{h_i}; \{a,b\}_P)}{f(z_{\text{accept}}^{h_i}; \{a,b\}_B)} \right) \times \prod_{j=1}^{n_{h_i}} \left(\frac{\hat{f} - f(z_{\text{reject}}^{j,h_i}; \{a,b\}_P)}{\hat{f} - f(z_{\text{reject}}^{j,h_i}; \{a,b\}_B)} \right)$$

$a = 0.5, m_\perp = 1\,\text{GeV}$

χ^2-loss landscape

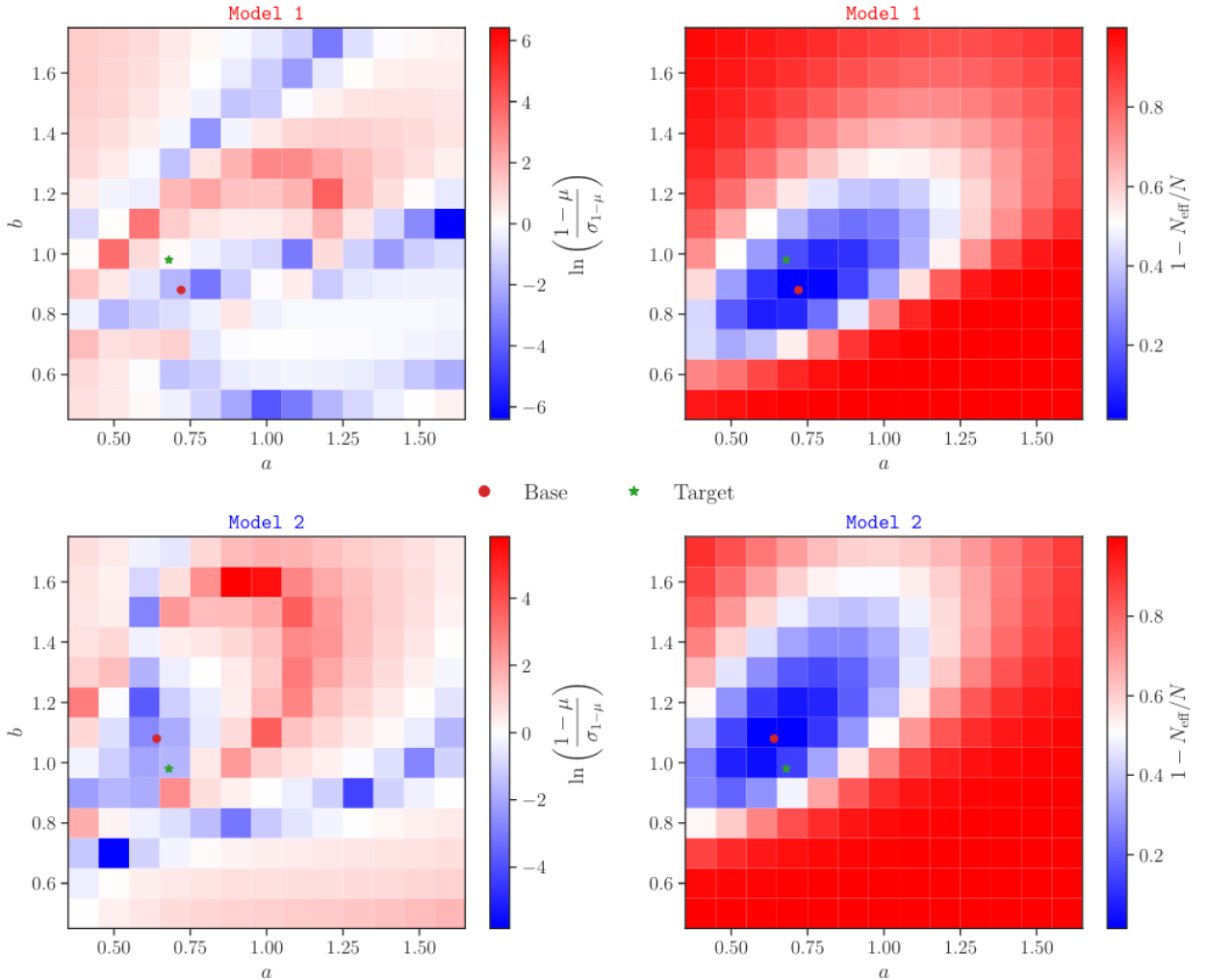
$$\mathcal{L}_{\chi^2}(\boldsymbol{y}_{\text{sim}}, \boldsymbol{y}_{\text{exp}}; \boldsymbol{w}) = \sum_{i=1}^{n_{\text{bins}}} \frac{(y_{\text{sim}}^{(i)} - y_{\text{exp}}^{(i)})^2}{\sigma_{\text{sim},i}^2 + \sigma_{\text{exp},i}^2}$$



No free lunch

Statistical power drops off quickly as you move away from the base parameterization

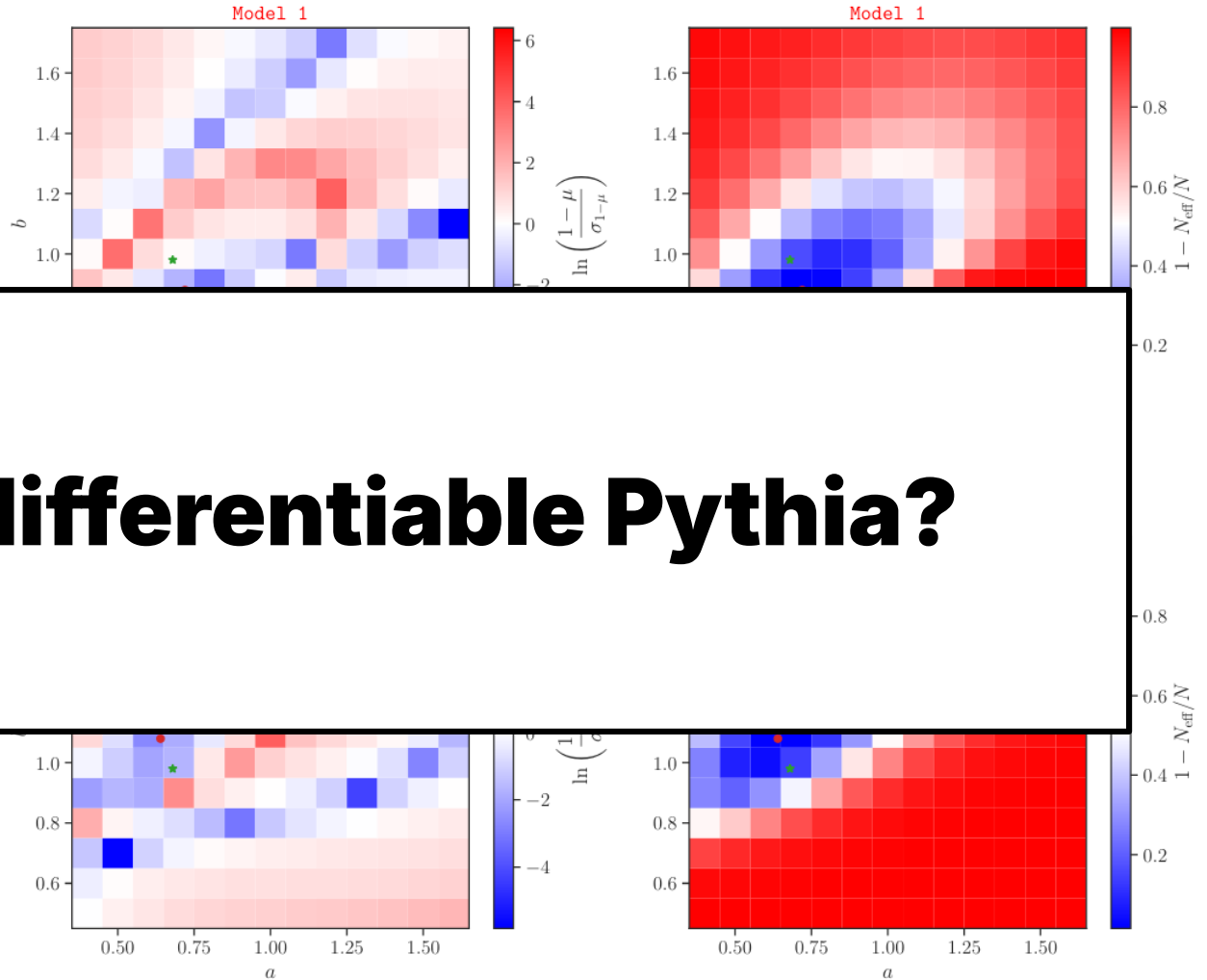
$$\mu \equiv \sum_{i=1}^{N} \frac{w_i}{N}, \qquad N_{\text{eff}} = \frac{\left(\sum_{i=1}^{N} w_i\right)^2}{\sum_{i=1}^{N} w_i^2}$$



No free lunch

Stati
quicl
the b

$$\mu \equiv \sum_{i=}$$



Road to differentiable Pythia?

WHAT is a differentiable simulator?

- All parameters are differential parameters
  - Differential with respect to what?

WHY a differentiable simulator?

- Parameter exploration (tuning)
- Uncertainty quantification
- Reproducibility

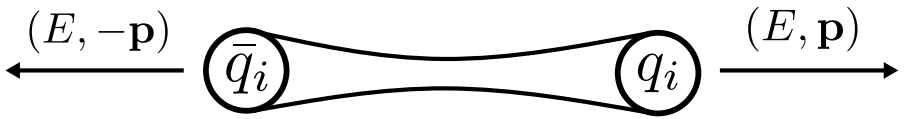
HOW to build a differentiable simulator?

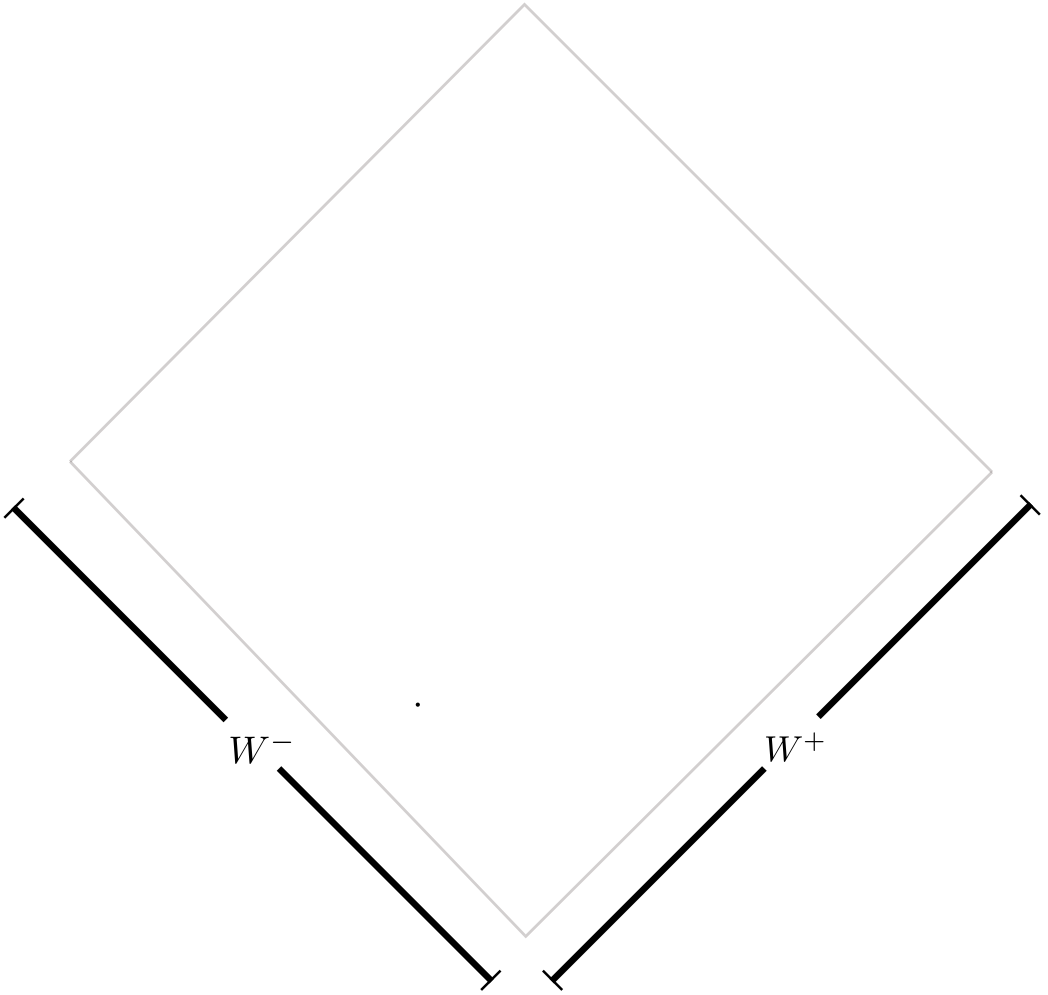
- Many roads to a fully differentiable event generator. Varying levels of “intrusivity” (from a developers perspective).

