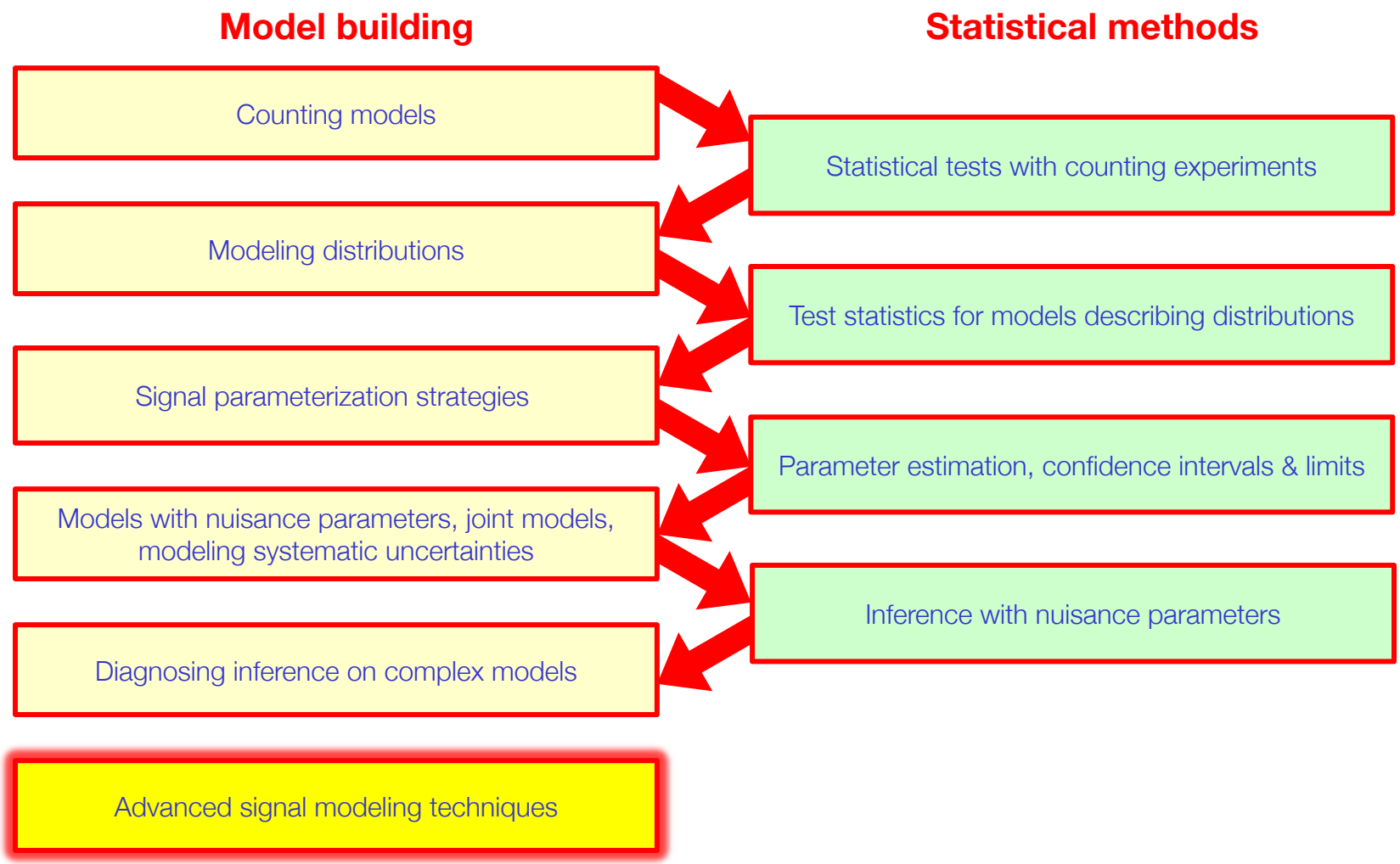
Model building 6

Advanced signal modeling: convolutions, matrix element methods, amplitude-based morphing models

Wouter Verkerke, NIKHEF

Roadmap of this course

- Start with basics, gradually build up to complexity

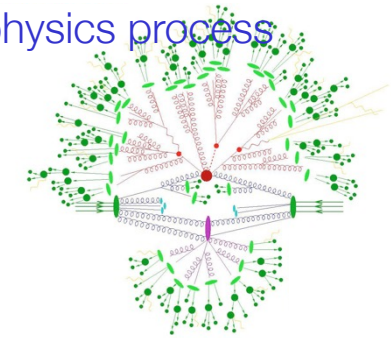


Goal of *measurement* – inference on a theory parameter

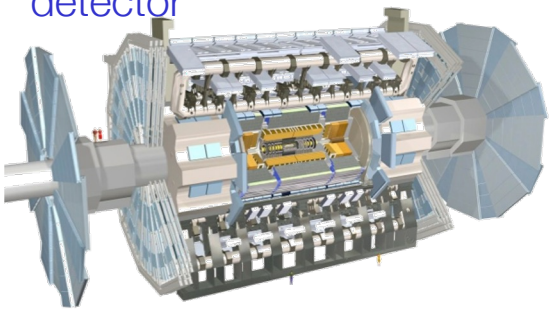
- So far – a lot of the material discussed modeling of background, general modeling uncertainties, and extract of a signal (discovery or limit setting)
- For established signals, the main goal is usually to measure as precisely as possible the value of parameter of the model, which connects to an underlying physics theory.
- In many cases, measured model parameters (e.g. mean of invariant mass) don't map *exactly* to underlying theory parameters (pole mass) because of smearing and bias effects in the detector.
- In this section we cover some examples of techniques to infer the underlying theory parameters

The great smearing machine

Simulation of 'soft physics' physics process

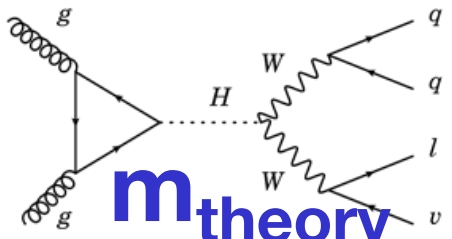


Simulation of ATLAS detector



LHC data

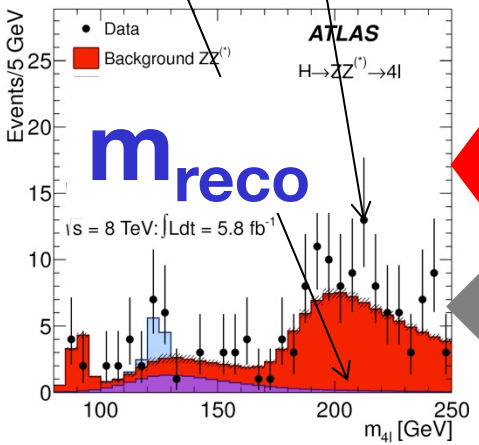
Simulation of high-energy physics process



m_{theory}

$P(m_{4l}|SM[m_H])$

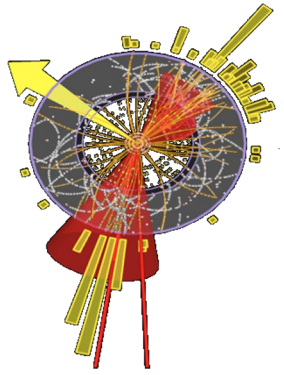
Observed m_{4l}



m_{reco}

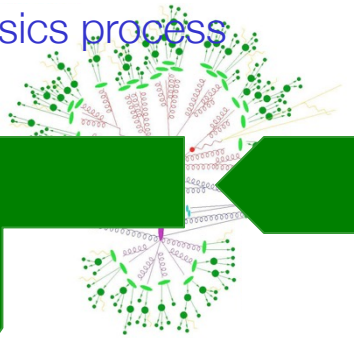
Analysis Event selection

Reconstruction of ATLAS detector

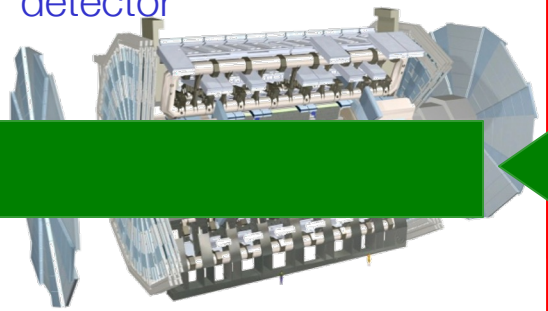


Can we invert this proces?

Simulation of 'soft physics' physics process



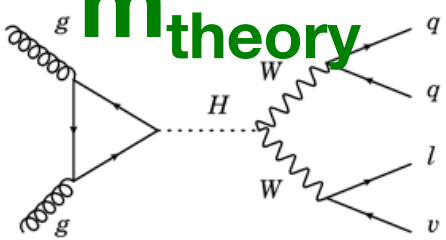
Simulation of ATLAS detector



LHC data

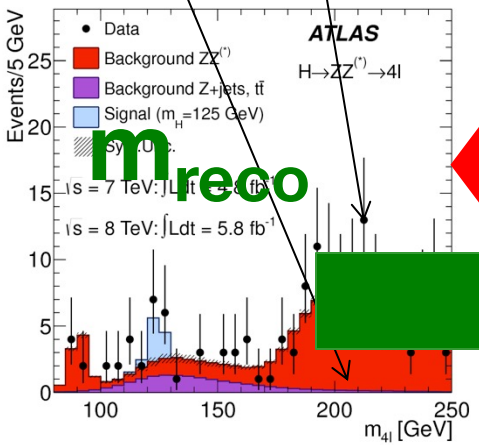
Simulation of high-energy physics process

m_{theory}



$P(m_{4l}|SM[m_H])$

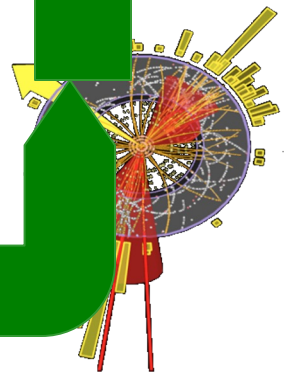
Observed m_{4l}



m_{reco}

Event selection

Reconstruction of ATLAS detector



The detector as convolution

- The effect of the detector (and analysis machinery) on the theory parameter can be expressed as a *convolution*

Theory distribution

$$f(x_{reco}) = \int f(x_{theo}) K(x_{reco}, x_{theo}) dx_{theo}$$

Reconstructed distribution

Kernel function
(resolution / migration)

The detector as convolution

- For a perfect detector the Kernel K is delta function

$$f(x_{reco}) = \int f(x_{theo})\delta(x_{reco} - x_{theo})dx_{theo}$$

- But if detector response function to x_{theo} is (to good approximation) independent of the value x_{theo} over a wide enough range, can then also represent problem as

$$f(x_{reco}) = \int f(x_{theo})R(x_{reco} - x_{theo})dx_{theo}$$

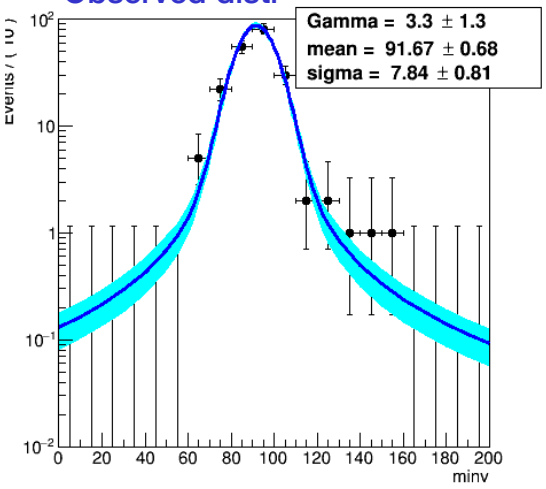
- Here R is a resolution model, typically a Gaussian (or sum of Gaussians)

Explicit modeling of the detector convolution

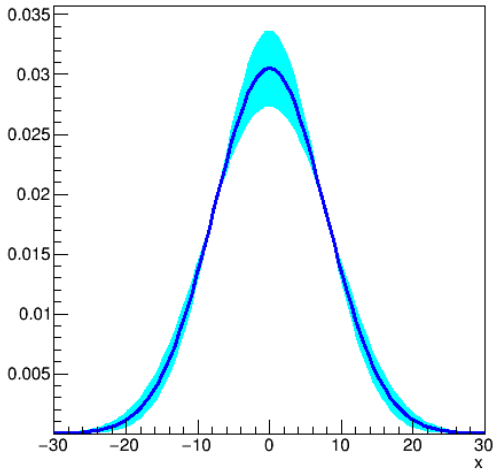
- Example: reconstruction of a particle theory mass m_{th}, from a reconstructed invariant mass m_{inv}

$$f(m_{inv} \mid M, \Gamma, b, \sigma) = \int Gaussian(m_{inv}^{th} - m_{inv}^{reco}, b, \sigma) \cdot BreitWigner(m_{inv}^{th} \mid \Gamma, M) dm_{inv}^{th}$$

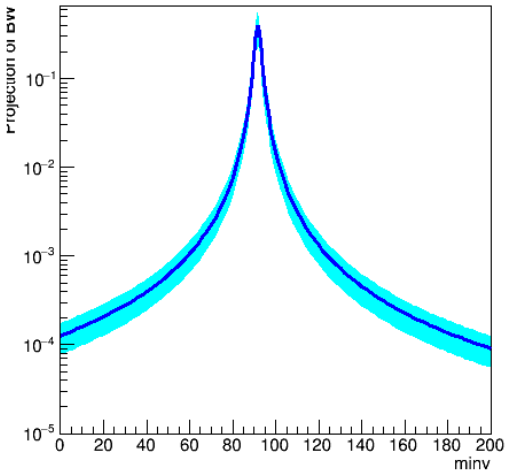
Observed distr



Fitted kernel function



Fitted theory distribution



- Note: probability model for observed m_{inv} now directly describes parameter M of underlying theory
  - Can also introduce parameters of resolution model as fittable model parameters
  - But not always sensitivity (e.g. bias *b* is inseparable from *M* in above model)

Calculating convolutions – analytical vs numeric

- Convolutions are best performed with analytical expressions, but no in majority of cases no known analytical form.
  - Some exceptions:
    Exponential (x) Gaussian → RooFit class RooDecay
    Breit-Wigner (x) Gaussian → RooFit class RooVoigtian
- But numeric calculation is also possible.
  - Direct numeric calculation of convolution integral usually a bad idea.
  - Very challenging computation, in particular to get required precision for use in MINUIT minimization (10^{-6} relative precision needed)
  - Explicit normalization needed of convolution into pdf → two integrals
  - In practice models with direct numeric integration unbearably slow

$$F(x) \otimes G(x) = \frac{\int_{-\infty}^{+\infty} F(x)G(x-x')dx'}{\int_{x_{\min}}^{x_{\max}} \int_{-\infty}^{+\infty} F(x)G(x-x')dx'dx}$$

Calculating convolutions – numeric direct vs FFT

- But numeric convolutions can also be calculated another way
- Makes use of the Circular Convolution Theorem:
  - For a circular observable (i.e. observable spaces loops back from x_{max} to x_{min}) convolution operation in normal space maps to multiplication in Fourier space!

$$\begin{aligned}\mathcal{F}^{-1}\{\mathbf{X}\cdot\mathbf{Y}\}_n &= \sum_{l=0}^{N-1} x_l \sum_{m=-\infty}^{\infty} y_m \left(\sum_{p=-\infty}^{\infty} \delta_{m(n-l-pN)}\right) \\ &= \sum_{l=0}^{N-1} x_l \sum_{p=-\infty}^{\infty} \left(\sum_{m=-\infty}^{\infty} y_m \cdot \delta_{m(n-l-pN)}\right) \\ &= \sum_{l=0}^{N-1} x_l \left(\sum_{p=-\infty}^{\infty} y_{n-l-pN}\right) \stackrel{\text{def}}{=} (\mathbf{x} * \mathbf{y_N})_n \ ,\end{aligned}$$