Half-baked thoughts on the interplay of “energy conservation” in hadronization and data-driven extraction of the fragmentation function

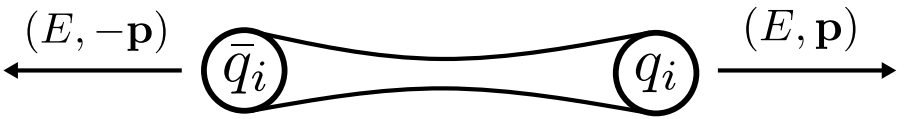
```
finalTwo
```

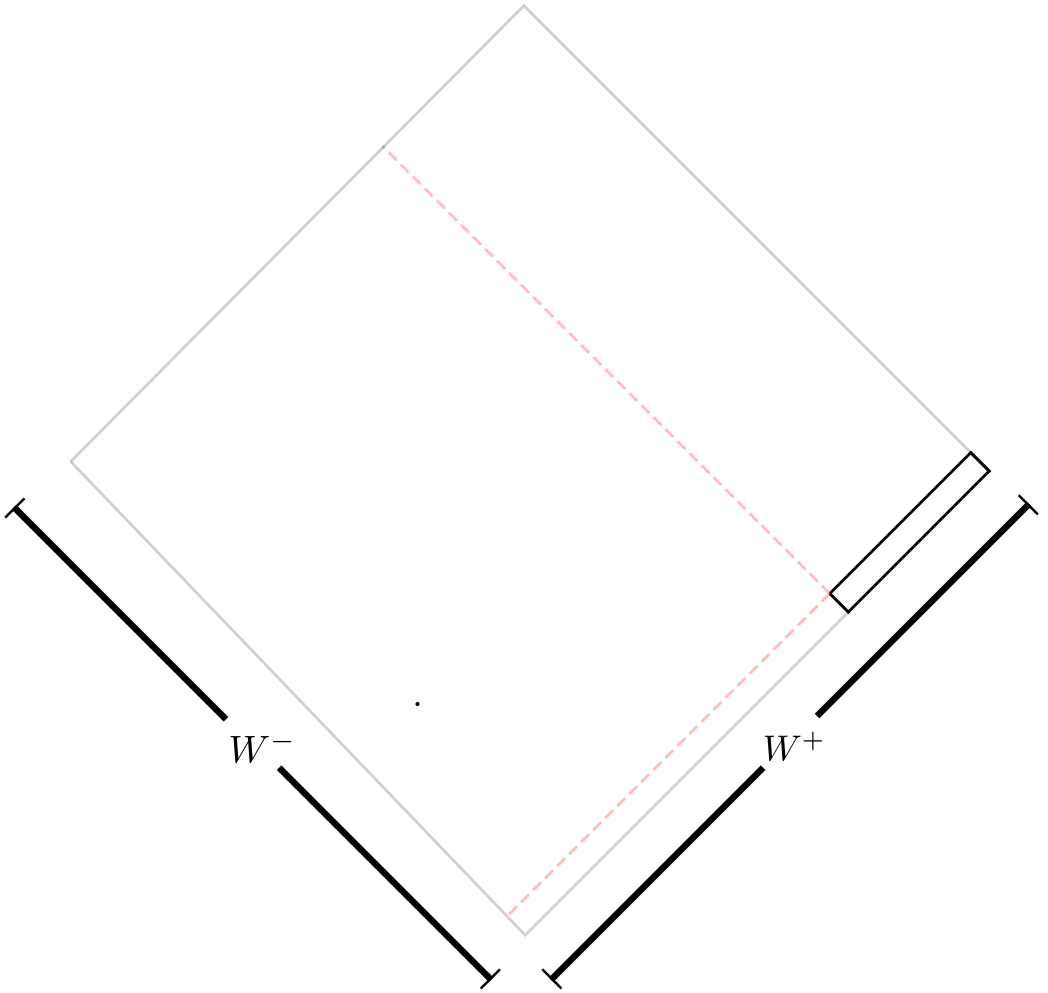
The algorithm ($q\bar{q}$)



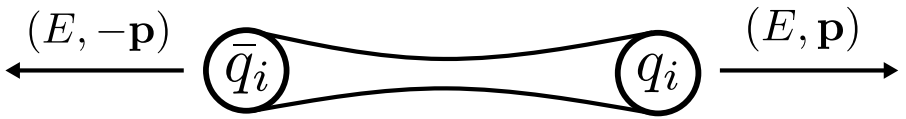


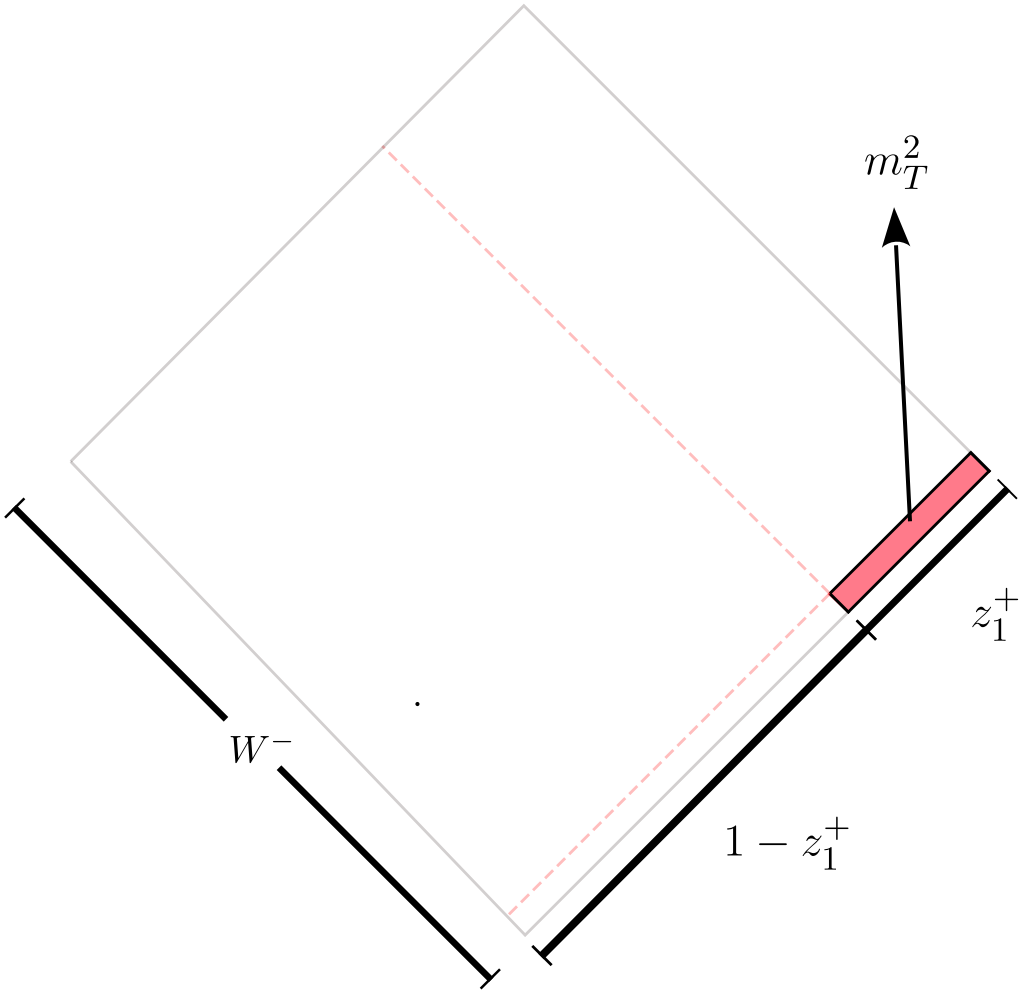
The algorithm ($q\bar{q}$)



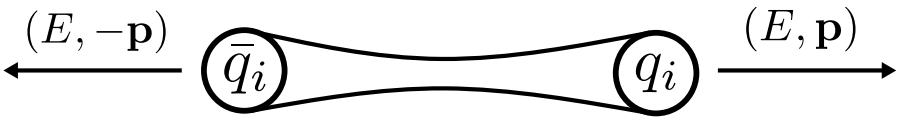


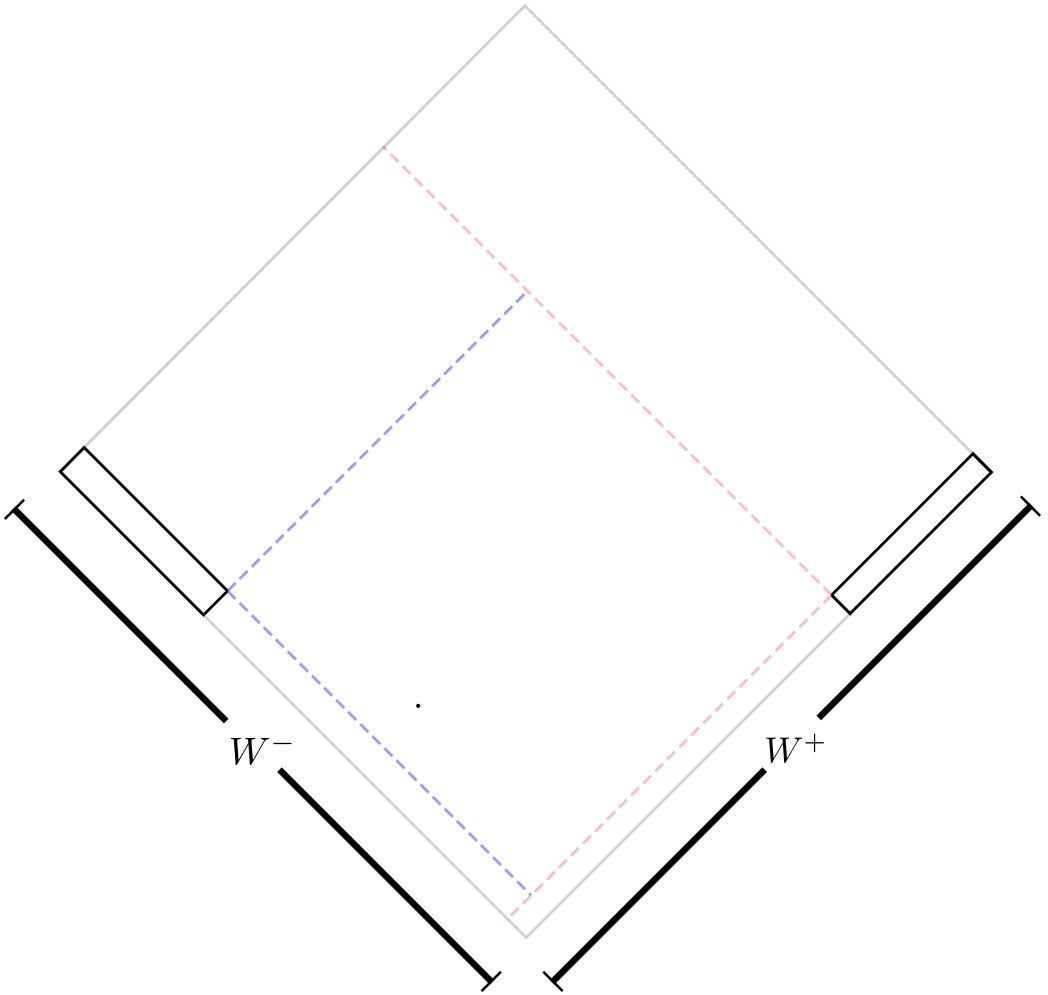
The algorithm ($q\bar{q}$)



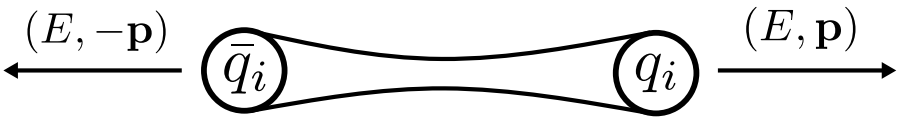


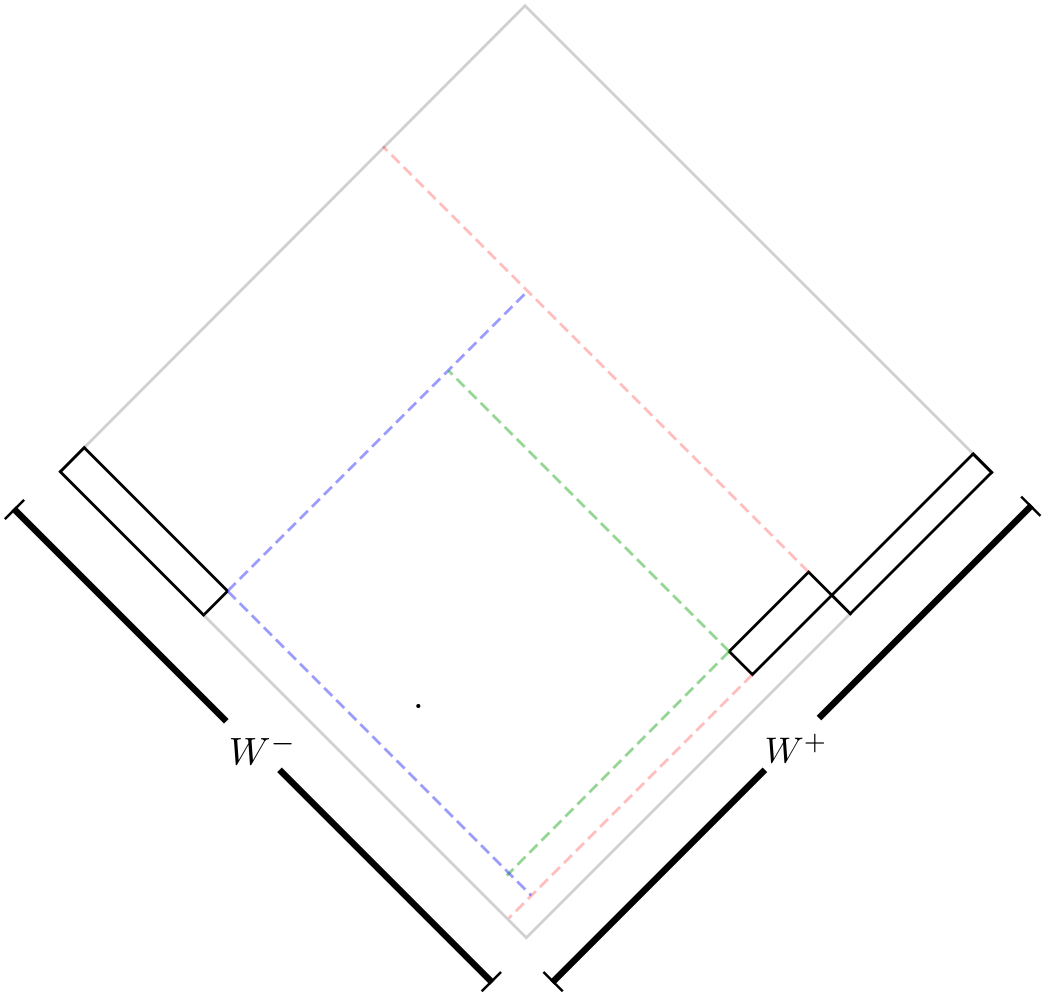
The algorithm ($q\bar{q}$)



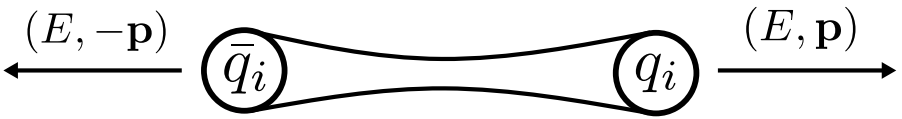


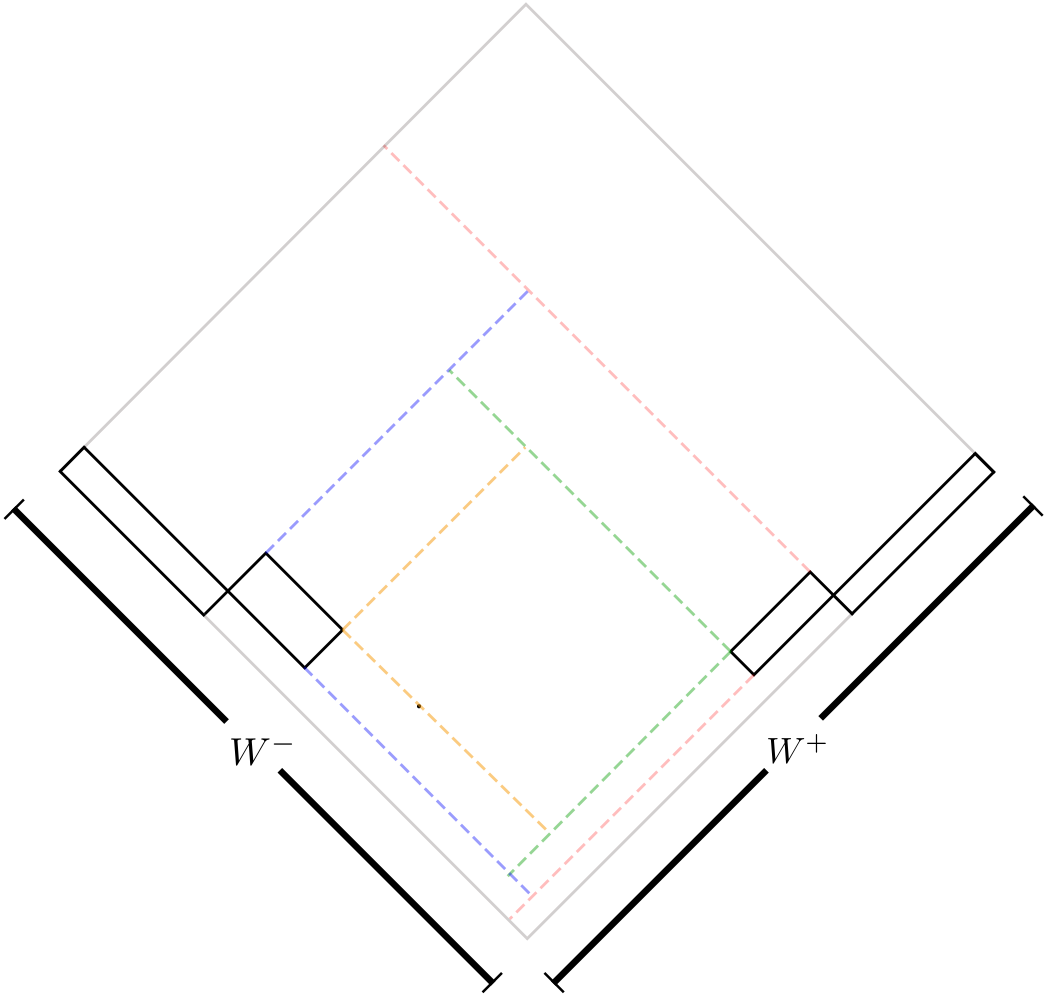
The algorithm ($q\bar{q}$)



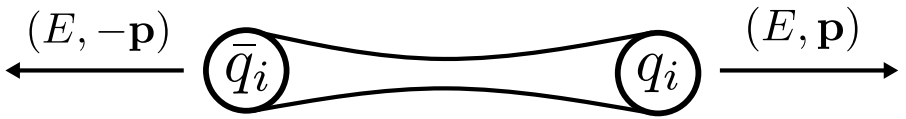


The algorithm ($q\bar{q}$)

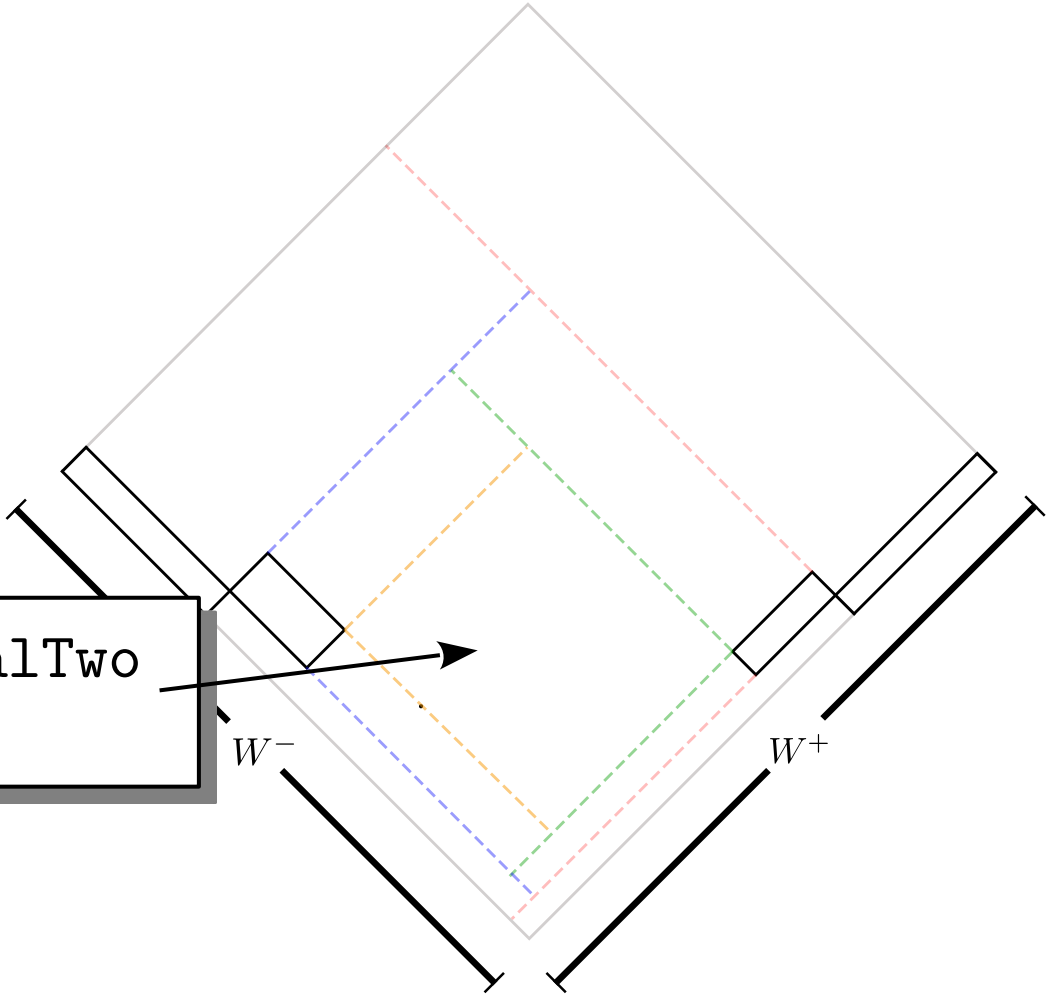




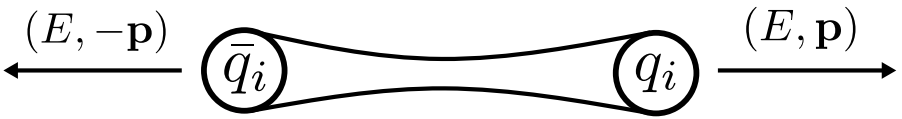
The algorithm ($q\bar{q}$)