  - Can also convolution as three-step process

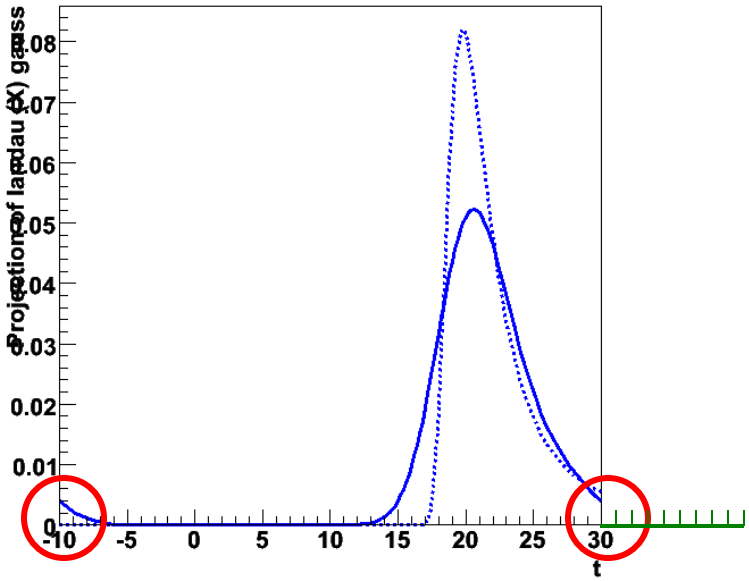
    1) Forward Fourier transform of p.d.f. and resolution model
    2) Multiplication of p.d.f and resolution model in Fourier space
    3) Inverse Fourier transform of product function back to normal space

  - Seems complicated, but excellent numeric implementation of Fourier Transforms exist (even for free!)
    - Fastest Fourier Transform in the West (freely available, installed unix library)
    - Key parts optimized in ASM code

FFTW

Calculating convolutions – numeric direct vs FFT

- Fourier Convolutions – two important features
  - Functions must first be discretized (sample) before FFT
    → minor effect at high enough sampling resolution
    (10.000 bins is numerically very feasible)
  - Circular space assumption can cause 'leakage' of probability from x_{max} to x_{min}



→ Can be mitigated by choosing observable range wide enough
→ Or if not possible introduce some 'buffer range' in observable that will absorb leakage before it loops back

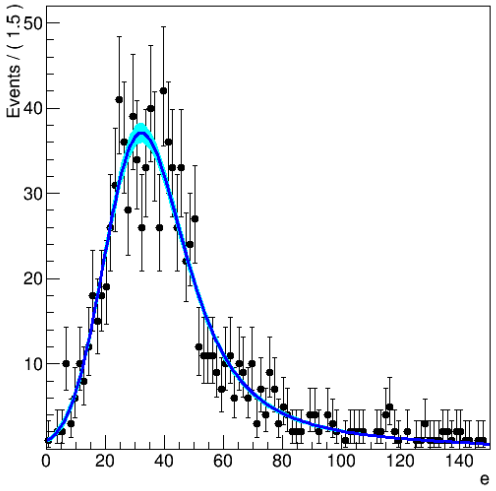
'ex18.C'

FFT Convolutions in RooFit

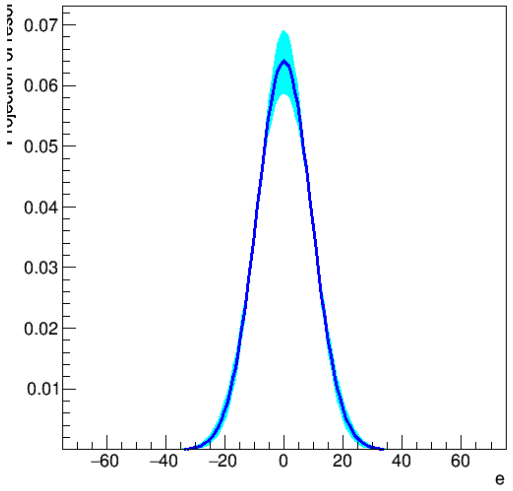
- Fourier convolution implement in FCONV operator

```
RooWorkspace w("w") ;
w.factory("Landau::phys(e[0,150],mean[30,0,60],sigma[5,1,10])") ;
w.factory("Gaussian::resol(e,0,sigma_gauss[10,0.1,20])") ;
w.factory("FCONV::conv(e,phys,resol)") ;

RooDataSet* d = w.pdf("conv")->generate(*w.var("e"),1000) ;

RooFitResult* r = w.pdf("conv")->fitTo(*d,Save()) ;
```

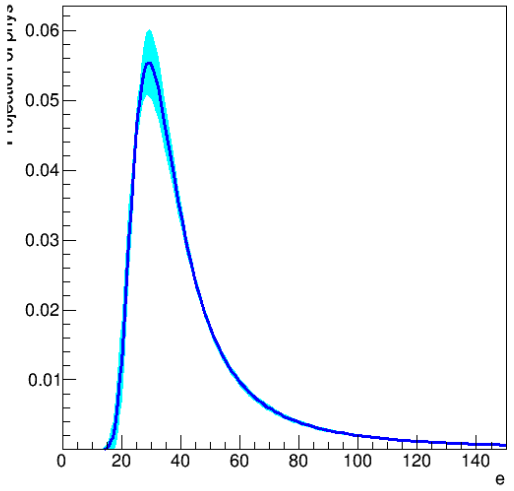
Observed distribution



Fitted kernel function



Fitted theory distribution



CPU time of fit = 400msec (1000 events, 53 likelihood evaluations)

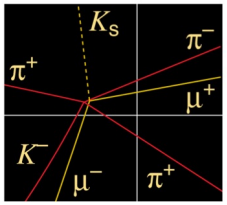
A more ambitious example – per-event errors

- In some cases kernel function (resolution model) depends on other observables y, but not on x → Can also model try to model that.
- Example per-event errors

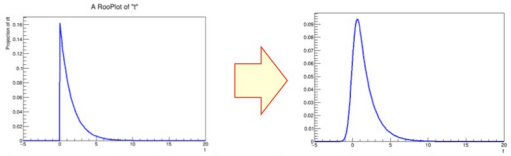
Resolution kernel depends on 2nd observable δt

$$f(t \mid \delta t) = \int f(t_{theo}) \cdot Gaussian(t - t_{theo}, b, \sigma \cdot \delta t)\, dt_{theo}$$

Case study – per-event errors



- Another common variant of this type of modeling problem is the so-called 'per-event' error
- Example: observable = decay time distribution, measured from reconstructed vertex.
  - In absence of a detector resolution, exponential decay distribution
  - In real life, distribution is convoluted with (Gaussian) reconstruction resolution



- But vertex reconstruction gives also estimate of uncertainty for every reconstructed vertex → the 'per-event error'
  - Can take this into account: well-reconstructed events carry more information
- How? Scale assumed resolution with per-event error

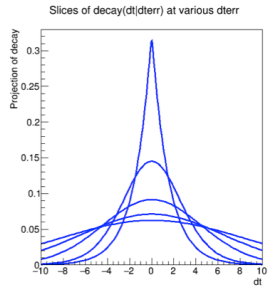
$$f(t \mid \delta t) = Decay(t) \otimes Gaussian(t, 0, \sigma \cdot \delta t)$$

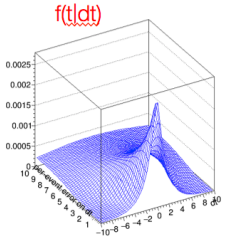
Case study – per-event errors

- Visualization of decay function with variable resolution

Decay function (symmetrized) convoluted with Gaussian resolution at 4 different values of per-event error

$$f(t \mid \delta t) = Decay(t) \otimes Gaussian(t, 0, \sigma \cdot \delta t)$$

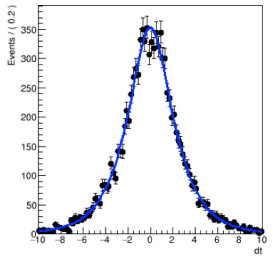




Gain: high-resolution events carrymore weight in likelihood → better estimate of model parameters

Full 2D-model:
F(t,dt) = F_1(t|dt)*F_2(dt)

Shown here: *projection* on t
F(t) = Int [F_1(t|dt)*F_2(dt)] dt



A more ambitious example – per-event errors

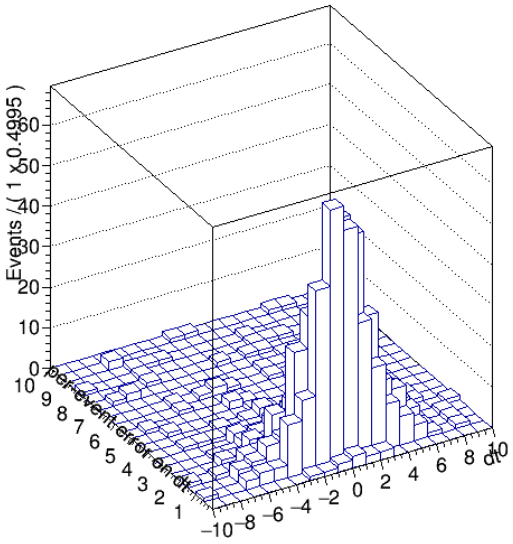
- Example fit with resolution kernel using per-event errors

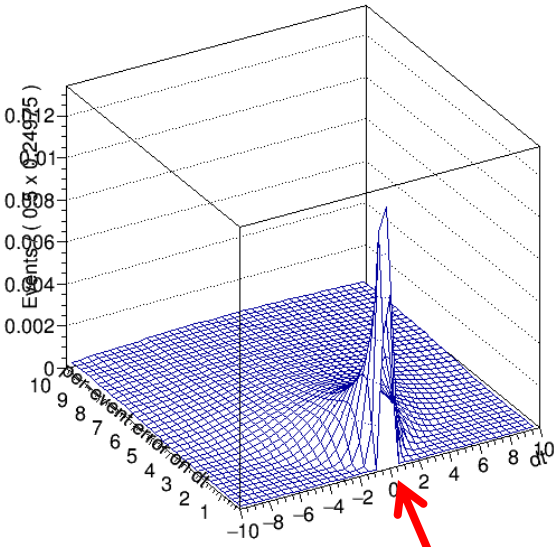
$$f(t \mid \delta t) = \int Gaussian(t - t_{theo}, b, \sigma \cdot \delta t) \cdot f(t_{theo}) dt_{theo}$$

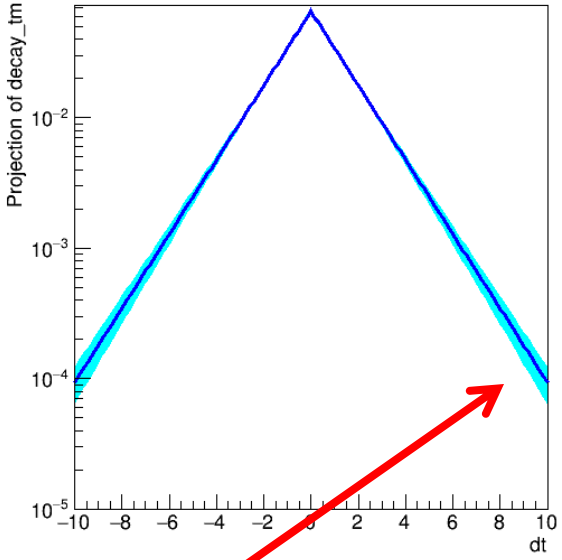
Observed distribution

Fitted kernel function

Fitted theory distribution







- For this physics example can fit all kernel parameters (b,σ) in addition to theory parameter τ

bias = 0.073 ± 0.055
sigma = 0.994 ± 0.083
tau = 1.525 ± 0.085

But it quickly gets very complicated

- Example 1: Convolution kernel constant in x_{reco}

$$f(x_{reco}) = \int Gaussian(x_{reco} - x_{theo}) \cdot f(x_{theo})\, dx_{theo}$$

- Example 2: Convolution kernel constant in x_{reco}, but depends on y_{reco}

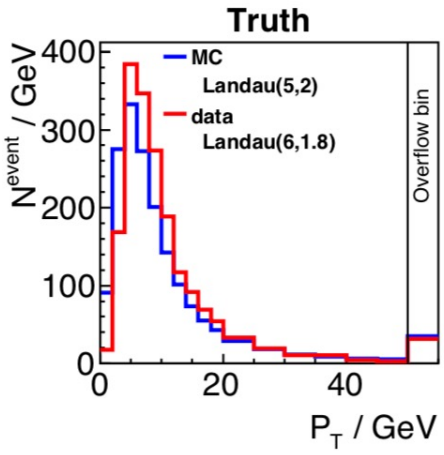
$$f(t \mid \delta t) = \int Gaussian(t - t_{theo}, b, \sigma \cdot \delta t) \cdot f(t_{theo}) dt_{theo}$$

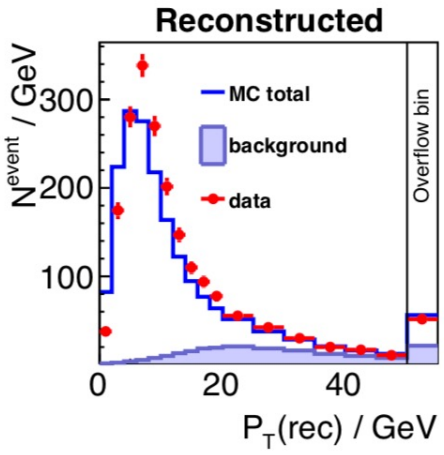
- What happens if resolution also varies in x_{reco}?
  - E.g. mass resolution depends on mass..
  - Then in very quickly becomes numerically very complex (hard-to-solve)
  - Numeric precision issues due to possible degeneracies.

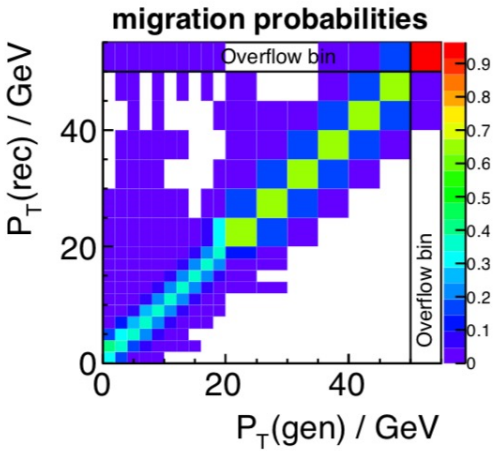
- Solution in that case is discretize model for x_{reco}
  in which case convolution *kernel* $K(x_{reco}, x_{theo})$ *becomes a matrix*
  - *Can no longer fit, but perform an unfolding procedure to obtain* $f(x_{theo})$

Unfolding – basic idea

- Unfolding basic idea – from simulation you know for each event both x_{reco} and x_{theo} → Use this to populate a response matrix K



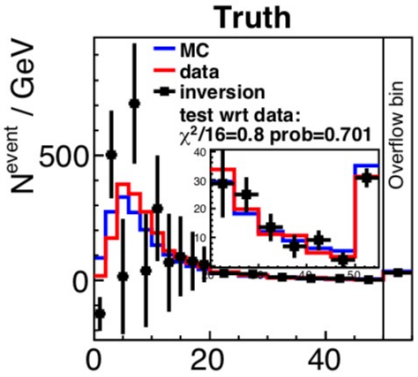


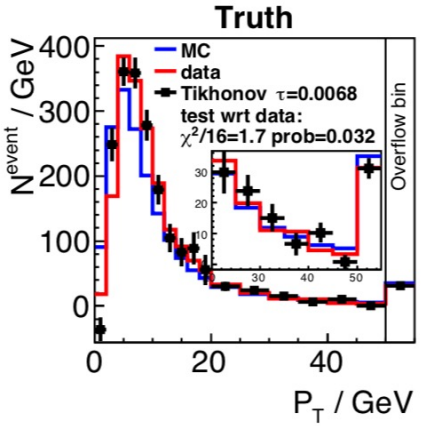


- You can then multiply the inverse of the response matrix (K^{-1}) with the observed distribution $f(X_{reco})$ to obtain the theory distribution $f(X_{theo})$

The Unfolding Industry

- There are a great number of unfolding algorithms 'out there'
  - Some historically in ROOT ('Tunfold')
  - Most in 'RooUnfold' (by Tim Adye)
- Algorithms make different assumptions to regulate intrinsic numerical difficulty of unfolding inversion problem
  - A bit more background on regularization approaches later
- Do all give equally good results?
  - And at what setting? Most algorithms have a tunable 'regularization' strength. When are we comparing apples to apples?
- Actually, what is a 'good result'?
  - Are smaller error bars on the unfolded data better?
  - What about biases? Can you unfold a (potentially BSM) data distribution with an unfolding matrix obtained from the SM without biasing the results to the SM?

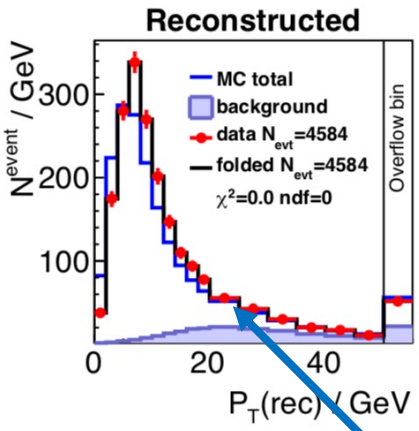


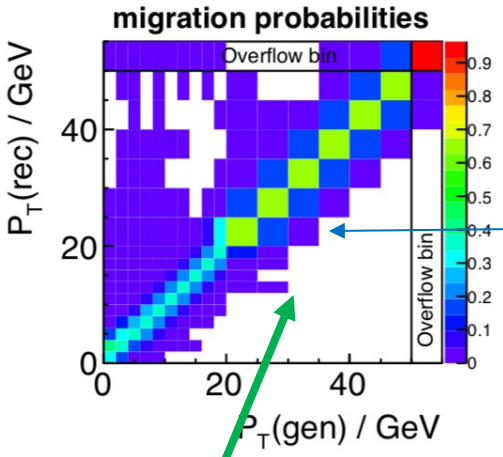


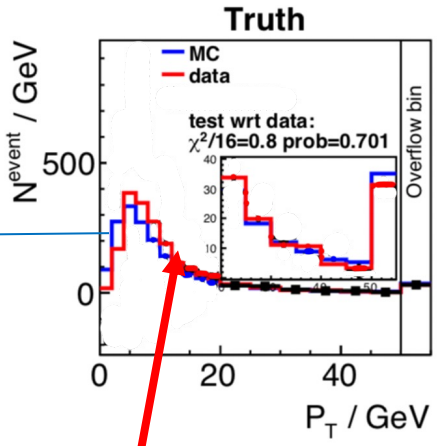
Folding and unfolding – the underlying mathematics

data ← truth

'folding' or 'convolution'







$$\nu_i = \sum_{j=1}^{M} R_{ij}\mu_j \,,$$

$$\hat{\vec{\mu}} = (\hat{\mu}_1, \ldots, \hat{\mu}_M).$$

'expected data ν_i'

→ **this is the prediction of the true histogram and the migration matrix ←**

'migration matrix'

→ **requires simulation with some assumed true model ←**

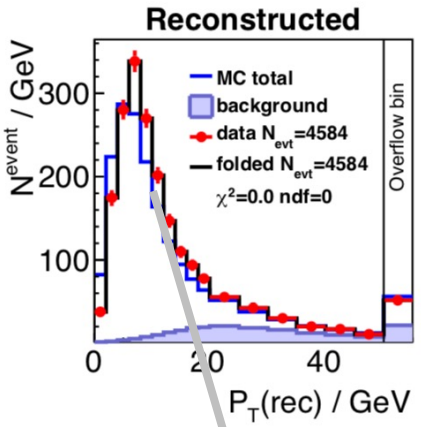
'true histogram'

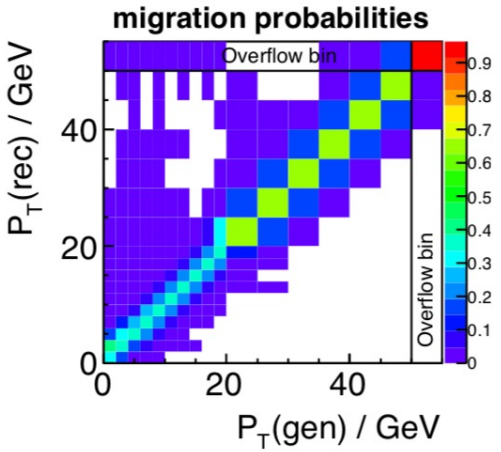
→ **this is what we are after ←**

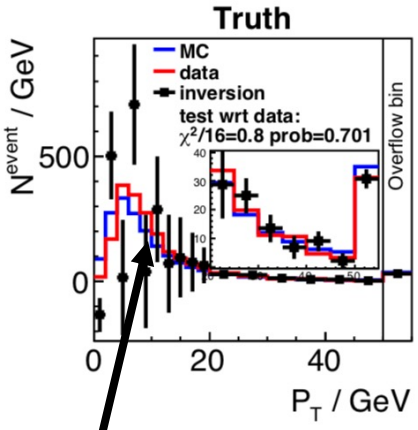
Folding and unfolding – the underlying mathematics

data → truth

'unfolding' or 'deconvolution'







'ML estimate of true histogram using L(n|μ)'

$$\hat{\vec{\mu}} = R^{-1}\vec{n}$$

'observed data n_i'

→ **this is what we measure and has stat. fluctuations←**

$$L(n|\mu) = \prod Poisson(n_i|\nu_i(\vec{\mu}))$$

$$\nu_i = \sum_{j=1}^{M} R_{ij}\mu_j \ ,$$

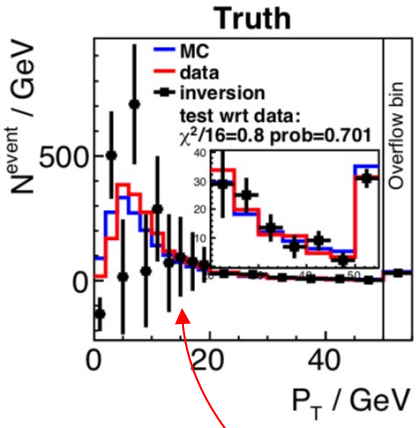
'expected data (given truth μ)'

Math is straightforwards – why is unfolding hard in practice?

- In principle – unfolding is a max.likelihood estimation problem

$$L(n|\mu) = \prod Poisson(n_i|v_i(\vec{\mu}))$$

$$\hat{\vec{\mu}} = R^{-1}\vec{n}$$



- ML estimator are
  - Unbiased → If you unfold many datasets you get the correct answer on average
  - Obeys Minimum Variance Bound → Estimated uncertainties ($\sigma_\mu == V_\mu^2$) are as small as they can be.
- Unfortunately, for most real-life problem MVB on uncertainties is very very loose → seriously large error bars (and fluctuations) common

Physicists always bring duct tape...

- Can we 'fudge' the problem a little bit so that it becomes more stable and gives smaller variances (scatter, error bars)?
- Yes – concept known as 'regularization'
- But it comes at a price! You will sacrifice 'no bias'
  - I.e. you will no longer get the correct answer on average (but this might be acceptable within limits)
- General idea – you bring in some extra information ('assumption') that will stabilize the numerical behavior
  - A) general notion of smoothness ('wild high-frequency oscillations should not happen')
  - B) 'target' truth distribution ('the unfolding data should look approximately like this')
- Approach B) will obviously bias unfolded result somewhat to 'target truth', but A) will invariably also result in some bias

Regularization through smoothness

- Most famous approach is 'Tikhonov' regularization.
  - General idea: add extra term to likelihood that is used for estimation of μ i.e minimize ϕ(μ) instead of L(μ)

$$\varphi(\vec{\mu}) = \ln L(\vec{\mu}) + \tau S(\vec{\mu}) \ ,$$

  - Extra term S(μ) 'rewards' smooth solutions. Tikhonov Ansatz:

$$S(\vec{\mu}) = -\sum_{i=1}^{M-2} (-\mu_i \ + \ 2\mu_{i+1} \ - \ \mu_{i+2})^2 .$$

→ *penalty if adjacent μ values differ strongly*

  - 'Regularization parameter' τ controls strength of regularization term

13. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed problems* (W.H. Winston, 1977).

Regularization through smoothness

- Another 'famous' approach is 'iterative Bayesian unfolding'
  - Uses Bayes formula to estimate Inverse of response matrix.

$$\nu_i = \sum_{j=1}^{M} R_{ij}\mu_j \ , \quad \Rightarrow \quad P(C_i|E_j) = \frac{P(E_j|C_i)\cdot P_0(C_i)}{\sum_{l=1}^{n_C} P(E_j|C_l)\cdot P_0(C_l)} \ .$$

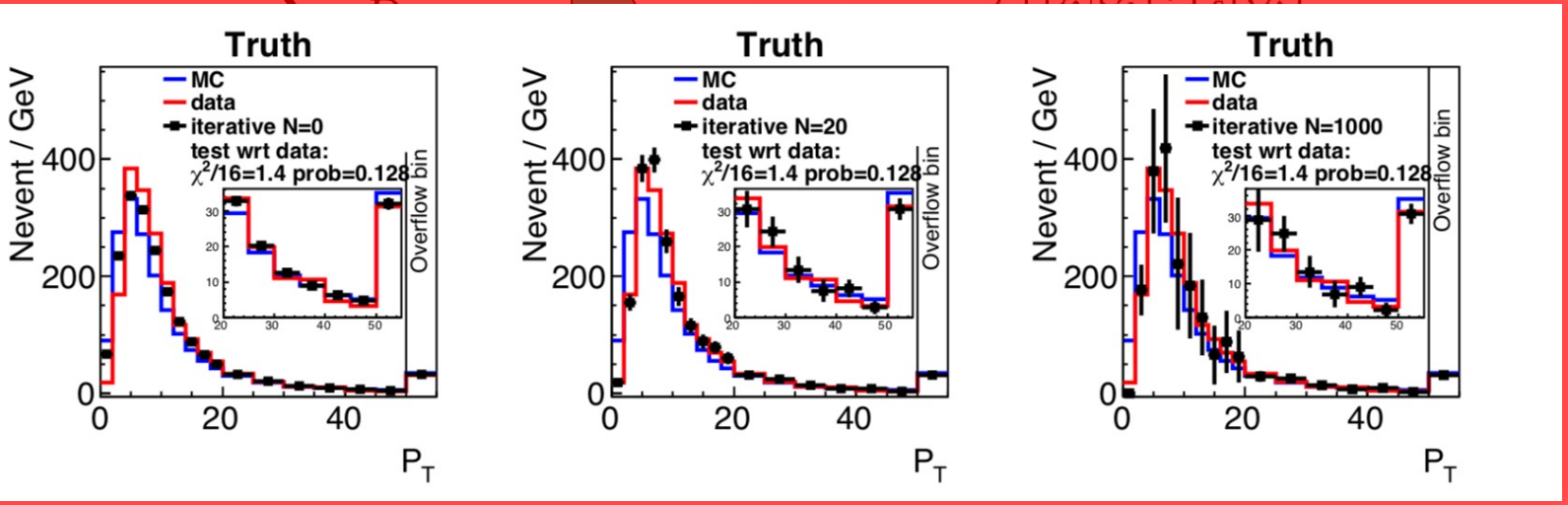
  - Iterative updating procedure of Ansatz for inverse of folding matrix will result in estimate of unfolded distribution

$$M_{ij} = \frac{P(E_j|C_i)\cdot P_0(C_i)}{[\sum_{l=1}^{n_E} P(E_l|C_i)]\cdot[\sum_{l=1}^{n_C} P(E_j|C_l)\cdot P_0(C_l)]} \ .$$

  - With infinite number of iterations will converge to ML estimator (which had the undesirable large variance)
- Why would solve problem iteratively, if matrix inversion gives analytical solution upfront?
  - Stop iterative updating earlier, fluctuations suppressed → effective regularization. Iteration count is effective regularization strength parameter

Regularization through smoothness

- Another ‘famous’ approach is ‘iterative Bayesian unfolding’
  - Uses Bayes formula to estimate Inverse of response matrix.



  - With infinite number of iterations will converge to ML estimator (which had the undesirable large variance)
  - If you stop iterative updating earlier, fluctuations suppressed → effective regularization. Iteration count is effective regularization strength parameter

What is the 'optimal' regularization strength?

- Regularization is a tradeoff of variance (scatter/uncertainty) vs bias
  - No universal correct choices
- 'Traditional' choice for this tradeoff is the one that *minimizes* the Mean Squared Error

$$\text{MSE(bin i)} = V[\mu_i] + b_i^2 = \sigma[\mu_i]^2 + b_i^2$$

i.e. you are willing to trade a 10% decrease in stat error for a 10% increase in bias

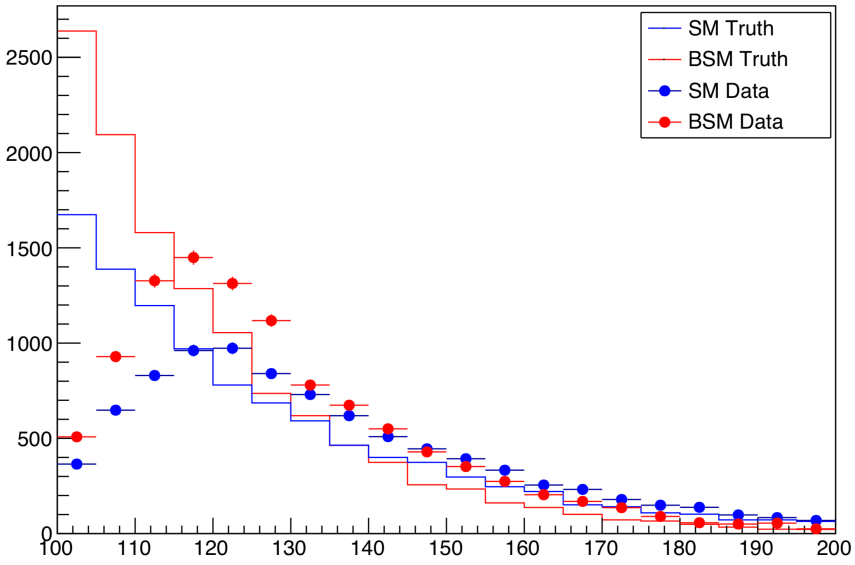
MSE is defined per bin – so in practice mimize *average* of MSE over bins

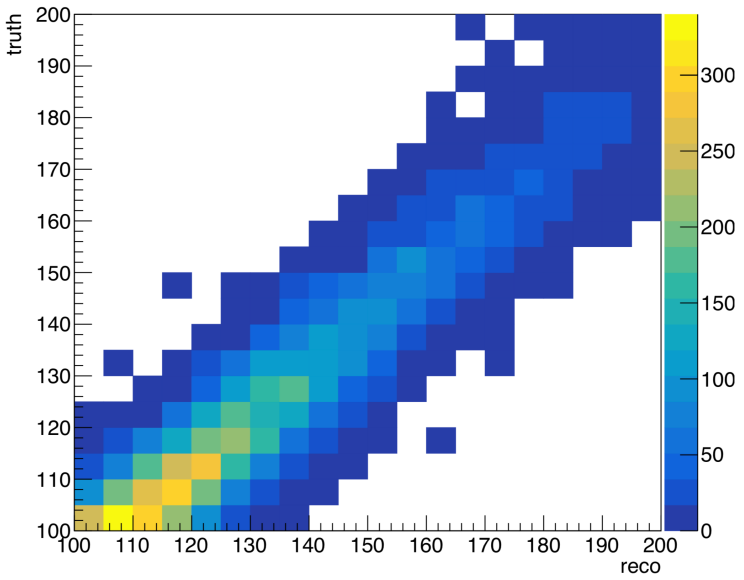
- MSE sounds 'reasonable' - Any other requirements?
  - Traditional HEP answer: 'no'
  - Professional statisticians: 'yes' – you also want good *coverage*

A benchmark test

- Consider a simple unfolding problem

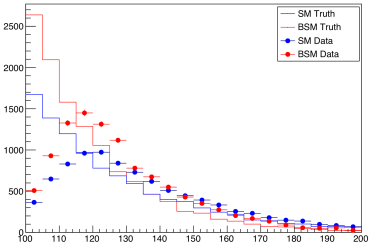
$$f(x|\alpha) = f_{\text{physics}}(x_{\text{true}}|\alpha) * f_{\text{detector}}(x_{\text{true}}, x)$$
$$= (\alpha \cdot \exp(-\alpha \cdot x_{\text{true}})) * \text{Gauss}\left(x - x_{\text{true}}, 7.5, 0.5 \cdot \sqrt{x_{\text{true}}} + 2.5\right),$$



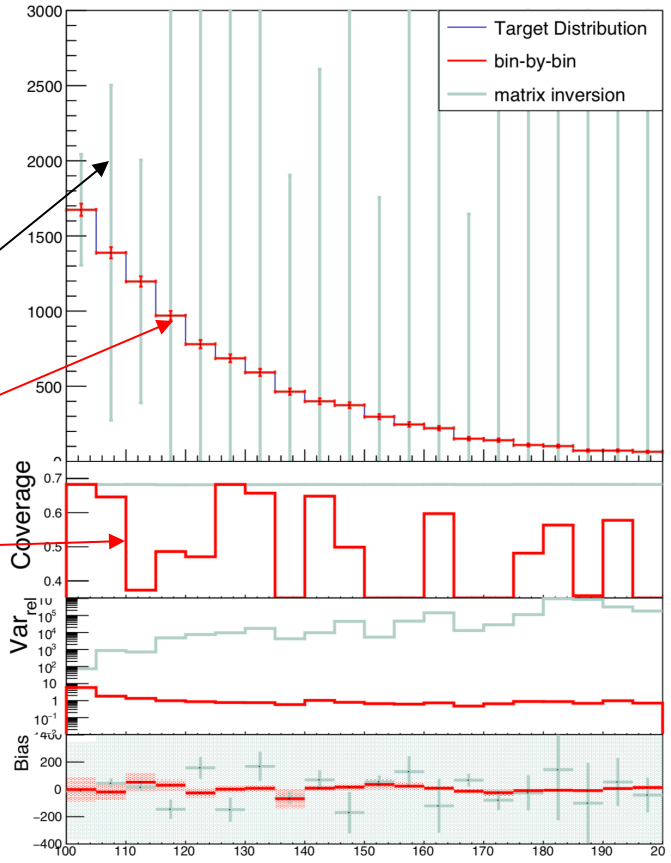


- Consider two truth models: SM (α=0.035) and BSM(α=0.05)
  - See plots above. The migration matrix is *always* constructed from SM events

How well do unfolding algorithms perform?