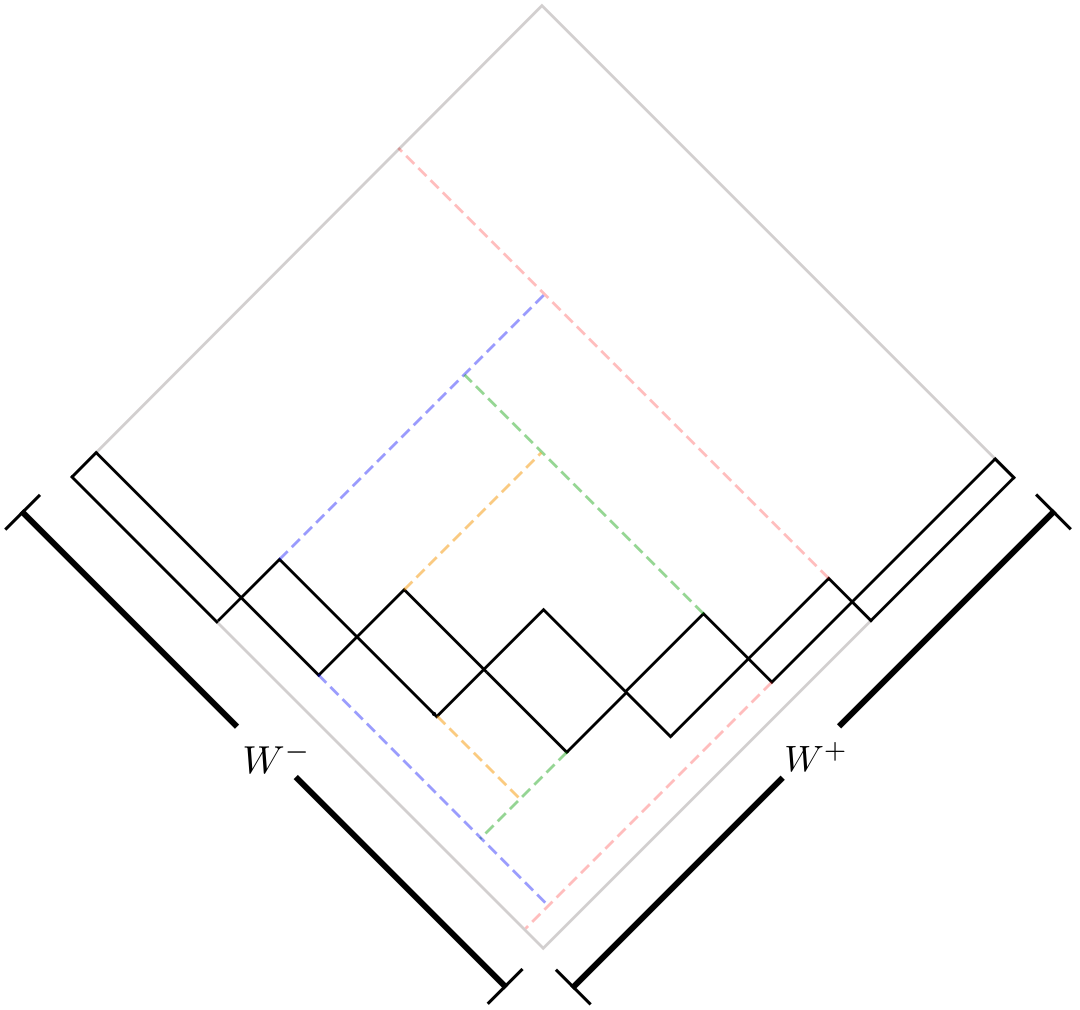


When E_{CM} goes below E_{cut}, `finalTwo` is called.

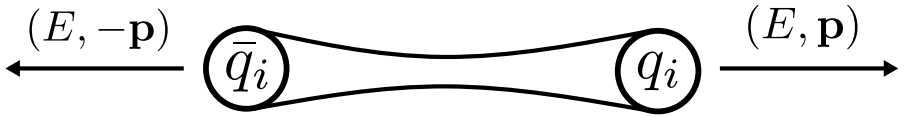


The algorithm ($q\bar{q}$)





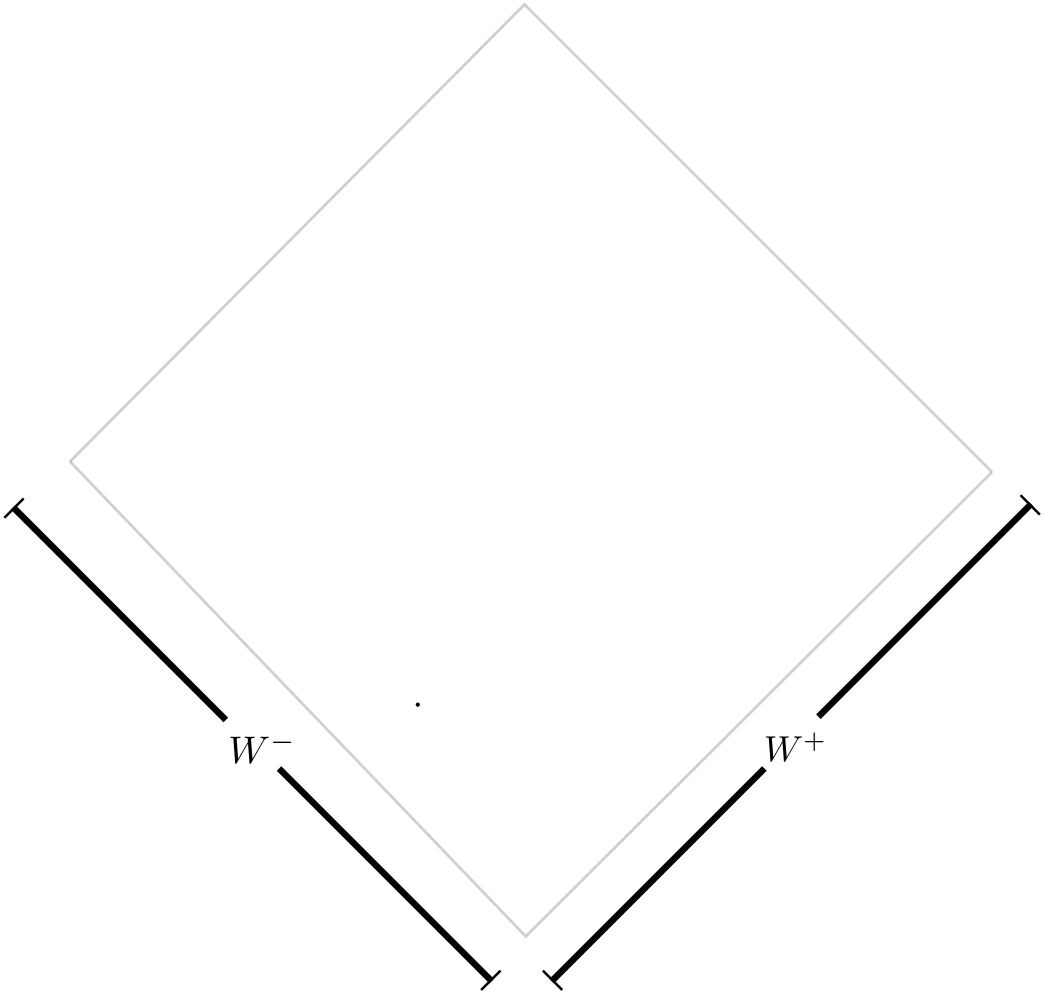
The algorithm ($q\bar{q}$)



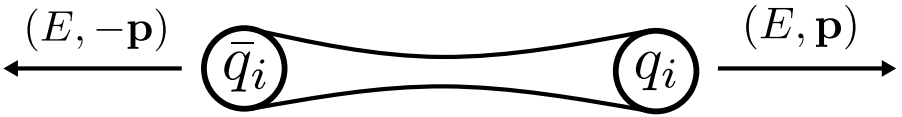
This system has two limits:

1. $E_{CM} >> m_h$

- Random walk in f(z)



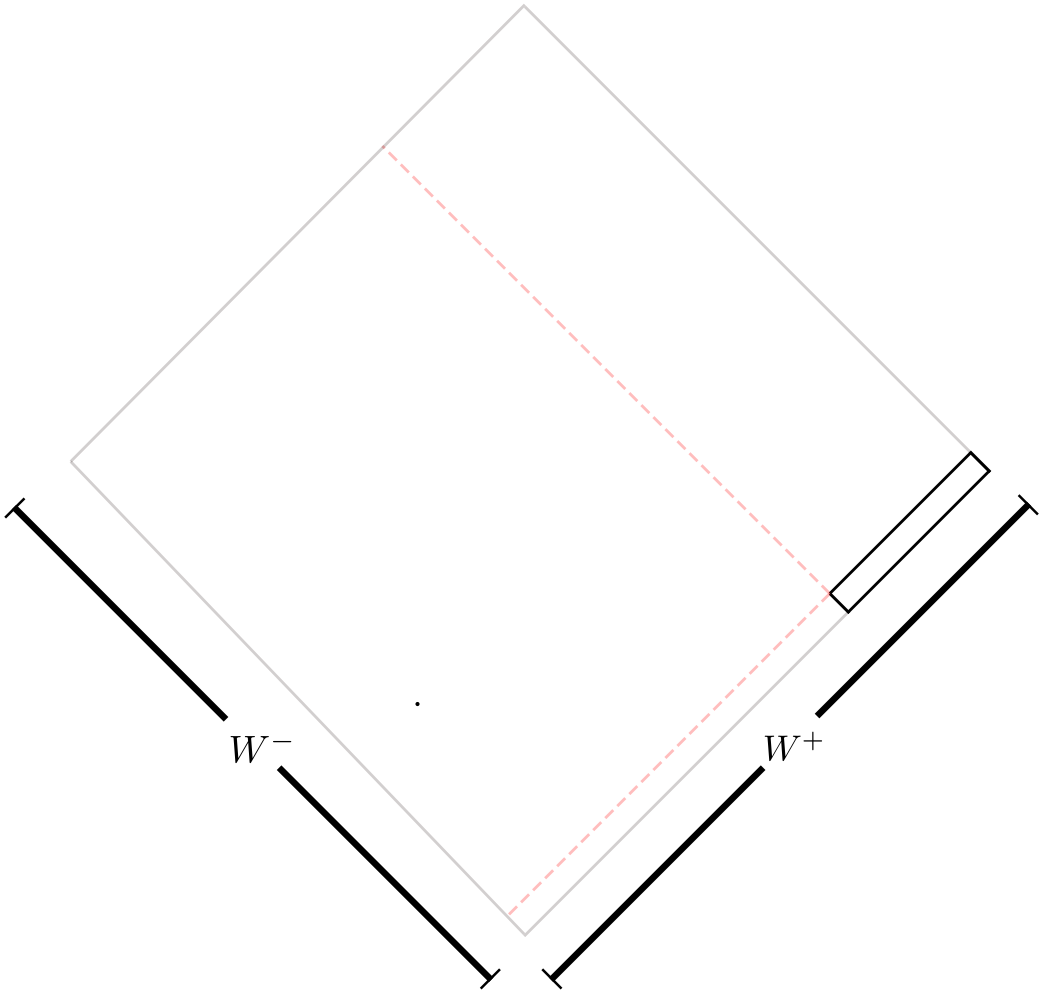
The algorithm ($q\bar{q}$)



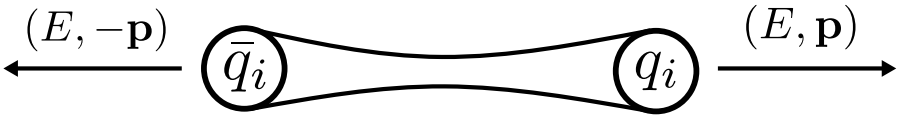
This system has two limits:

1. $E_{CM} >> m_h$

- Random walk in f(z)



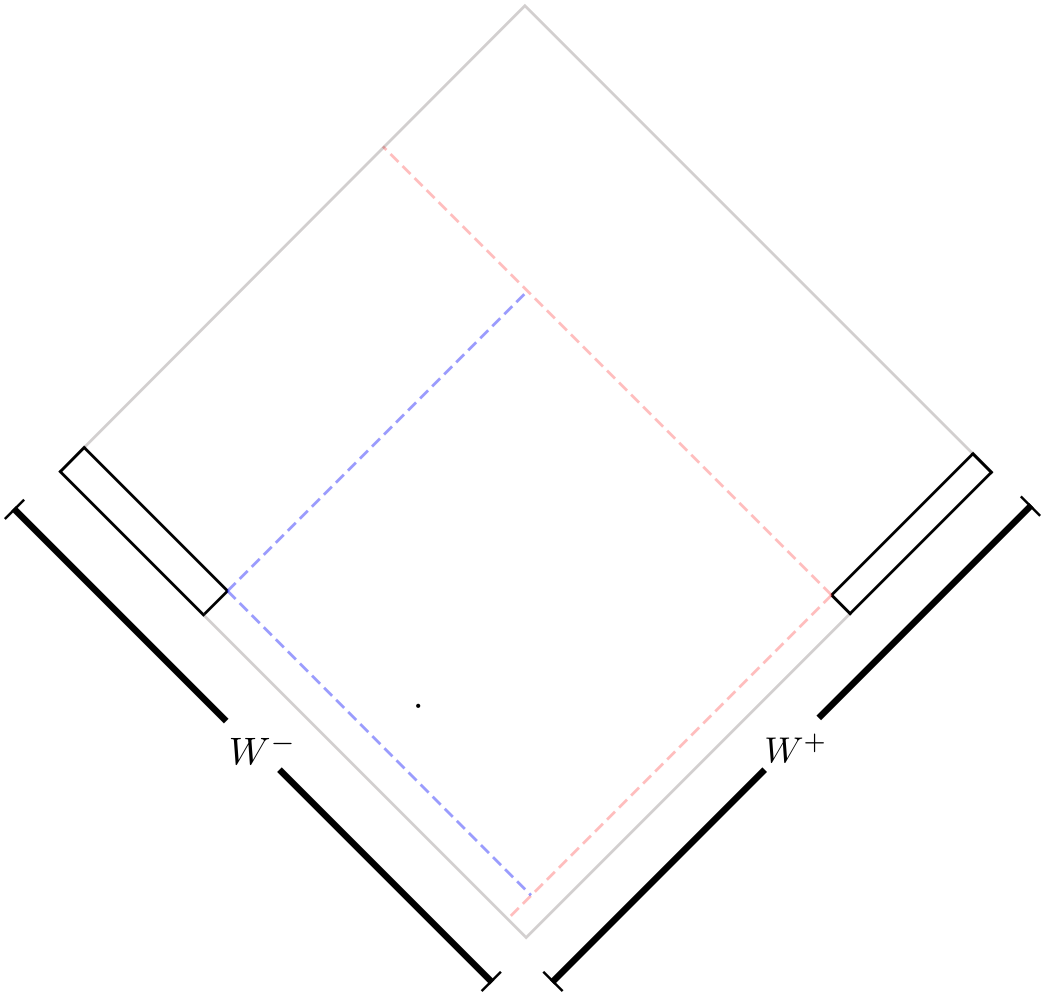
The algorithm ($q\bar{q}$)



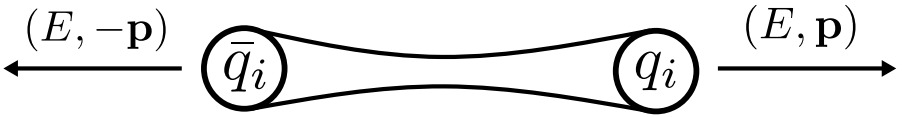
This system has two limits:

1. $E_{CM} >> m_h$

- Random walk in f(z)



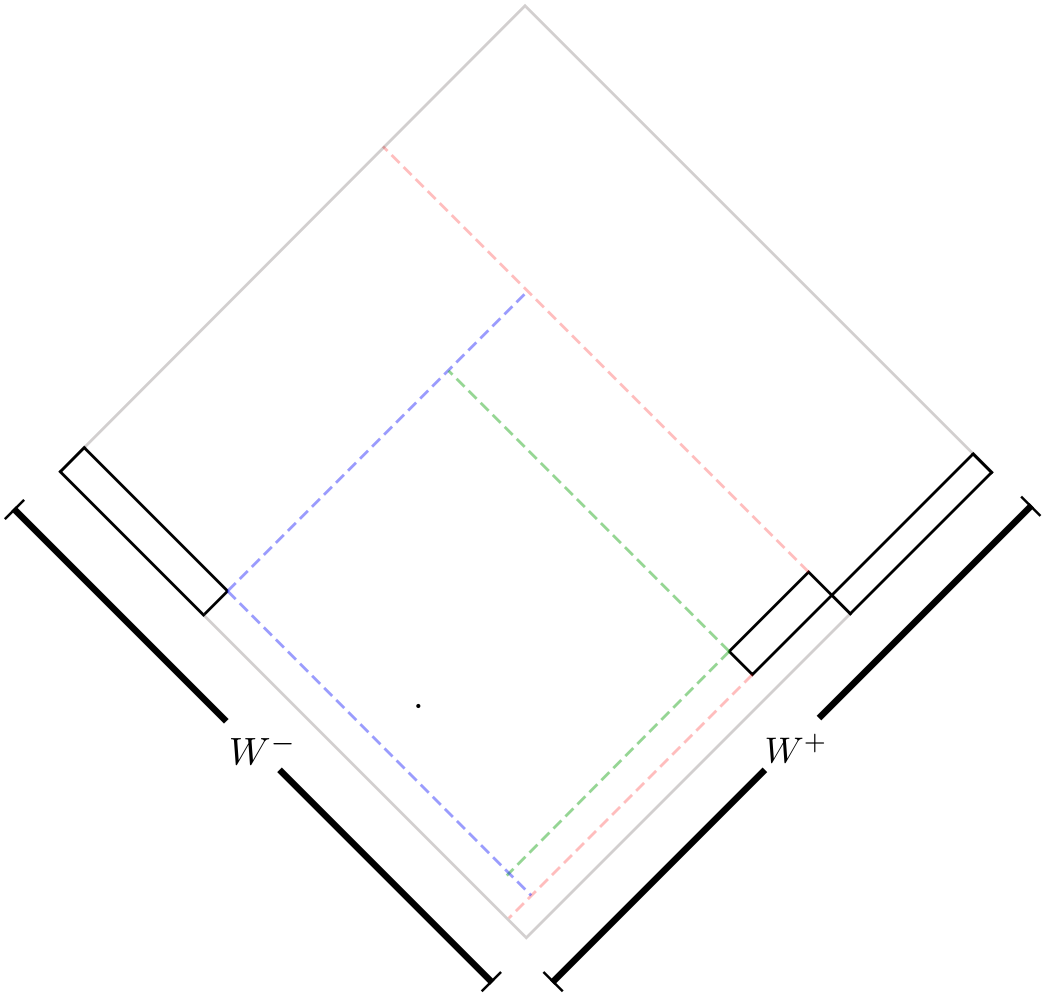
The algorithm ($q\bar{q}$)



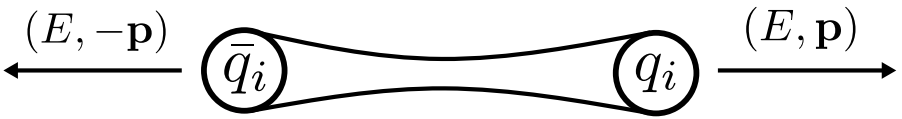
This system has two limits:

1. $E_{CM} >> m_h$

- Random walk in f(z)



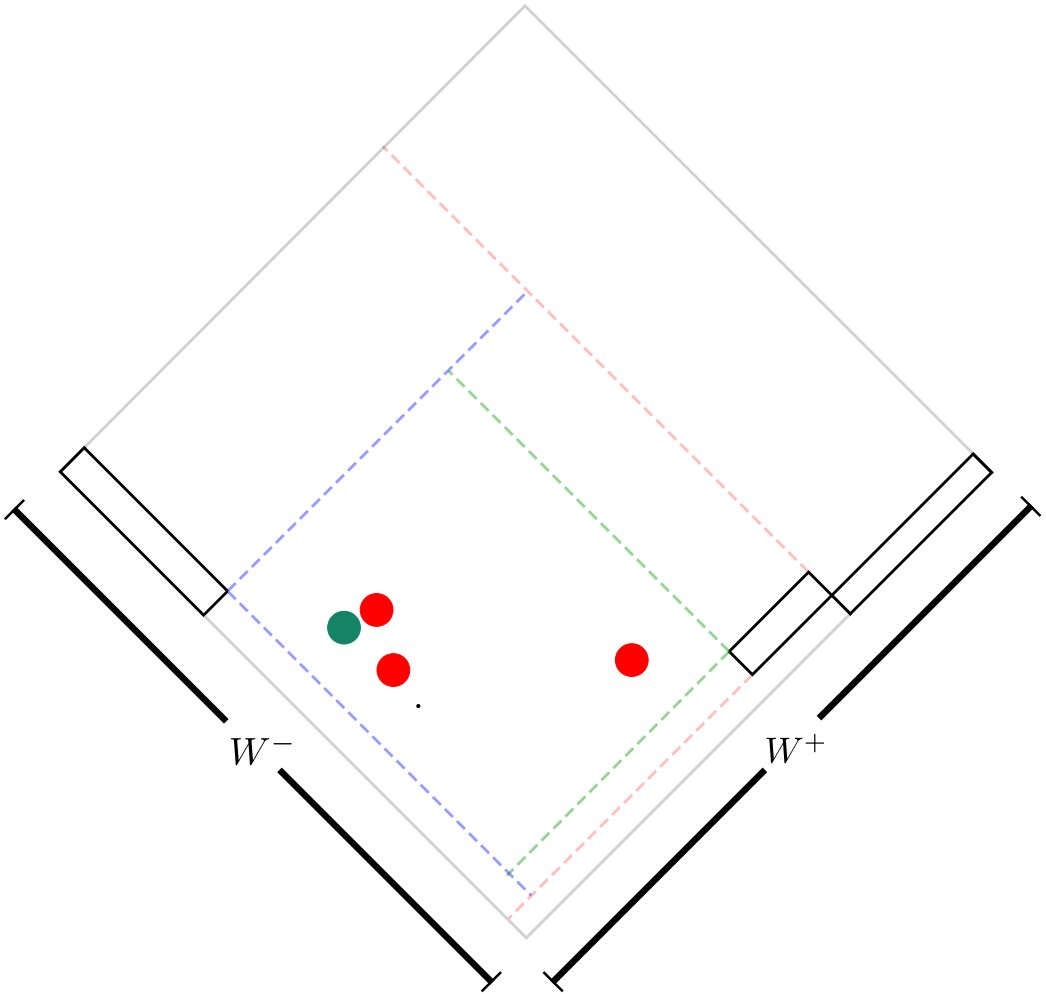
The algorithm ($q\bar{q}$)



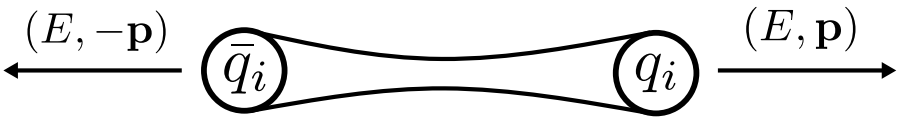
This system has two limits:

1. $E_{CM} >> m_h$
   - Random walk in f(z)