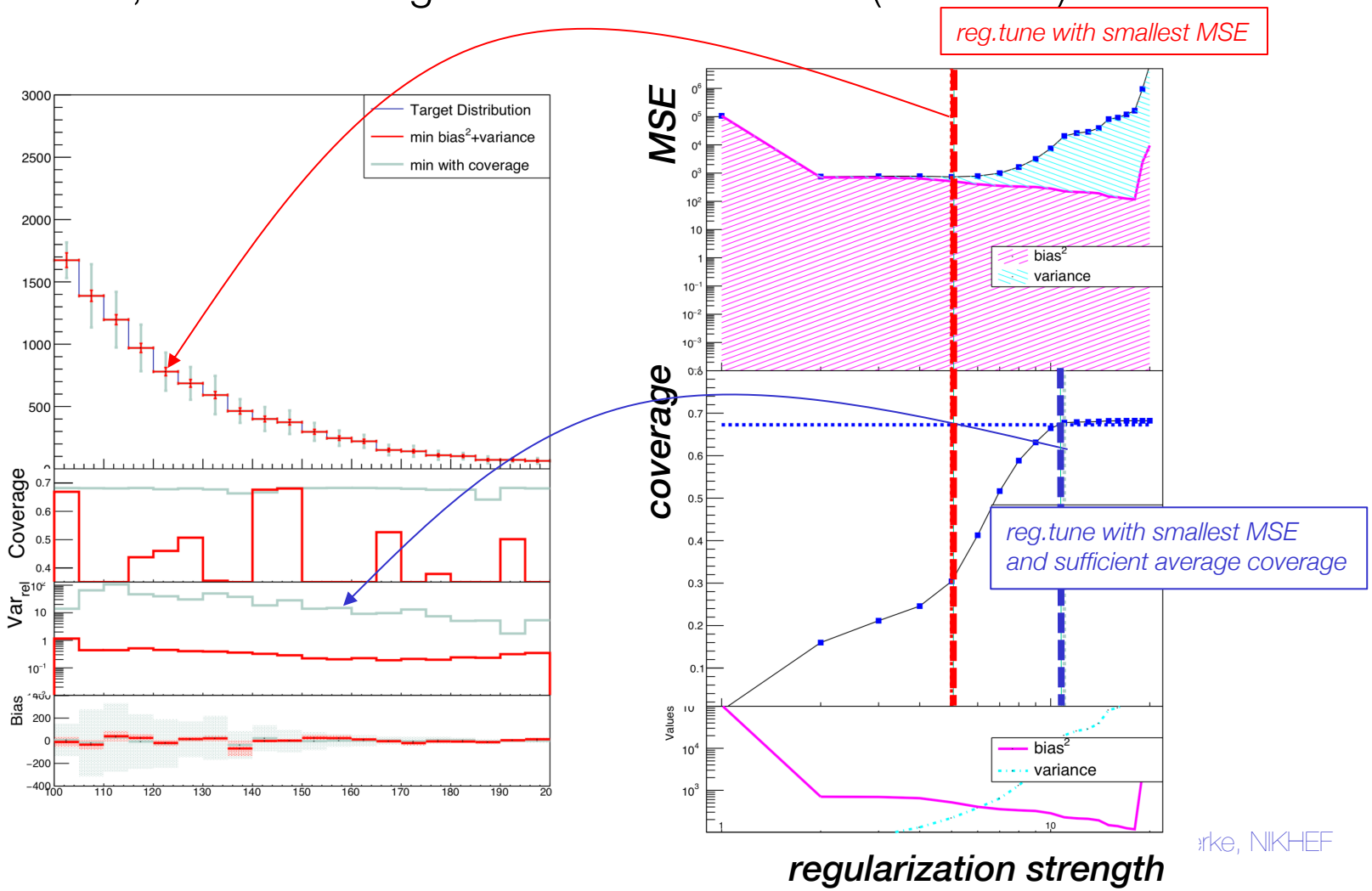


- *Simplest case*: unfold SM data with SM migration matrix
- First consider simple unregularized unfolding algorithms: Matrix Inversion & Bin-by-Bin unfolding
- The good: no bias
  - *Because* they are unregularized
- The bad: terrible variance
  - Huge error bars on matrix inversion (but perfect coverage!)
  - Tiny error bars for bin-by-bin (too small → poor coverage)
- Unfolding is hard!
  - Even for seemingly simple problems



How well do unfolding algorithms perform?

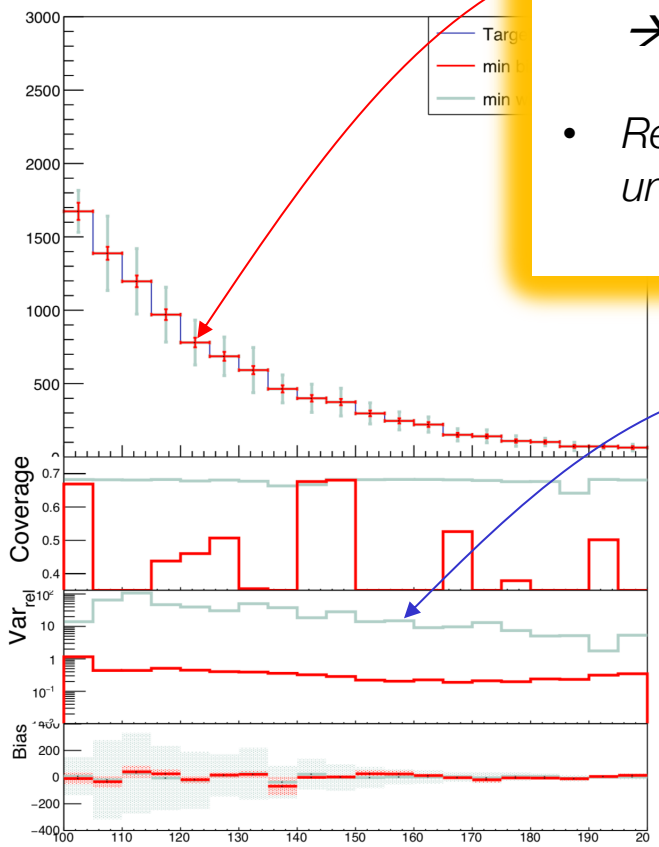
- *Simplest case*: unfold SM data with SM migration matrix
- Next, consider a regularized method: SVD (Tikhonov)

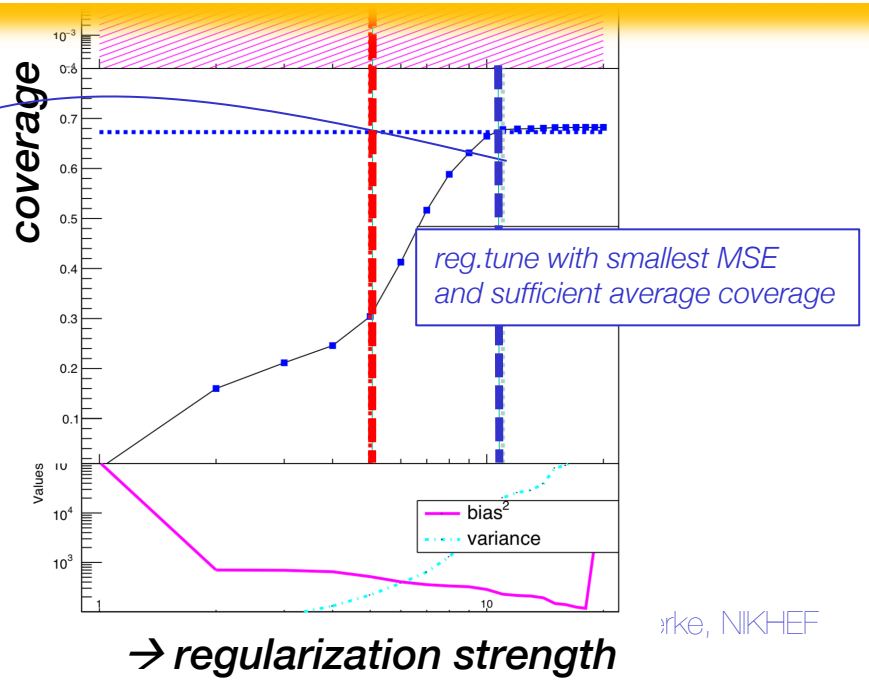


How well do unfolding algorithms perform?

- *Simplest case*: unfold SM data with SM migration matrix
- Next, consider a met

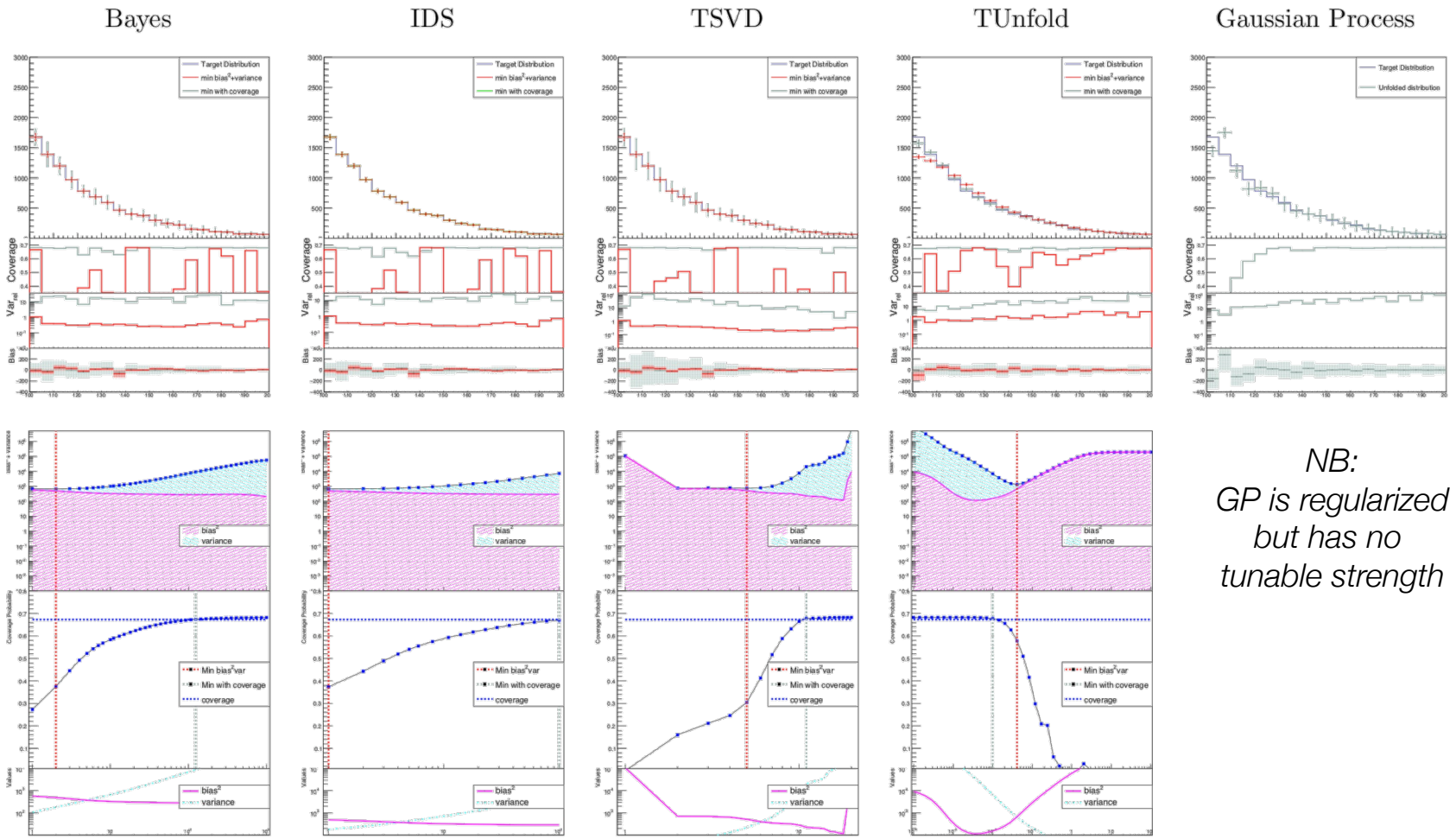
- *Optimizing regularization strength on MeanSquaredError results in strong undercoverage*
  → Error bars are too small!
- *Requiring good <u>average</u> coverage in tuning results in uniformly good coverage for all bins*





→ ***regularization strength***

A comparison of 5 regularized methods



NB:
GP is regularized but has no tunable strength

1) Optimizing on MSE always results in poor coverage (error bars too small)
2(Requiring good average coverage results in uniformly good coverage