2. $E_{CM} \sim m_h$



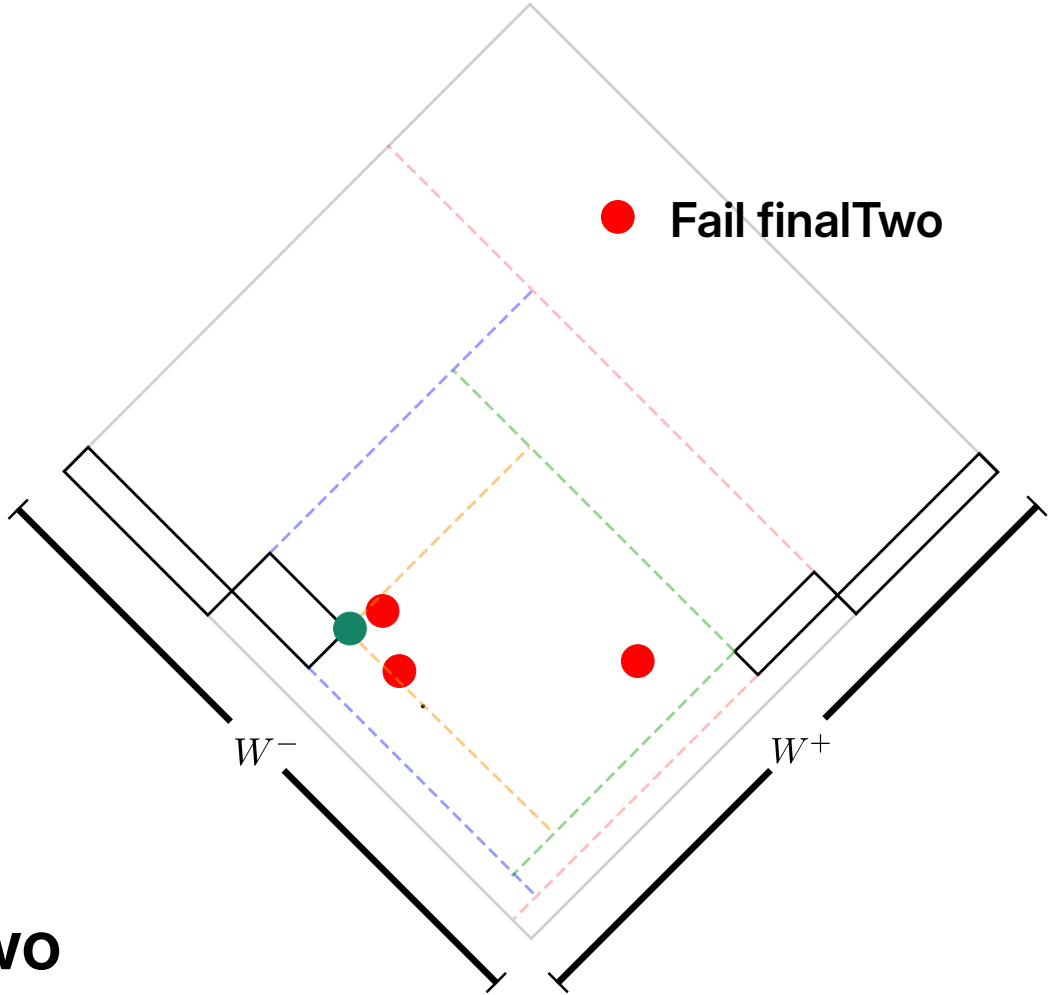
The algorithm ($q\bar{q}$)



This system has two limits:

1. $E_{CM} >> m_h$
   - Random walk in f(z)

2. $E_{CM} \sim m_h$
   - f′(z) influenced by finalTwo



The algorithm ($q\bar{q}$)

● Fail finalTwo

We want to extract f(z), not f′(z) – cleaner extraction would factorize these two pieces.

How?

- f′(z) influenced by finalTwo

Brownian bridge

- Gaussian random walk ($E_{CM} >> m_h$)
  - Markovian
- Gaussian random walk with termination condition, i.e. Gaussian bridge
  - Psuedo-Markovian

Brownian bridge

- Gaussian random walk ($E_{CM} >> m_h$)

$$dX_t = dW_t$$

- Gaussian bridge
  - Boundary conditions

$$X_0 = x_0 \qquad X_T = x_T$$

$$dX_t = \frac{x_T - X_t}{T - t} dt + dW_t$$

Brownian bridge

- Gaussian random walk ($E_{CM} >> m_h$)

$$dX_t = dW_t$$

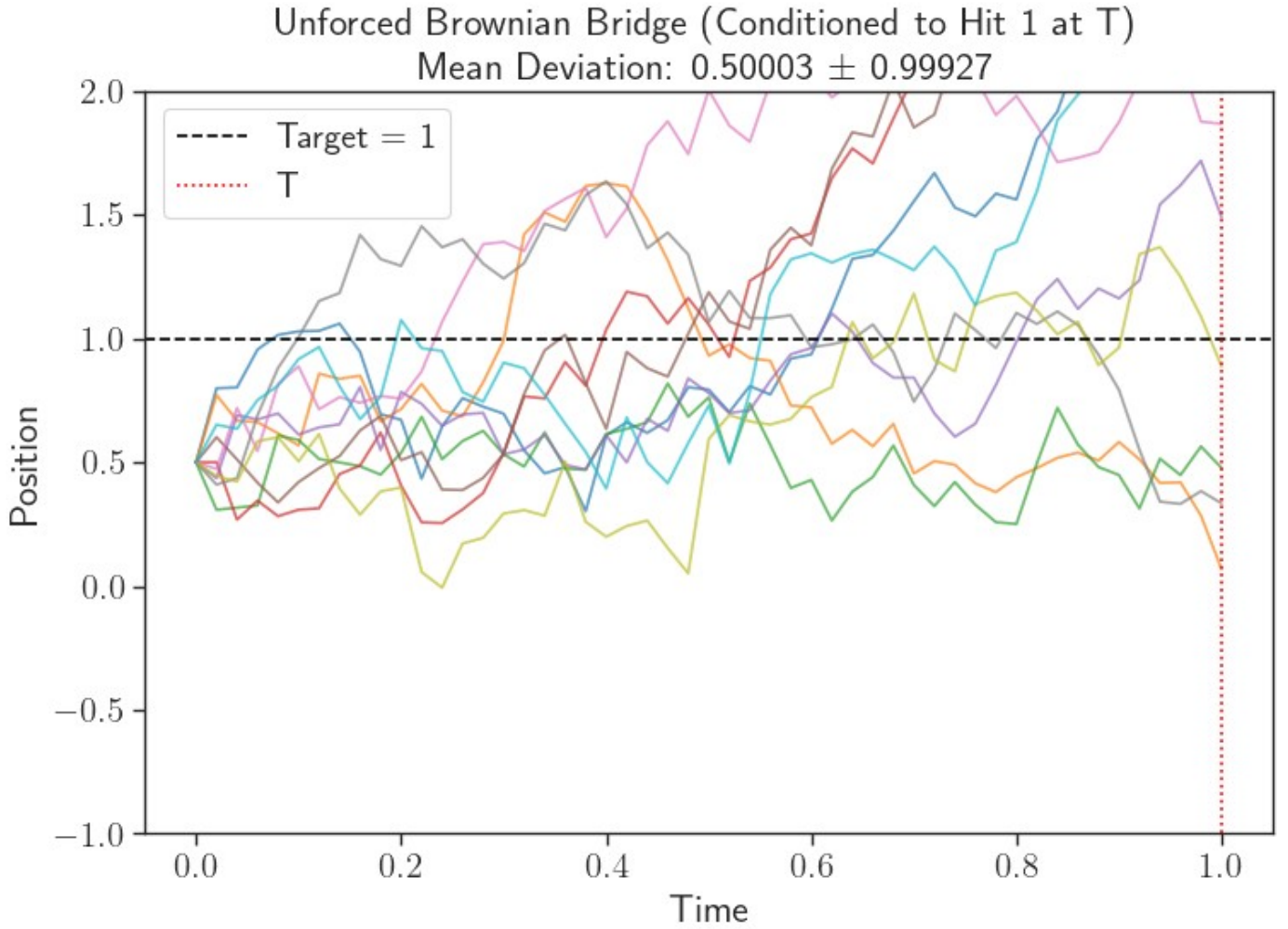
- Gaussian bridge
  - Boundary conditions