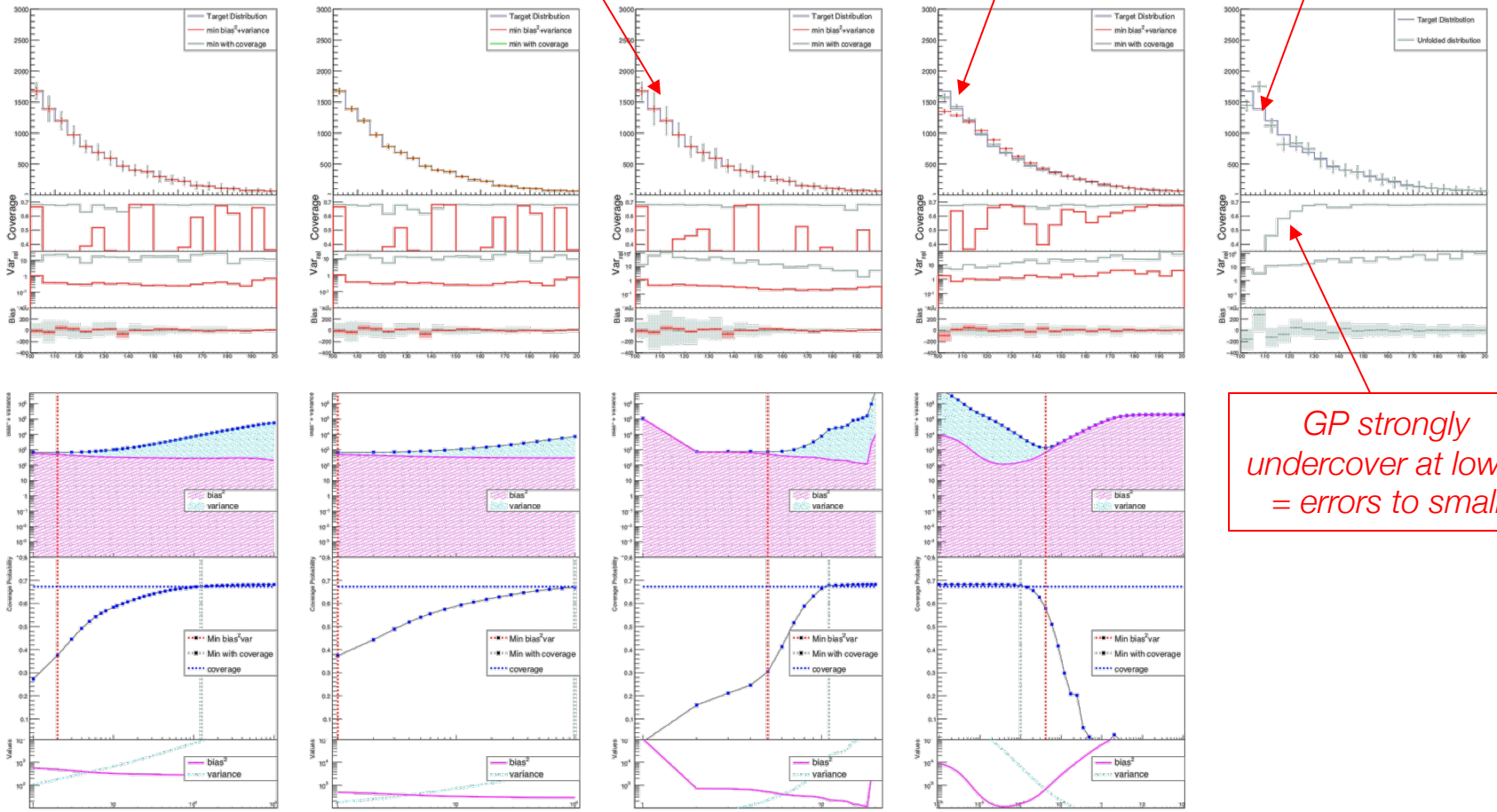
A co ularized metho

SVD has largest errors when coverage is demanded

TUnfold biased at low x values

GP has biases at low x due to boundary effects

Bayes | IDS | TSVD | TUnfold | Gaussian Process



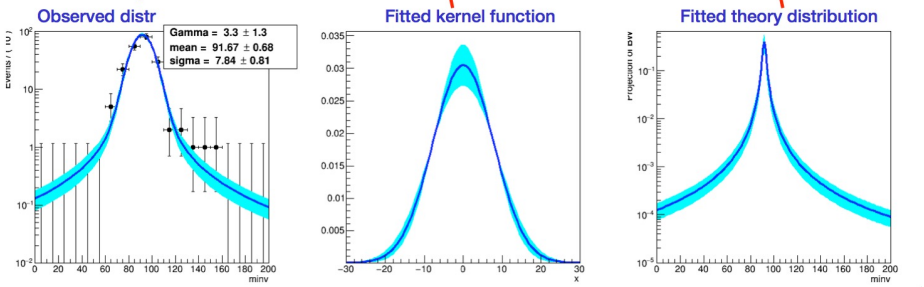
GP strongly undercover at low x = errors to small

Recap on convoluted signal extraction

- If response function of the detector is relatively simple,
  e.g. ~constant as function truth observable

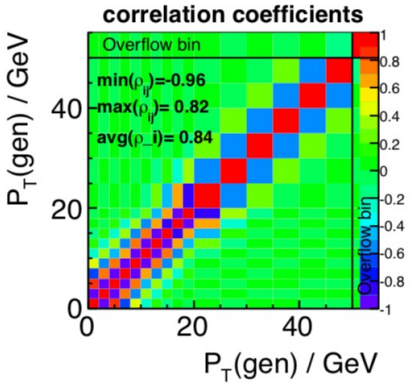
  → Can fit data directly with a convolution model
  → Result is a fitted ‘unfolded’ truth distribution at the theory level
  → (Some) parameters of convolution kernel may be fitted simultaneously

$$f(m_{inv} \mid M,\Gamma,b,\sigma) = \int Gaussian(m_{inv}^{th} - m_{inv}^{reco}, b, \sigma) \cdot BreitWigner(m_{inv}^{th} \mid \Gamma, M) dm_{inv}^{th}$$



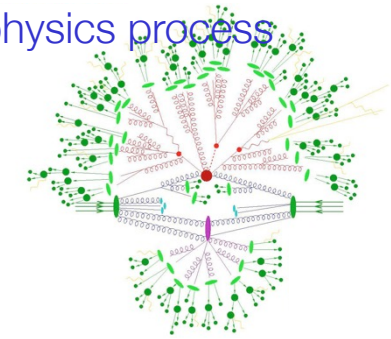
- If response function of detector is complex,
  ML estimation numerically not tractable/stable

  → If truth-level distribution is needed,
  discretized unfolding ‘after burner’ needed.
  → Despite discretization numerically still challenging!

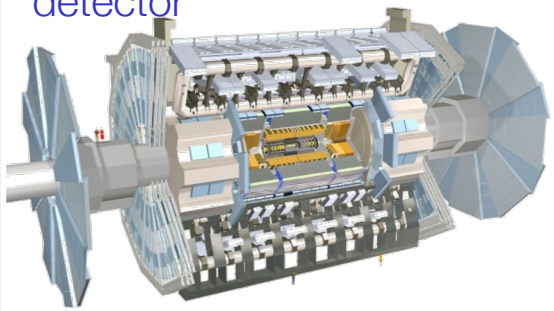


The experiment as convolution – one step back…

Simulation of ‘soft physics’ physics process

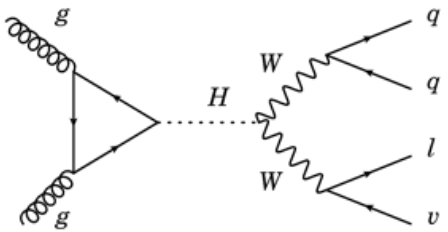


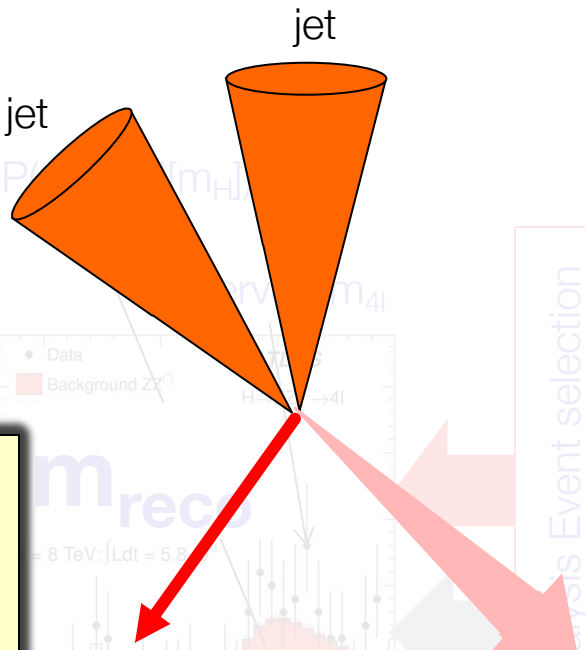
Simulation of ATLAS detector



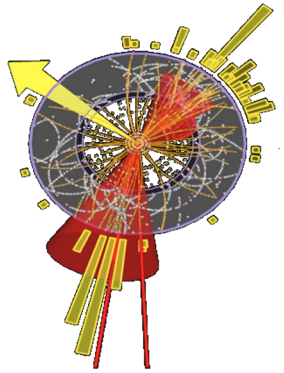
LHC data

Simulation of high-energy physics process





Reconstruction of ATLAS detector

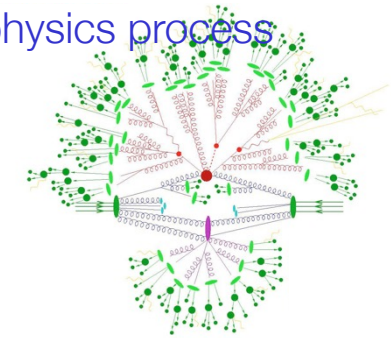


What if we scale back our ambition one notch formulate convolution and take **4-vectors of reco objects as observables** in convolution?

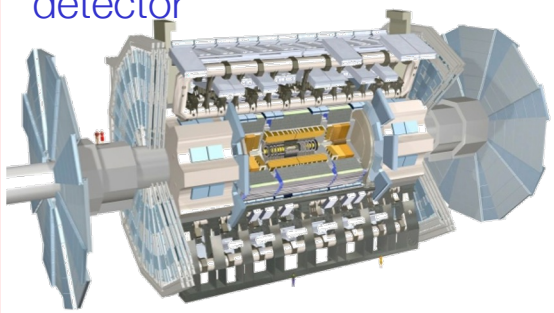
$$f(x_{reco}) = \int f(x_{theo}) K(x_{reco}, x_{theo}) dx_{theo}$$

The experiment as convolution – one step back…

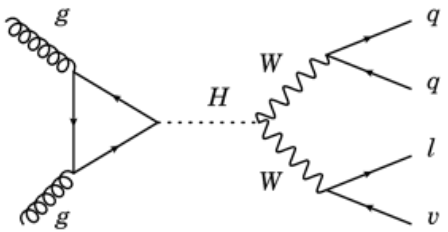
Simulation of 'soft physics' physics process

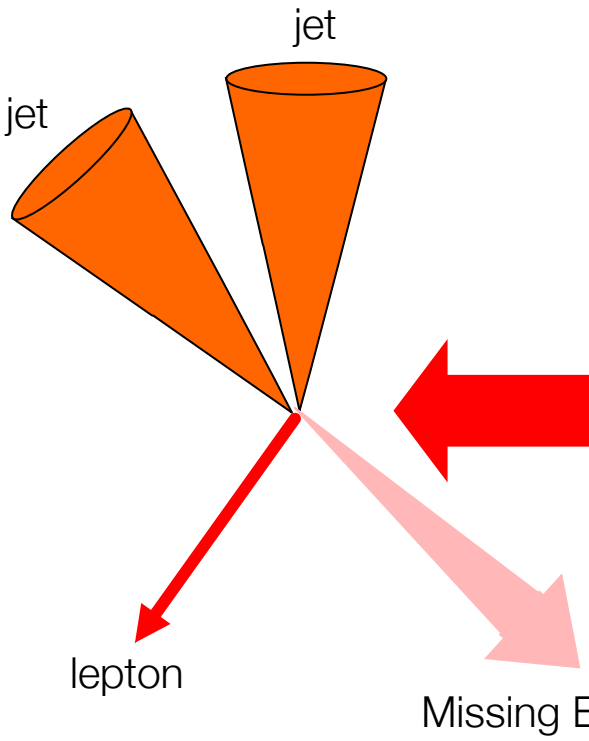


Simulation of ATLAS detector

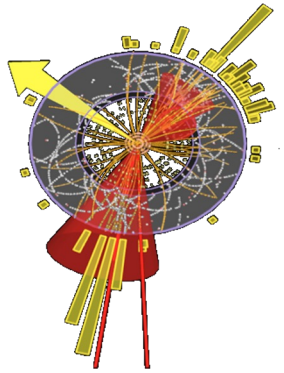


Simulation of high-energy physics process





Reconstruction of ATLAS detector



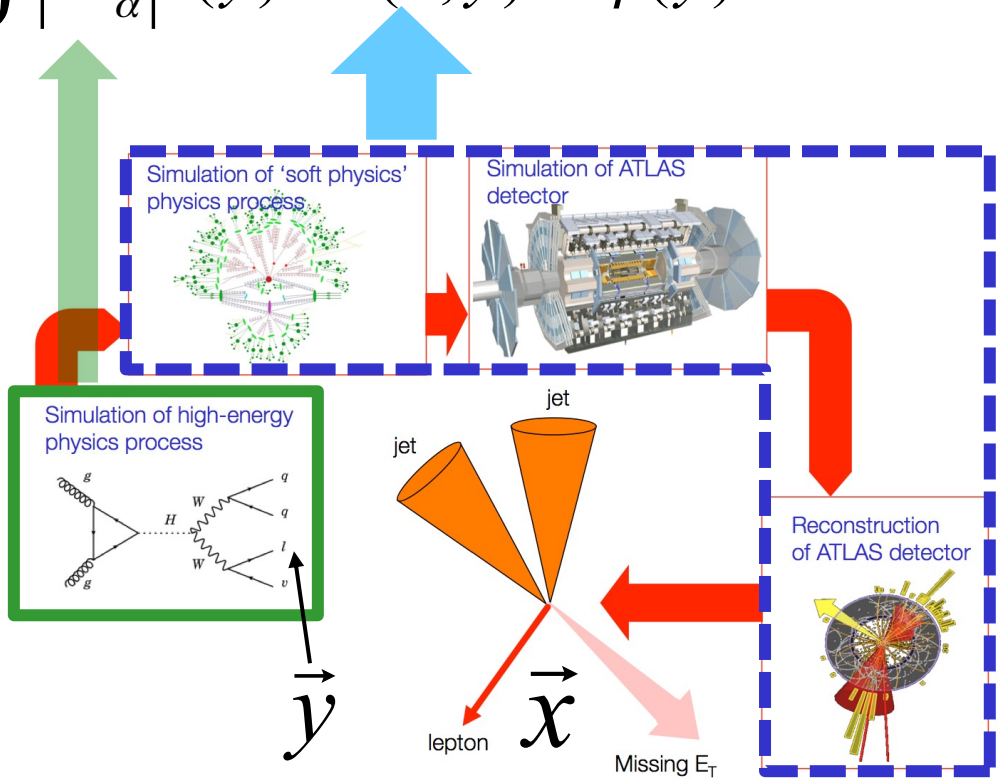
The detector as convolution

- Then convolution can be written in terms of parton kinematics

$$f(x_{reco}) = \int f(x_{theo}) K(x_{reco}, x_{theo}) dx_{theo}$$

$$f_\alpha(\vec{x}) = \frac{1}{\sigma} \int \left| M_\alpha \right|^2 (\vec{y}) \cdot W(\vec{x}, \vec{y}) \cdot d\varphi(\vec{y})$$

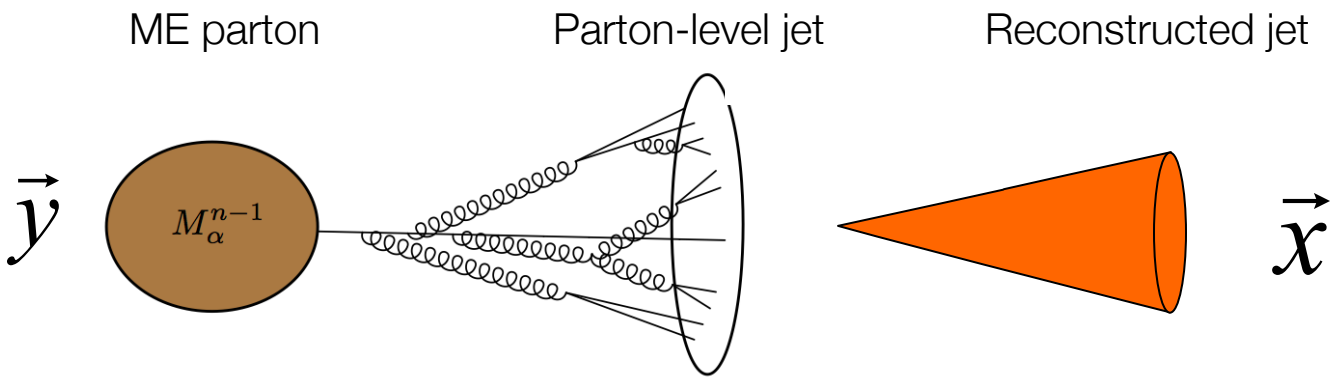
phase-space factor



The detector as convolution – matrix element methods

- The transfer function W(x,y) maps *parton kinematics* onto *reco kinematics*

$$f_\alpha(\vec{x}) = \frac{1}{\sigma}\int \left|M_\alpha\right|^2(\vec{y})\cdot W(\vec{x},\vec{y})\cdot d\varphi(\vec{y})$$



- The transfer function factorizes by parton. Usually it is further approximated to also kinematically factorize in terms for E,ϕ,y

$$W(\vec{x},\vec{y}) = \prod_{partons} W_i(\vec{x}_i,\vec{y}_i) = \prod_{partons} W_i^E(E_i^{reco},E_i^{part})\cdot W_i^E(\varphi_i^{reco},\varphi_i^{part})\cdot W_i^E(y_i^{reco},y_i^{part})$$

typically Gaussian models

The detector as convolution – matrix element methods

- The transfer function W(x,y) maps *parton kinematics* onto *reco kinematics*

$$f_\alpha(\vec{x}) = \frac{1}{\sigma}\int \left|M_\alpha\right|^2(\vec{y})\cdot W(\vec{x},\vec{y})\cdot d\varphi(\vec{y})$$

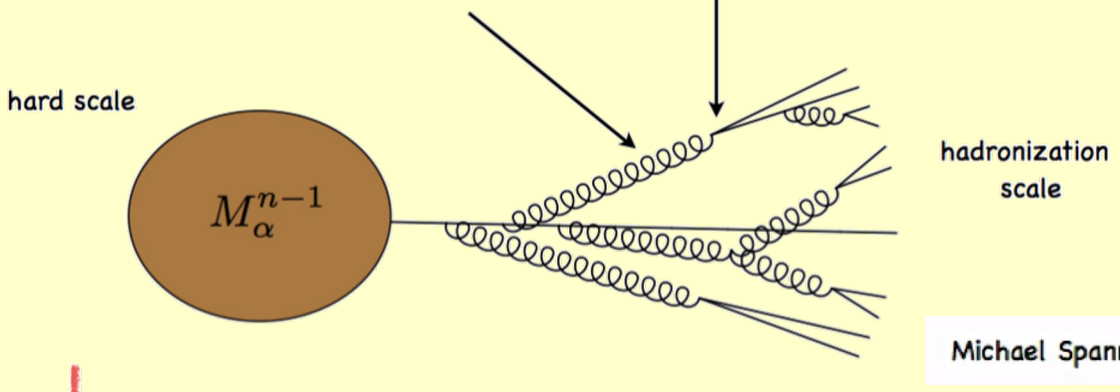
The hard part of W(x,y) is dealing with gluon splittings
(makes mapping of partons to truth jets fuzzy)

Factorization of emissions in soft/collinear limit

and Sudakov factors allow semiclassical approximation of quantum process:

propagator-lines = Sudakov factors

vertices = Splitting functions



Can calculate weight for shower history iteratively

Can use smaller objects and more objects (more information)

-

y

$, y_i^{part})$

typically Gaussian models

What can you do MEM models?

- The output of the Matrix Element Method is a probability model for events of a fixed reco-level topology (e.g. 2 jets, 1 lepton, MET) under a physics process hypothesis (e.g. pp→H→WW→qqlv)
- With MEM models for multiple hypothesis, can do hypothesis testing (event selection) in an Neyman-Pearson optimal way

The Neyman-Pearson lemma

- In 1932-1938 Neyman and Pearson developed a theory in which one must consider competing hypotheses
  - Null hypothesis (H_0) = Background only
  - Alternate hypotheses (H_1) = e.g. Signal + Background

and proved that

- The region W that minimizes the rate of the type-II error (not reporting true discovery) is a contour of the Likelihood Ratio

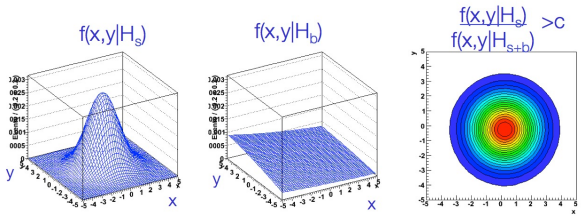
$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

- Any other region of the same size will have less power

Wouter Verkerke,

The Neyman-Pearson lemma

- Example of application of NP-lemma with two observables



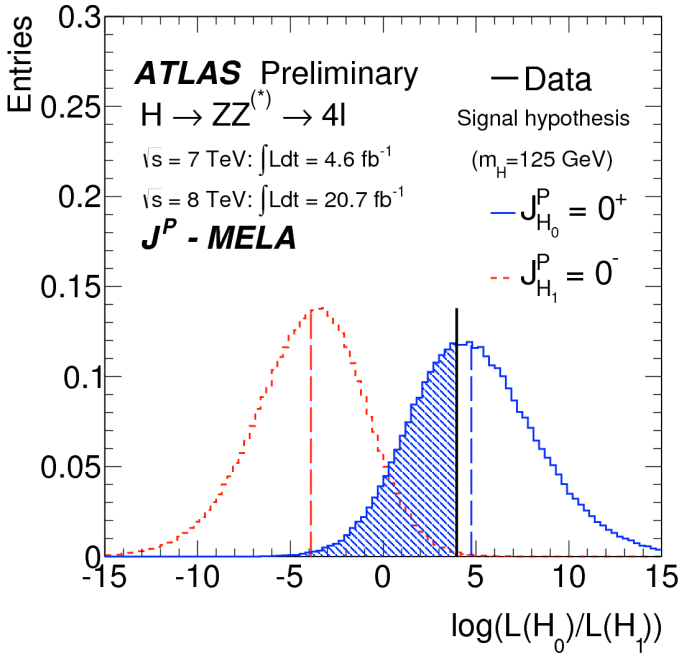
- Cut-off value c controls type-I error rate ('size' = bkg rate) Neyman-Pearson: LR cut gives best possible 'power' = signal eff.
- So why don't we *always* do this? (instead of training neural networks, boosted decision trees etc)

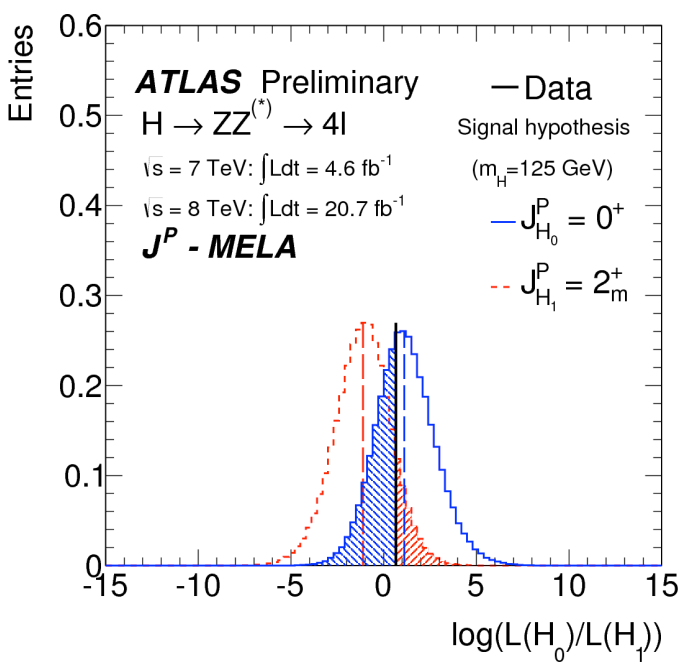
The problem is that we usually don't have explicit formulae for the pdfs $f(\vec{x}|\mathrm{s})$, $f(\vec{x}|\mathrm{b})$.

But with MEM we do!

Separate signal from background, or spin-0 from spin-2

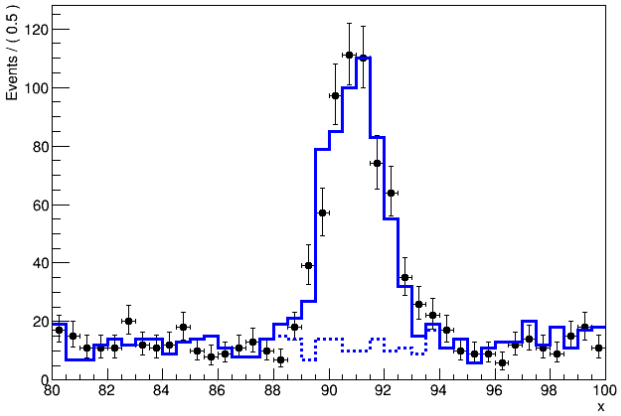
- Ratio of MEM probability models is Neyman-Pearson optimal discriminant between two hypothesis
  - Can be signal vs background
  - Can also be two different types of signal
- Example MEM applications in HEP: ATLAS H→ZZ decays:
  - Comparison between pairs in Higgs spin/parity states 0^+, 0^-, 2^+





Amplitude extraction – exploiting quantum mechanics

- Theory parameters entangled with the detector response are difficult as some form deconvolution is needed – typically masses, lifetimes etc
- But also plenty of theory parameters that are effectively amplitudes → more like cross-sections (easy) but not quite
- Cross-sections of processes that arise from a single Matrix Element are 'easy' → p.d.f in discriminating observable with signal and background component



- What if 2 or more ME amplitudes contribute? → interference!

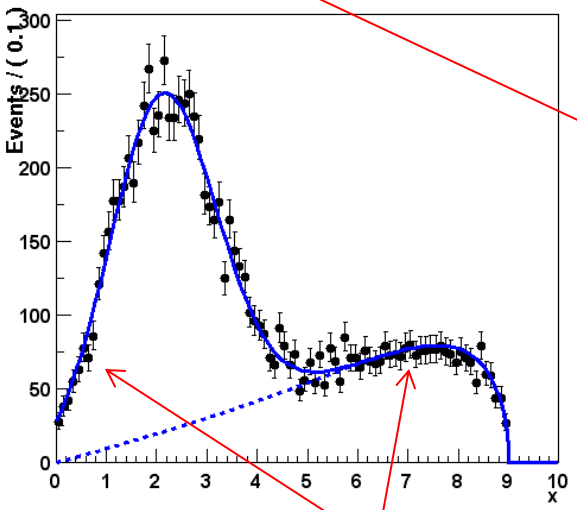
Amplitude modeling – the basics

- When it's possible to formulate p.d.f.s for observable distributions analytically, constructing probability models that sum physics amplitudes with interference effects is straightforward

Sum of p.d.f.s

Components are pdfs

$$f(x \mid s,b) = \frac{s}{s+b} f_s(x) + \frac{b}{s+b} f_b(x)$$



Each component is positive definite

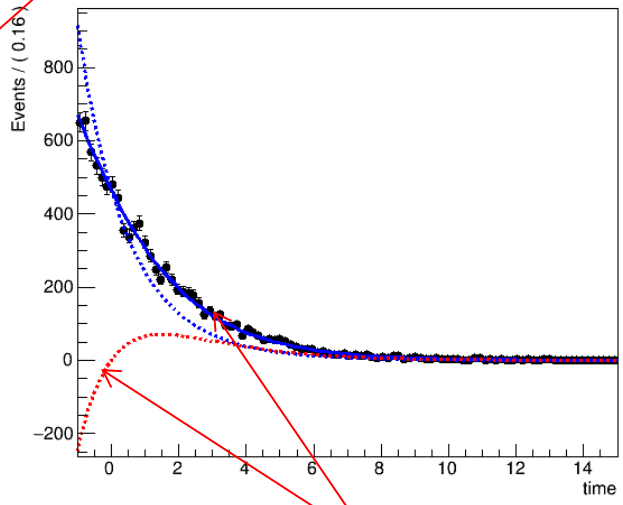
Both models are p.d.f.s

Sum of amplitudes

Components are functions

$$f(x \mid \vec{c}) = \frac{\sum_i c_i \cdot H_i(x)}{\sum_i c_i \cdot \int H_i(x)\,dx}$$

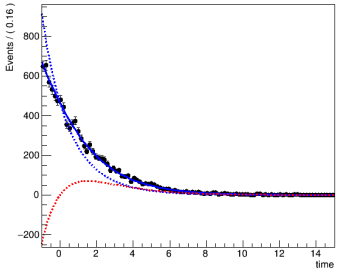
Explicit normalization required



Components can be negative, sum must be positive definite

Amplitude modeling – amplitudes from MC generators

- In many LHC analyses, observable distributions can only be obtained from MC simulation chain. *If so, this is also true for amplitudes*
- How can we simulate observable distributions corresponding to individual amplitudes?
  - Requires some support in MC generators → ability to selectively enable/disable individual amplitudes
  - Amplitudes can be negative → How does this translate to an MC event sample? Potential difficulty (e.g. allow for negative event weights)
- But otherwise straightforward – no complex deconvolution or template morphing needed to deform template histograms
  - Every physics model with (only) amplitude parameters can always be described as a weighted sum of amplitude templates (histograms!)



$$f(x \mid \vec{c}) = \frac{\sum_i c_i \cdot H_i(x)}{\sum_i c_i \cdot \int H_i(x)\,dx}$$

Generating a probability model from a Lagrangian

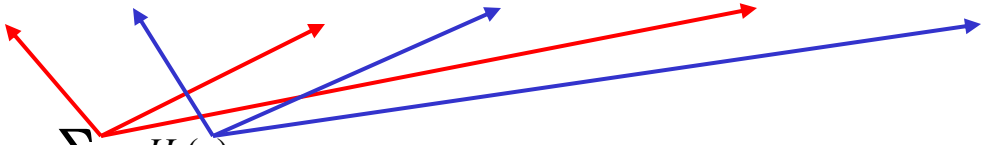
- Given a Lagrangian describing a Field Theory → can now model any transition amplitude (pp → X → Y) as a sum of real-valued amplitudes
- Consider example with two operators labeled SM and BSM, with strengths g_{SM} and g_{BSM} respectively. Matrix Element is

$$\mathcal{M}(g_{SM}, g_{BSM}) = g_{SM} O_{SM} + g_{BSM} O_{BSM}$$

- Transition amplitude is $|M|^2$:

$$T(g_{SM}, g_{BSM}) \propto |\mathcal{M}(g_{SM}, g_{BSM})|^2$$

$$|\mathcal{M}(g_{SM}, g_{BSM})|^2 = g_{SM}^2 |O_{SM}|^2 + g_{BSM}^2 |O_{BSM}|^2 + 2 g_{SM} g_{BSM} \mathcal{R}(O_{SM}^* O_{BSM})$$



$$f(x|\vec{c}) = \frac{\sum_i c_i \cdot H_i(x)}{\sum_i c_i \cdot \int H_i(x)\,dx}$$

Mapping of Wilson coefficients to template scale factors Wouter Verkerke, NIKHEF

The mapping of templates to operators

- Note that templates do not need to correspond one-to-one to single operators or pure interference terms

$$|\mathcal{M}(g_{SM}, g_{BSM})|^2 = g_{SM}^2 |O_{SM}|^2 + g_{BSM}^2 |O_{BSM}|^2 + 2 g_{SM} g_{BSM} \mathcal{R}(O_{SM}^* O_{BSM})$$

- For 2 operator, any three independent pairs of g_{SM},g_{BSM} values can generate templates that will span the whole parameter space. E.g.

$$T_{in}(1,0) \propto |O_{SM}|^2$$
$$T_{in}(0,1) \propto |O_{BSM}|^2$$
$$T_{in}(1,1) \propto |O_{SM}|^2 + |O_{BSM}|^2 + 2\mathcal{R}(O_{SM}^* O_{BSM})$$

$$|\mathcal{M}(g_{SM}, g_{BSM})|^2 = \underbrace{(g_{SM}{}^2 - g_{SM} g_{BSM})}\, T_{in}(1,0) + \underbrace{(g_{BSM}{}^2 - g_{SM} g_{BSM})}\, T_{in}(0,1) + \underbrace{g_{SM} g_{BSM}}\, T_{in}(1,1)$$

The mapping of t

- |M

- F

g

Note that in this choice for T_{in} there are
no templates corresponding to pure interference terms

→ All templates are positive definite!
even though the underlying templates are not (necessarily)

→ This result is general: for any amplitude sum there is
always a configuration of templates that are all positive

BSM)

n

space. E.g.