$$X_0 = x_0 \qquad X_T = x_T$$

$$dX_t = \frac{x_T - X_t}{T - t} dt + dW_t$$

Doob h-transform

- **Gau … ge**
  wall … ditions

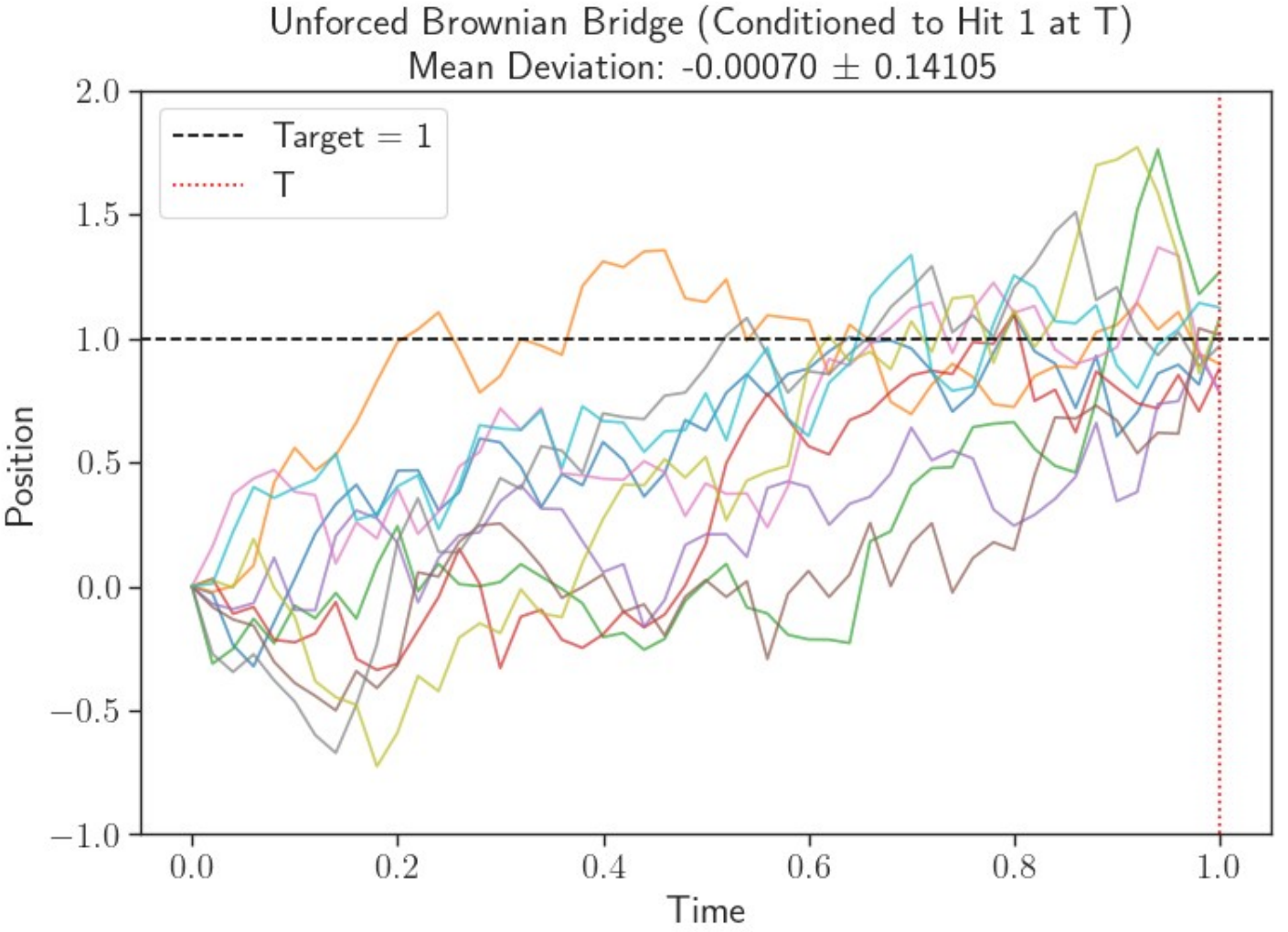


c … $T = x_T$

… $+ dW_t$

…rm

- **Gaus**
  walk



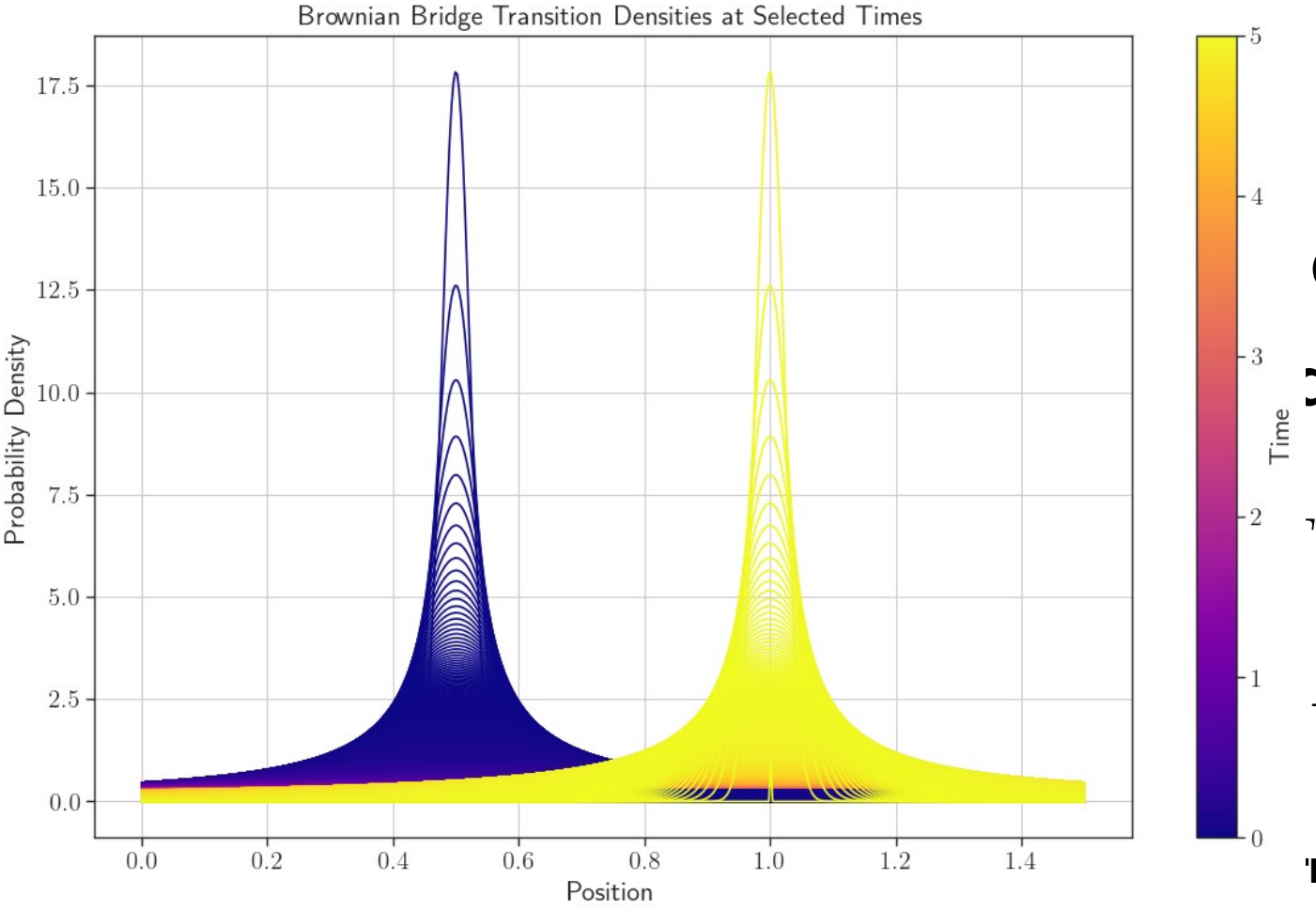
tions

dX

$= x_T$

dW_t

- Gau
walk



e

ditions

$= x_T$

$+ dW_t$

m

Conclusions

- Interesting invasive and non-invasive avenues for making a Pythia “fully” differentiable
- Interesting solutions for finalTwo? Parallels with constrained Brownian motion

Backups

The algorithm ($q\bar{q}$)

$(E, -\mathbf{p})$ $\bar{q}_i$

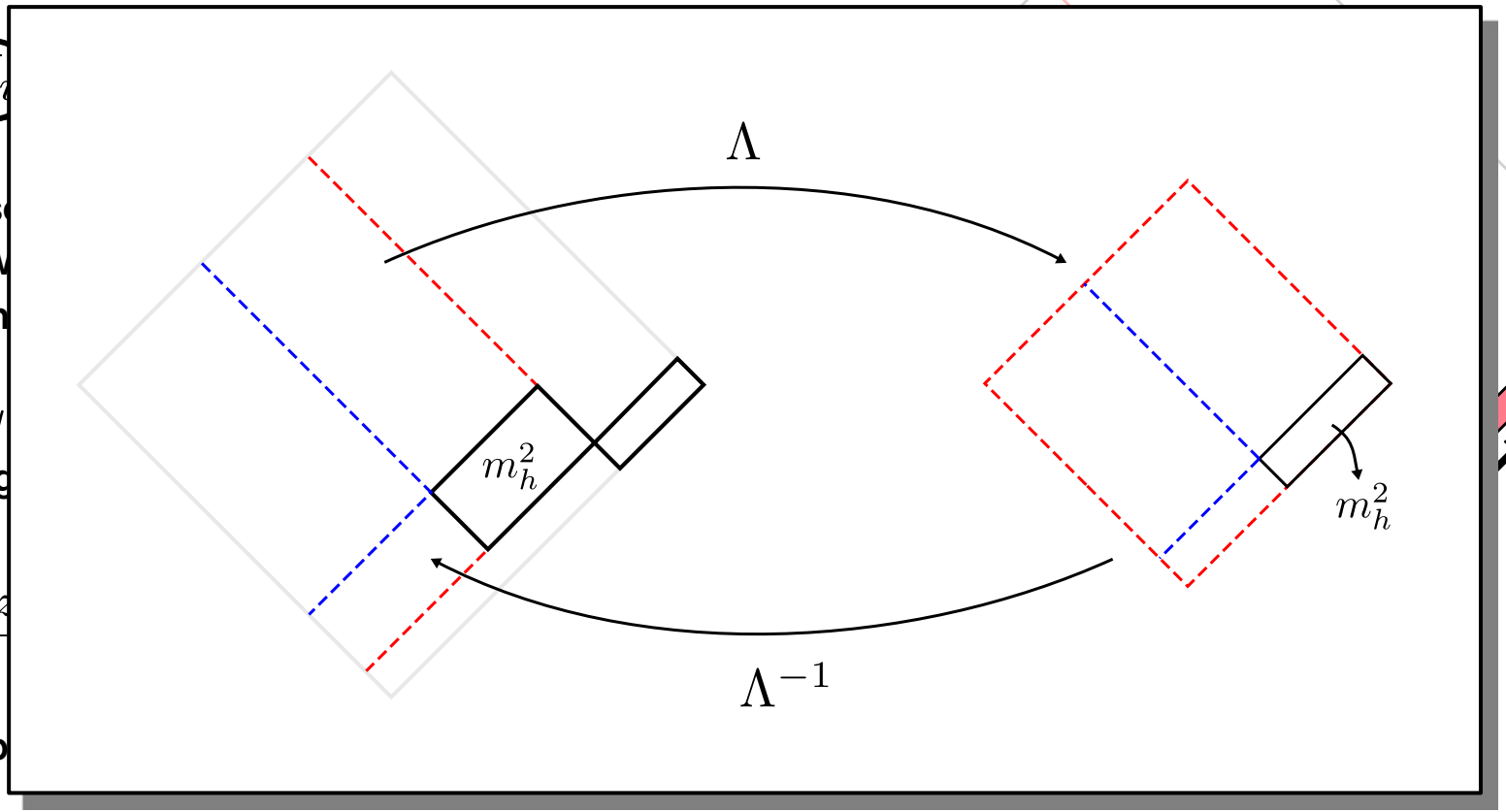
1) Randomly s
2) Sample new
3) Sample tran

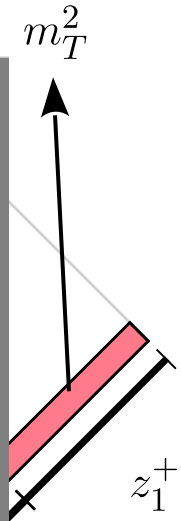
$\mathcal{P}(p_x, p_y$

4) Sample long
new hadron

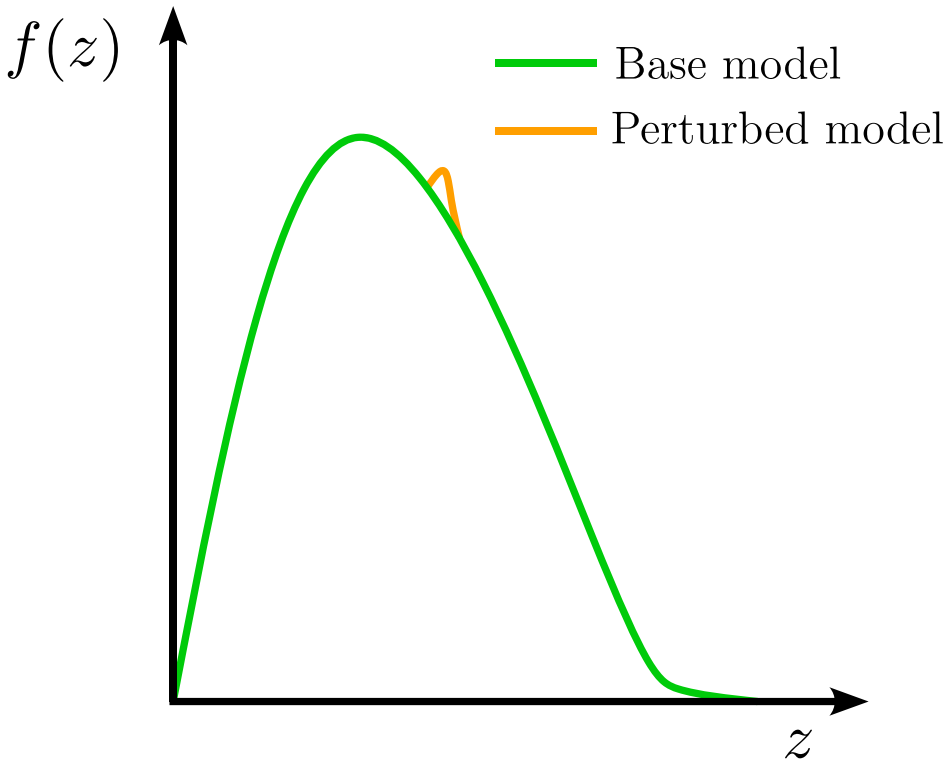
$f(z) \propto \frac{(1-z}{z}$

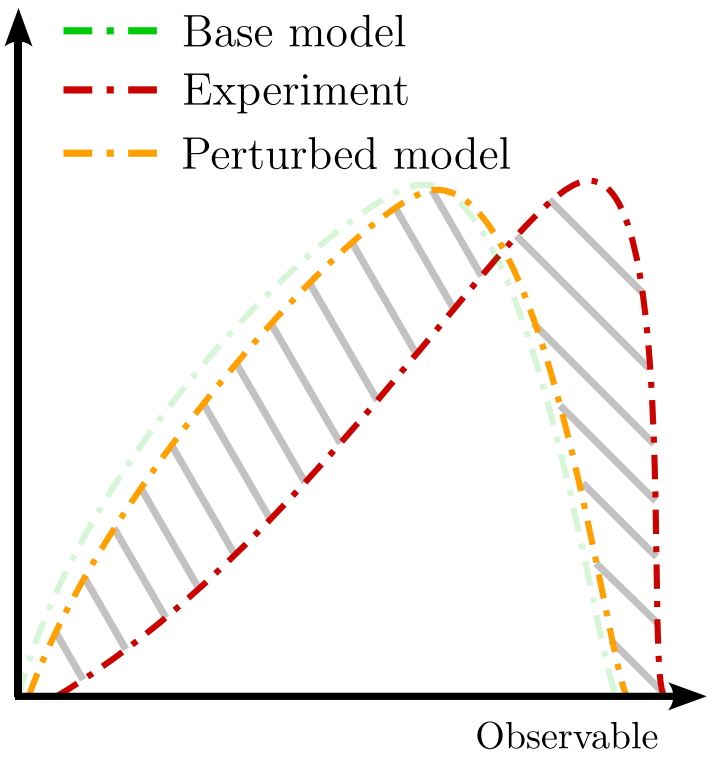
5) Repeat step



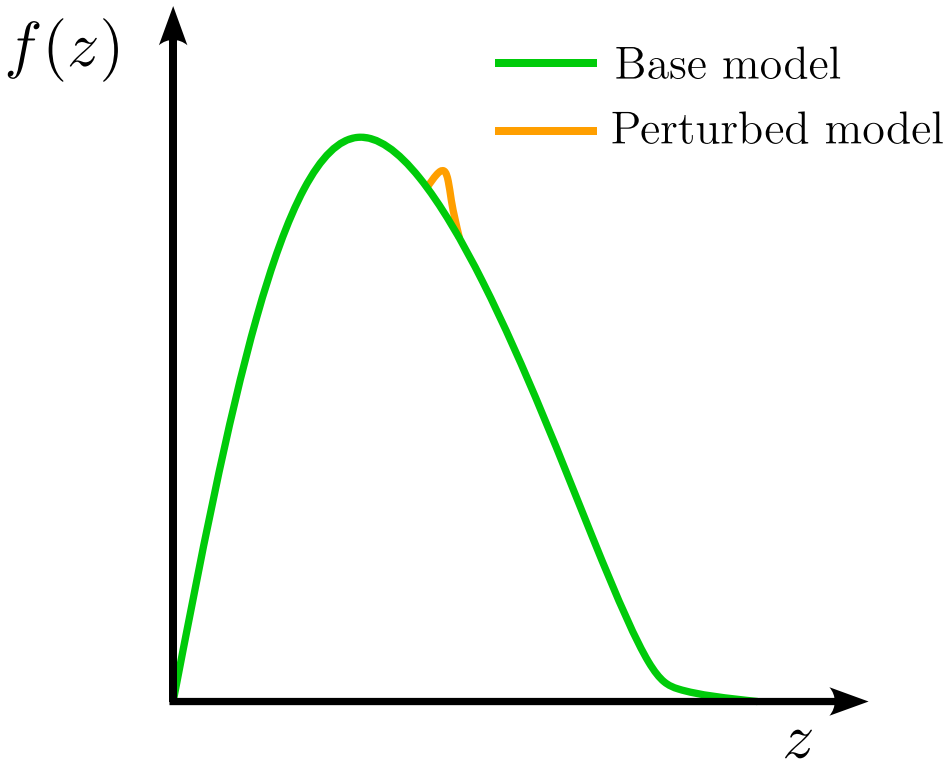


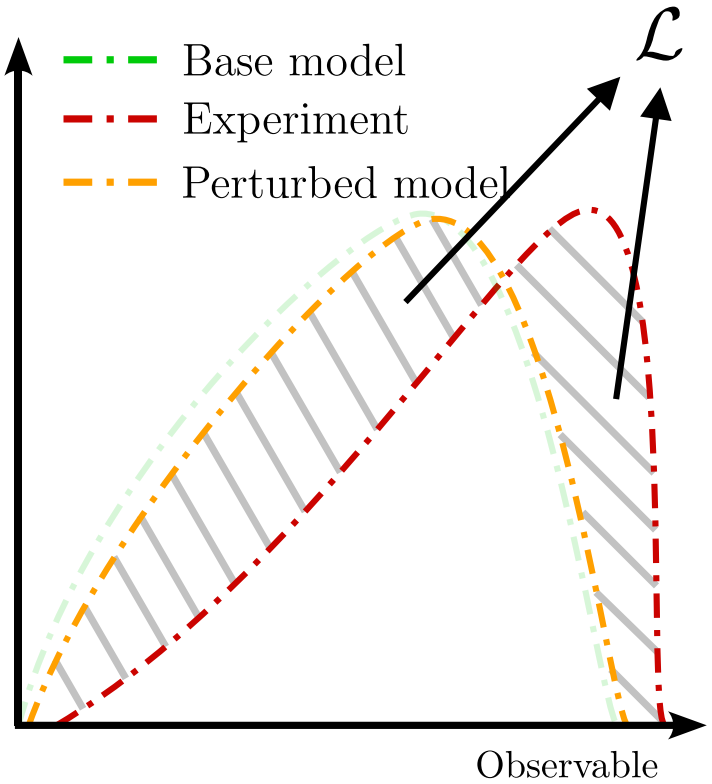
MLHAD efforts: big picture





MLHAD efforts: big picture





Applications

- Efficient means of exploring parameter space
- Inherently differentiable
- Very useful for tuning!
  - Embed the computation of weights into numerical autodifferentiation engine
  - Picking new parameters in the update step facilitated by well-developed optimizers (SGD, Adam, etc.)

Rejection sampling with Autodifferentiation (RSA)

The "score" observable

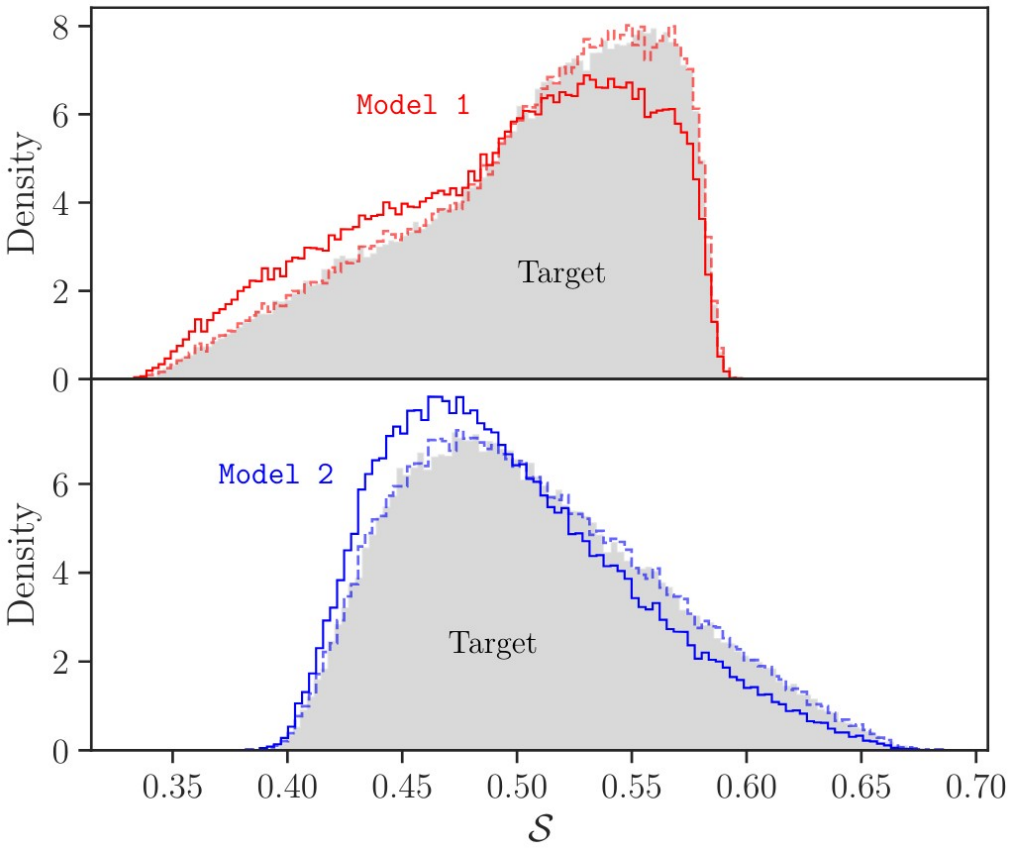
- Train a deepsets classifier to distinguish simulation from experiment
  - Takes full event information as input

$$(E, p_x, p_y, p_z)_1$$

$$(E, p_x, p_y, p_z)_2$$

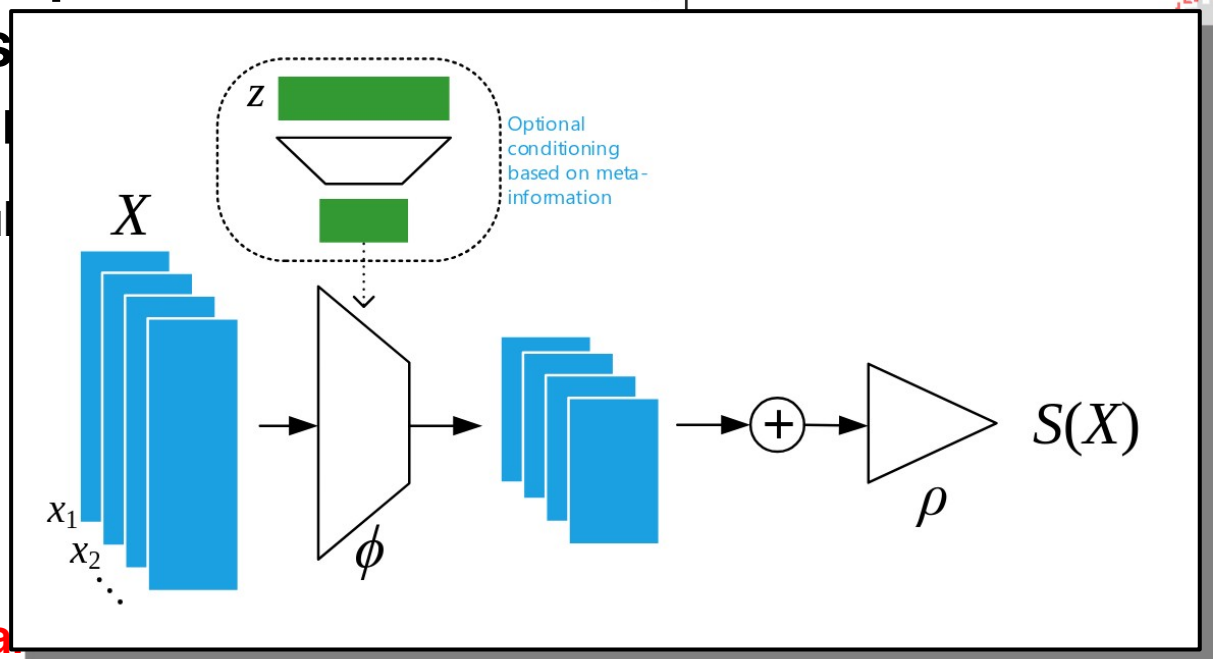
$$\ldots$$

Use trained classifier output (score) as an observable

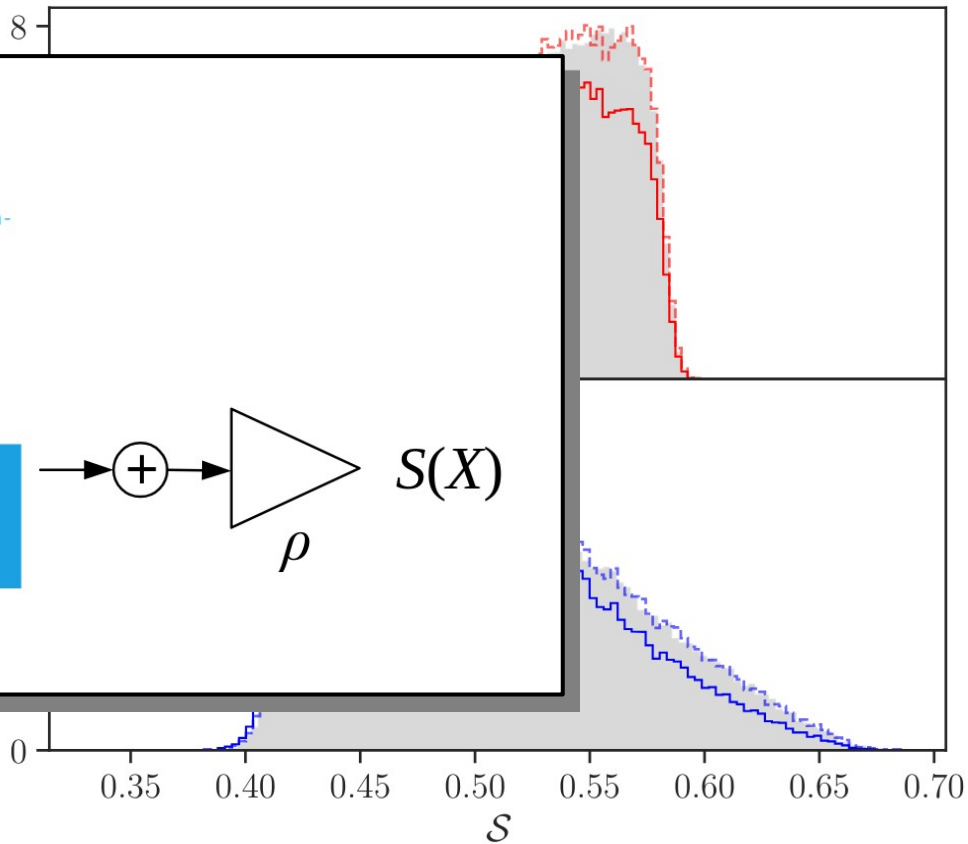


The "score" observable

- Train a deepsets classifier to distinguis
  experimer
  - Takes ful
    input



Use tra
(score) as an observable



Classifier score with full event info improves tuning convergence