$$T_{in}(1,0) \propto |O_{\mathrm{SM}}|^2$$

$$T_{in}(0,1) \propto |O_{\mathrm{BSM}}|^2$$

$$T_{in}(1,1) \propto |O_{\mathrm{SM}}|^2 + |O_{\mathrm{BSM}}|^2 + 2\mathcal{R}(O_{\mathrm{SM}}^* O_{\mathrm{BSM}})$$

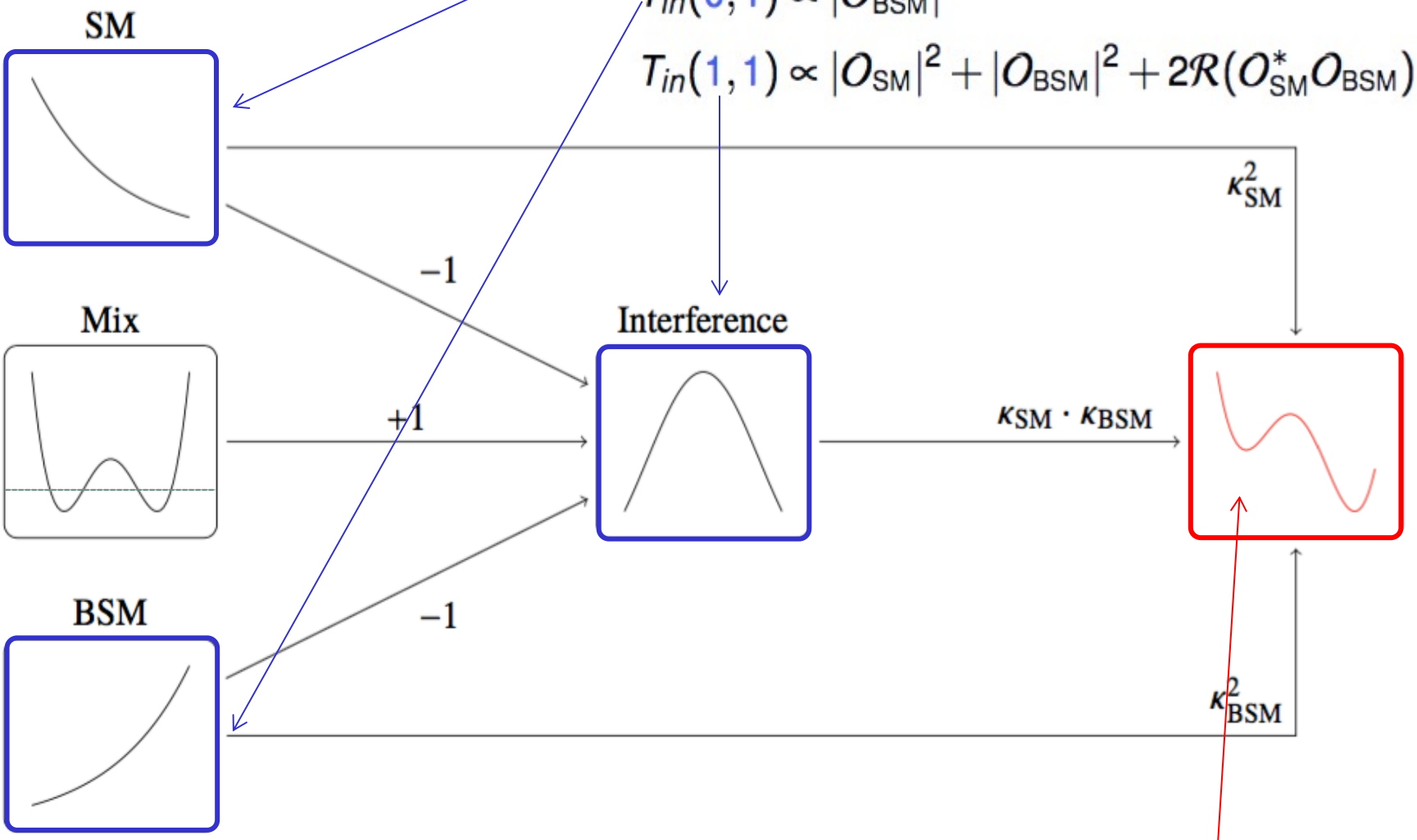
$$|\mathcal{M}(g_{\mathrm{SM}}, g_{\mathrm{BSM}})|^2 = \underbrace{(g_{\mathrm{SM}}{}^2 - g_{\mathrm{SM}} g_{\mathrm{BSM}})}\, T_{in}(1,0) + \underbrace{(g_{\mathrm{BSM}}{}^2 - g_{\mathrm{SM}} g_{\mathrm{BSM}})}\, T_{in}(0,1) + \underbrace{g_{\mathrm{SM}} g_{\mathrm{BSM}}}\, T_{in}(1,1)$$

Rearranged amplitude sums

$$T_{in}(1,0) \propto |O_{SM}|^2$$

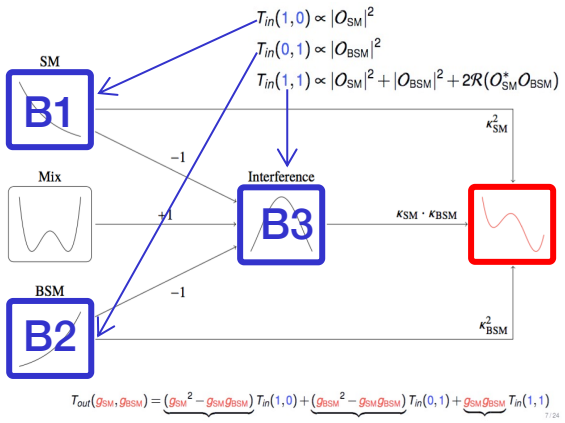
$$T_{in}(0,1) \propto |O_{BSM}|^2$$

$$T_{in}(1,1) \propto |O_{SM}|^2 + |O_{BSM}|^2 + 2\mathcal{R}(O_{SM}^* O_{BSM})$$

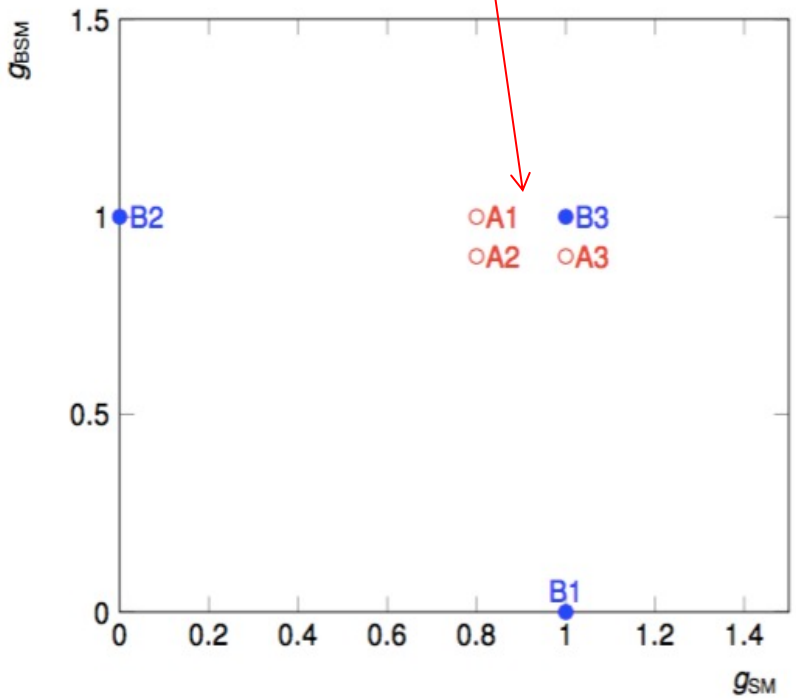


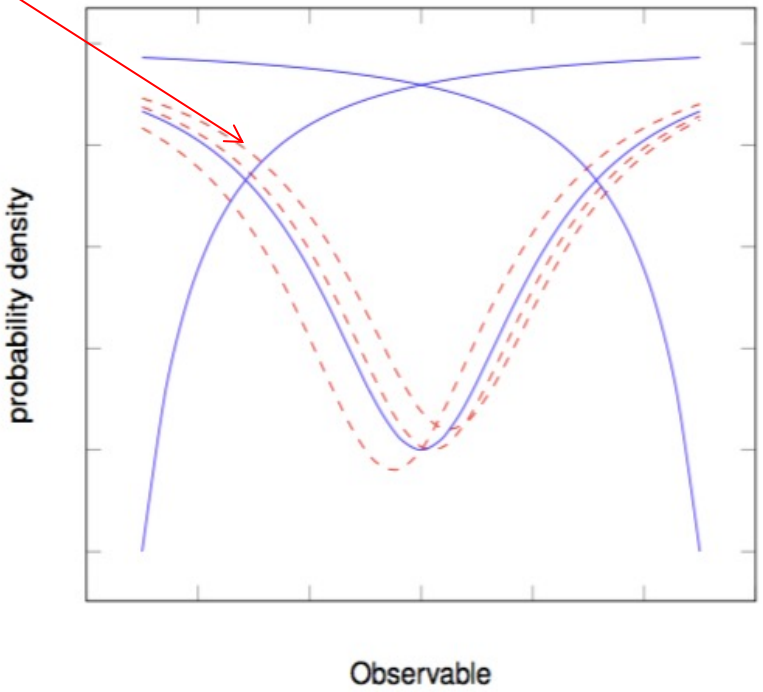
$$T_{out}(g_{SM}, g_{BSM}) = \underbrace{(g_{SM}{}^2 - g_{SM} g_{BSM})}\, T_{in}(1,0) + \underbrace{(g_{BSM}{}^2 - g_{SM} g_{BSM})}\, T_{in}(0,1) + \underbrace{g_{SM} g_{BSM}}\, T_{in}(1,1)$$

Rearranged amplitude sums



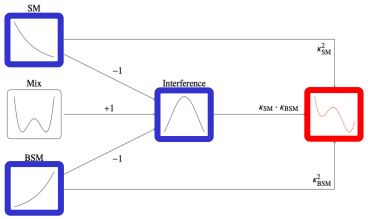
A1,A2,A3





The mapping of templates to operators

- Generalizing further, we will now work with 3 templates T_1, T_2, T_3 that are sampled at arbitrary points in the (g_{SM}, g_{BSM}) parameter space



$$T_{\text{in}}(1,0) \qquad T_{\text{in}}(g_{\text{SM},1}, g_{\text{BSM},1})$$
$$T_{\text{in}}(0,1) \qquad T_{\text{in}}(g_{\text{SM},2}, g_{\text{BSM},2})$$
$$T_{\text{in}}(1,1) \qquad T_{\text{in}}(g_{\text{SM},3}, g_{\text{BSM},3}).$$

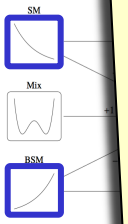
- The output probability model then takes the general form

$$T_{\text{out}}(g_{\text{SM}}, g_{\text{BSM}}) = \underbrace{(g_{\text{SM}}^2 - g_{\text{SM}}g_{\text{BSM}})}_{=w_1} T_{\text{in}}(1,0) + \underbrace{(g_{\text{BSM}}^2 - g_{\text{SM}}g_{\text{BSM}})}_{=w_2} T_{\text{in}}(0,1) + \underbrace{g_{\text{SM}}g_{\text{BSM}}}_{=w_3} T_{\text{in}}(1,1).$$

$$\begin{aligned} T_{\text{out}}(g_{\text{SM}}, g_{\text{BSM}}) = & \underbrace{(a_{11}g_{\text{SM}}^2 + a_{12}g_{\text{BSM}}^2 + a_{13}g_{\text{SM}}g_{\text{BSM}})}_{w_1} T_{\text{in}}(g_{\text{SM},1}, g_{\text{BSM},1}) \\ & + \underbrace{(a_{21}g_{\text{SM}}^2 + a_{22}g_{\text{BSM}}^2 + a_{23}g_{\text{SM}}g_{\text{BSM}})}_{w_2} T_{\text{in}}(g_{\text{SM},2}, g_{\text{BSM},2}) \\ & + \underbrace{(a_{31}g_{\text{SM}}^2 + a_{32}g_{\text{BSM}}^2 + a_{33}g_{\text{SM}}g_{\text{BSM}})}_{w_3} T_{\text{in}}(g_{\text{SM},3}, g_{\text{BSM},3}). \end{aligned}$$

The mapping of templates to operators

- Generalizing further ... at



This then only leaves the calculation of the appropriate coefficients a_{ij} that occur in the weights functions w_1, w_2, w_3 for templates T_1, T_2, T_3

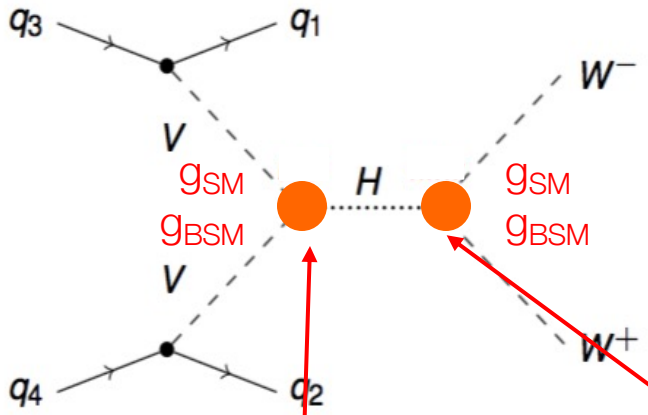
Their solution is found by solving the matrix equation

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} g^2_{\mathrm{SM},1} & g^2_{\mathrm{SM},2} & g^2_{\mathrm{SM},3} \\ g^2_{\mathrm{BSM},1} & g^2_{\mathrm{BSM},2} & g^2_{\mathrm{BSM},3} \\ g_{\mathrm{SM},1} g_{\mathrm{BSM},1} & g_{\mathrm{SM},2} g_{\mathrm{BSM},2} & g_{\mathrm{SM},3} g_{\mathrm{BSM},3} \end{pmatrix} = \mathbb{1}$$

$$\begin{aligned} T_{\mathrm{out}}(g_{\mathrm{SM}}, g_{\mathrm{BSM}}) = & \underbrace{(a_{11} g^2_{\mathrm{SM}} + a_{12} g^2_{\mathrm{BSM}} + a_{13} g_{\mathrm{SM}} g_{\mathrm{BSM}})}_{w_1} T_{\mathrm{in}}(g_{\mathrm{SM},1}, g_{\mathrm{BSM},1}) \\ & + \underbrace{(a_{21} g^2_{\mathrm{SM}} + a_{22} g^2_{\mathrm{BSM}} + a_{23} g_{\mathrm{SM}} g_{\mathrm{BSM}})}_{w_2} T_{\mathrm{in}}(g_{\mathrm{SM},2}, g_{\mathrm{BSM},2}) \\ & + \underbrace{(a_{31} g^2_{\mathrm{SM}} + a_{32} g^2_{\mathrm{BSM}} + a_{33} g_{\mathrm{SM}} g_{\mathrm{BSM}})}_{w_3} T_{\mathrm{in}}(g_{\mathrm{SM},3}, g_{\mathrm{BSM},3}). \end{aligned}$$

A more realistic physics example

- In many scenarios new physics can enter amplitudes in both the production and decay vertex of a t-channel process



- Corresponding matrix element of this process

$$\mathcal{M}(g_{\mathrm{SM}}, g_{\mathrm{BSM}}) = \left(g_{\mathrm{SM}} \cdot O_{\mathrm{SM},p} + g_{\mathrm{BSM}} \cdot O_{\mathrm{BSM},p}\right) \cdot \left(g_{\mathrm{SM}} \cdot O_{\mathrm{SM},d} + g_{\mathrm{BSM}} \cdot O_{\mathrm{BSM},d}\right).$$

A more realistic physics example

- A little math shows we now need 5 independent templates

$$\mathcal{M}(g_{\rm SM}, g_{\rm BSM}) = \left(g_{\rm SM}\cdot O_{{\rm SM},p} + g_{\rm BSM}\cdot O_{{\rm BSM},p}\right)\cdot\left(g_{\rm SM}\cdot O_{{\rm SM},d} + g_{\rm BSM}\cdot O_{{\rm BSM},d}\right).$$

$$\begin{aligned}
|\mathcal{M}(g_{\rm SM}, g_{\rm BSM})|^2 &= \left(g_{\rm SM}O_{{\rm SM},p} + g_{\rm BSM}O_{{\rm BSM},p}\right)^2\cdot\left(g_{\rm SM}O_{{\rm SM},d} + g_{\rm BSM}O_{{\rm BSM},d}\right)^2 && \leftarrow 1\\
&= g_{\rm SM}^4\cdot O_{{\rm SM},p}^2 O_{{\rm SM},d}^2 + g_{\rm BSM}^4\cdot O_{{\rm BSM},p}^2 O_{{\rm BSM},d}^2 && \leftarrow 2\\
&+ g_{\rm SM}^3 g_{\rm BSM}\cdot\left(O_{{\rm SM},p}^2\,\Re(O_{{\rm SM},d}^* O_{{\rm BSM},d}) + \Re(O_{{\rm SM},p}^* O_{{\rm BSM},p})O_{{\rm SM},d}^2\right) && \leftarrow 3\\
&+ g_{\rm SM}^2 g_{\rm BSM}^2\cdot\left(O_{{\rm SM},p}^2 O_{{\rm BSM},d}^2 + O_{{\rm BSM},p}^2 O_{{\rm SM},d}^2\right) && \leftarrow 4\\
&+ g_{\rm SM} g_{\rm BSM}^3\cdot\left(O_{{\rm BSM},p}^2\,\Re(O_{{\rm SM},d}^* O_{{\rm BSM},d}) + \Re(O_{{\rm SM},p}^* O_{{\rm BSM},p})O_{{\rm BSM},d}^2\right). && \leftarrow 5
\end{aligned}$$

- And the template model can be written as

$$\begin{aligned}
T_{\rm out}(g_{\rm SM}, g_{\rm BSM}) &= \underbrace{\left(a_{11}g_{\rm SM}^4 + a_{12}g_{\rm SM}^3 g_{\rm BSM} + a_{13}g_{\rm SM}^2 g_{\rm BSM}^2 + a_{14}g_{\rm SM}g_{\rm BSM}^3 + a_{15}g_{\rm BSM}^4\right)}_{w_1} T_{\rm in}(g_{{\rm SM},1}, g_{{\rm BSM},1})\\
&= \underbrace{\left(a_{21}g_{\rm SM}^4 + a_{22}g_{\rm SM}^3 g_{\rm BSM} + a_{23}g_{\rm SM}^2 g_{\rm BSM}^2 + a_{24}g_{\rm SM}g_{\rm BSM}^3 + a_{25}g_{\rm BSM}^4\right)}_{w_2} T_{\rm in}(g_{{\rm SM},2}, g_{{\rm BSM},2})\\
&= \underbrace{\left(a_{31}g_{\rm SM}^4 + a_{32}g_{\rm SM}^3 g_{\rm BSM} + a_{33}g_{\rm SM}^2 g_{\rm BSM}^2 + a_{34}g_{\rm SM}g_{\rm BSM}^3 + a_{35}g_{\rm BSM}^4\right)}_{w_3} T_{\rm in}(g_{{\rm SM},3}, g_{{\rm BSM},3})\\
&= \underbrace{\left(a_{41}g_{\rm SM}^4 + a_{42}g_{\rm SM}^3 g_{\rm BSM} + a_{43}g_{\rm SM}^2 g_{\rm BSM}^2 + a_{44}g_{\rm SM}g_{\rm BSM}^3 + a_{45}g_{\rm BSM}^4\right)}_{w_4} T_{\rm in}(g_{{\rm SM},4}, g_{{\rm BSM},4})\\
&= \underbrace{\left(a_{51}g_{\rm SM}^4 + a_{52}g_{\rm SM}^3 g_{\rm BSM} + a_{53}g_{\rm SM}^2 g_{\rm BSM}^2 + a_{54}g_{\rm SM}g_{\rm BSM}^3 + a_{55}g_{\rm BSM}^4\right)}_{w_5} T_{\rm in}(g_{{\rm SM},5}, g_{{\rm BSM},5}).
\end{aligned}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15}\\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25}\\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35}\\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45}\\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} \cdot \begin{pmatrix} g_{{\rm SM},1}^4 & g_{{\rm SM},2}^4 & g_{{\rm SM},3}^4 & g_{{\rm SM},4}^4 & g_{{\rm SM},5}^4\\ g_{{\rm SM},1}^3 g_{{\rm BSM},1} & g_{{\rm SM},2}^3 g_{{\rm BSM},2} & g_{{\rm SM},3}^3 g_{{\rm BSM},3} & g_{{\rm SM},4}^3 g_{{\rm BSM},4} & g_{{\rm SM},5}^3 g_{{\rm BSM},5}\\ g_{{\rm SM},1}^2 g_{{\rm BSM},1}^2 & g_{{\rm SM},2}^2 g_{{\rm BSM},2}^2 & g_{{\rm SM},3}^2 g_{{\rm BSM},3}^2 & g_{{\rm SM},4}^2 g_{{\rm BSM},4}^2 & g_{{\rm SM},5}^2 g_{{\rm BSM},5}^2\\ g_{{\rm SM},1} g_{{\rm BSM},1}^3 & g_{{\rm SM},2} g_{{\rm BSM},2}^3 & g_{{\rm SM},3} g_{{\rm BSM},3}^3 & g_{{\rm SM},4} g_{{\rm BSM},4}^3 & g_{{\rm SM},5} g_{{\rm BSM},5}^3\\ g_{{\rm BSM},1}^4 & g_{{\rm BSM},2}^4 & g_{{\rm BSM},3}^4 & g_{{\rm BSM},4}^4 & g_{{\rm BSM},5}^4 \end{pmatrix} = \mathbb{1}$$

Generalizing to processes with N amplitudes contributing

- General form of matrix element for process p → X → Y is

$$|\mathcal{M}(\vec{g})|^2 = \underbrace{\left(\sum_{x\in p,b} g_x O(g_x)\right)^2}_{\text{production}} \cdot \underbrace{\left(\sum_{x\in d,b} g_x O(g_x)\right)^2}_{\text{decay}}.$$

- Number of independent terms N for general form

n_p amplitudes excl. in production
n_s amplitudes in both
n_d amplitudes excl. in decay

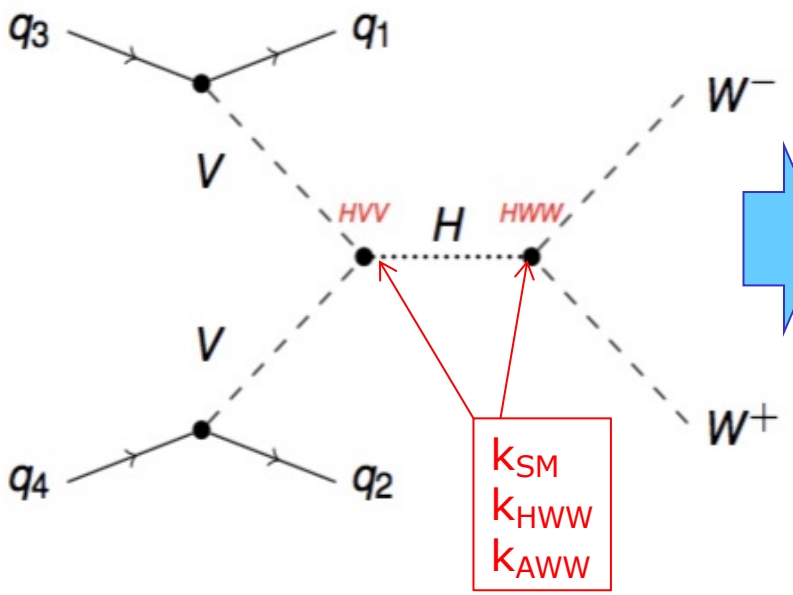
$$N = \frac{n_p\left(n_p+1\right)}{2}\cdot\frac{n_d\left(n_d+1\right)}{2} + \binom{4+n_s-1}{4} + \left(n_p\cdot n_s + \frac{n_s\left(n_s+1\right)}{2}\right)\cdot\frac{n_d\left(n_d+1\right)}{2}$$
$$+\left(n_d\cdot n_s + \frac{n_s\left(n_s+1\right)}{2}\right)\cdot\frac{n_p\left(n_p+1\right)}{2} + \frac{n_s\left(n_s+1\right)}{2}\cdot n_p\cdot n_d + \left(n_p+n_d\right)\binom{3+n_s-1}{3}.$$

- Numeric answers for some Higgs processes

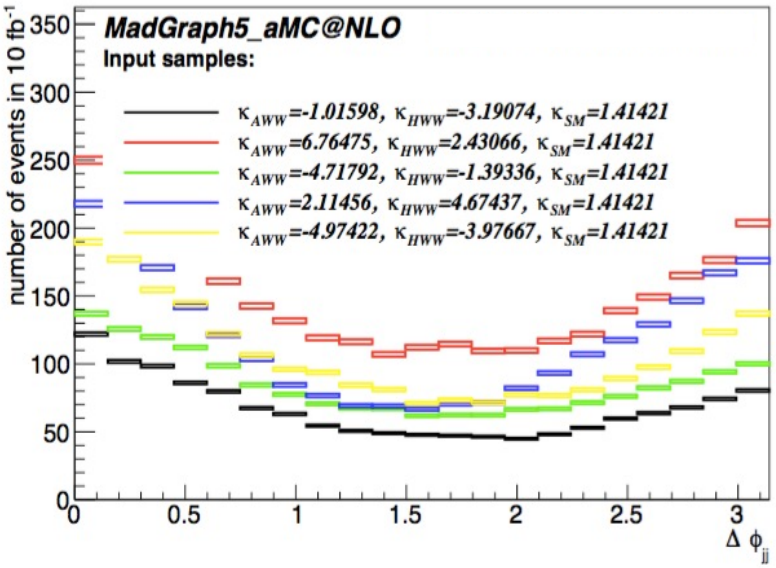
Process	n_p	n_d	n_s	N
ggF $H \to ZZ^* \to 4\ell$ truth	1	2	0	3
VBF $H \to WW^* \to e\nu\mu\nu$ truth	0	0	3	15
ggF $H \to ZZ^* \to 4\ell$ reconstructed	1	3	0	6
VBF $H \to \mu\mu$ truth	13	1	0	91

A concrete example VBH → H → WW

3 shared parameters → 15 terms in $|M|^2$ expression → 15 input distributions needed

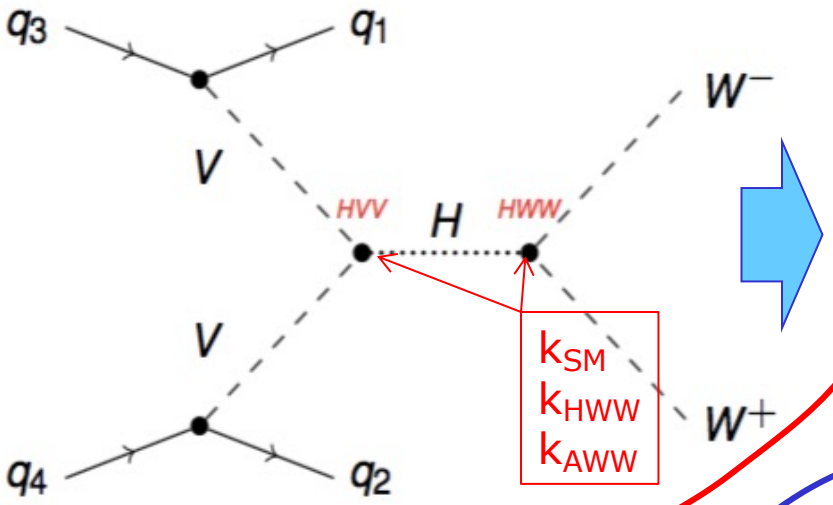


$$T_{out}(\Delta\phi_{jj}|k_{SM},k_{HWW},k_{AWW}) = \Sigma\, w_i(k_{SM},k_{HWW},k_{AWW}) * T_{in,i}(\Delta\phi_{jj})$$

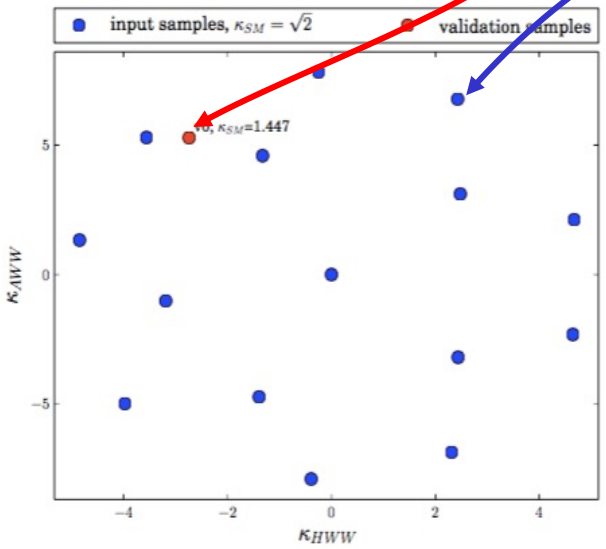


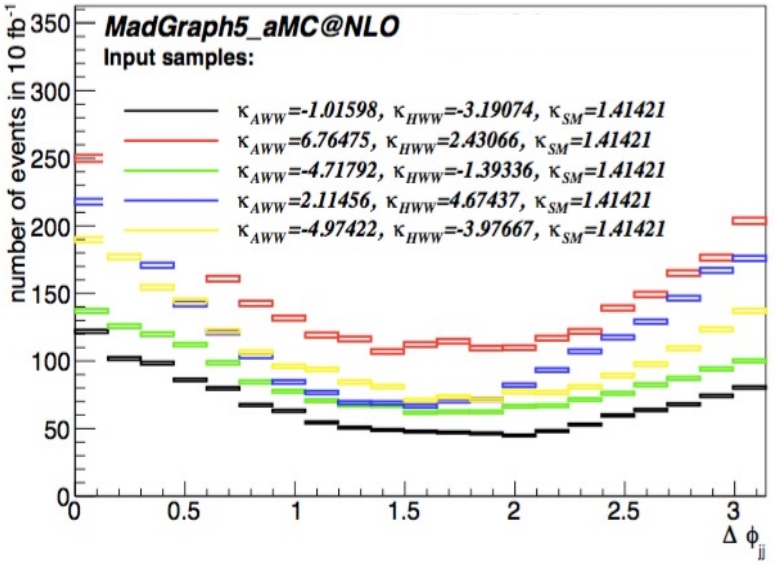
A concrete example VBH → H → WW

3 shared parameters → 15 terms in $|M|^2$ expression → 15 input distributions needed



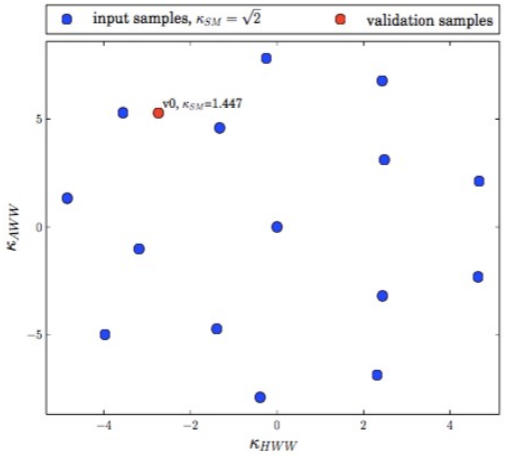
$$T_{out}(\Delta\phi_{jj}|k_{SM},k_{HWW},k_{AWW}) = \Sigma\, w_i(k_{SM},k_{HWW},k_{AWW}) * T_{in,i}(\Delta\phi_{jj})$$





Summary on amplitude models

- Amplitude sum models work (in terms of mathematics) exactly the same as template interpolation models.
  - But have difference choice of coefficient (polynomials instead of linear terms)
  - Appropriate choice results in interpolation mechanism that is physically meaningful → no approximation in morphing (beyond assumption of LO physics)
  - Freedom of choice in sampling points ensures that all sampled distributions are positive definite (no interference-only terms)
  - Computationally fast & efficient process
- But need to watch configuration of sampled points
  - → if interpolated states (e.g. measured minimum) is far from important samples then large scale factors might be applied
  - → If so blow-up of MC statistical fluctuations
- Quite new approach – first physics results with amplitude morphing now published



Roadmap of this course

- Start with basics, gradually build up to complexity