ML and QCD global analysis

Nobuo Sato

AI4NP Winter School
JAN 2021

Jefferson Lab

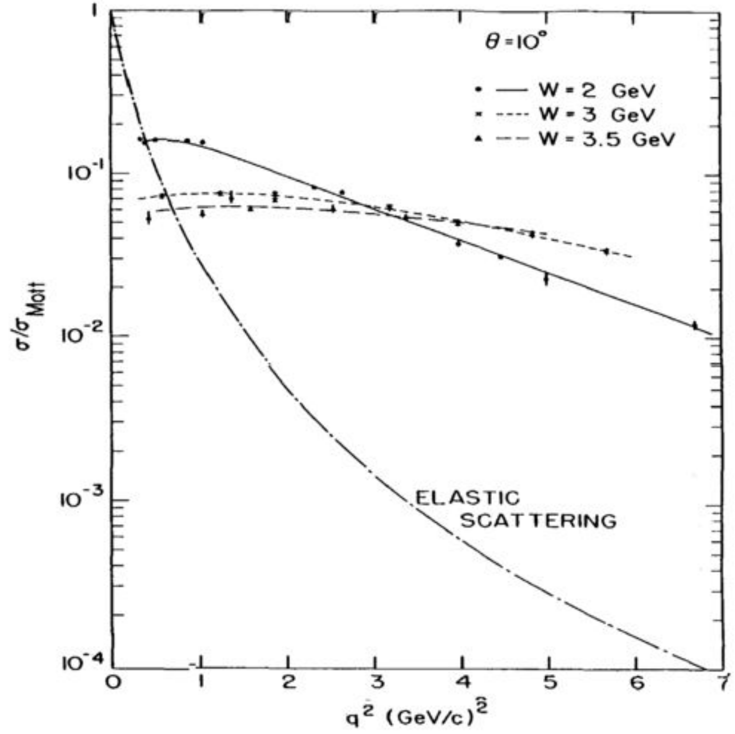
Outline

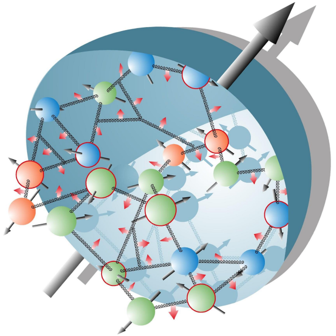
Lecture 1
- Motivations
- QCD carpentry setup
- Solving QCD's beta function

Lecture 2
- Mellin transforms
- Solving DGLAP
- Modeling input scale PDFs

Lecture 3
- DIS theory
- World DIS data
- The chi2 function
- Global analysis

Lecture 4
- Bayesian inference
  - Maximum likelihood
  - MC methods
- JAM history
- Machine learning

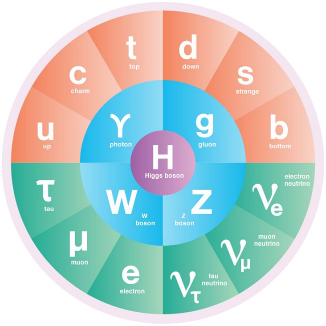




Discovery of point-like particles inside proton

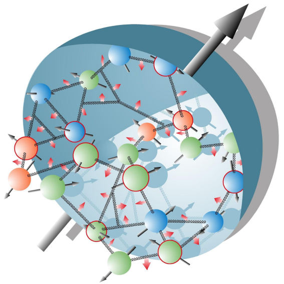
Motivations

Understanding the emergent phenomena of QCD

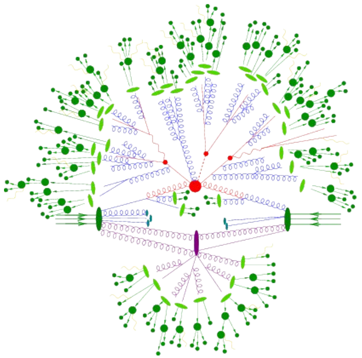


parts

“*In philosophy, systems theory, science, and art, emergence occurs when an* ***entity is observed*** *to have properties* ***its parts*** *do not have on their own, properties or behaviors which emerge only when the parts interact in a wider whole.*” Wiki



Hadron Structure



Hadron formation

Observed entity

What do we mean by "hadron structure" ? (1D)

$$\xi = \frac{k^+}{P^+}$$

Parton momentum fraction relative to parent hadron

$$f_i(\xi) = \int \frac{\mathrm{d}w^-}{4\pi} e^{-i\xi p^+ w^-} \boxed{\langle N|\bar{\psi}_i(0, w^-, \mathbf{0}_\mathrm{T})\gamma^+\psi_i(0)|N\rangle}$$

parton distribution function (PDF)

Interpretation in non-interacting QCD

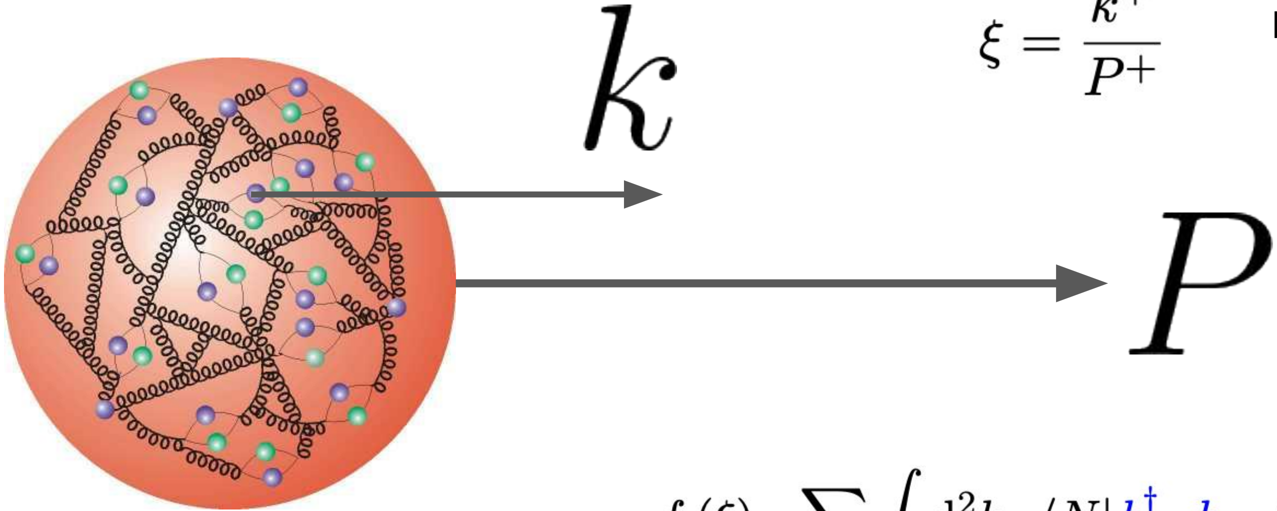
$$\psi_i(x) = \sum_{k,\alpha} b_{k,\alpha}(x^+) u_{k,\alpha} e^{-ik^+x^- + ik_\mathrm{T}\cdot x_\mathrm{T}} + d^\dagger_{k,\alpha}(x^+) u_{k,-\alpha} e^{ik^+x^- - ik_\mathrm{T}\cdot x_\mathrm{T}}$$

$$f_i(\xi) \sim \sum_\alpha \int \mathrm{d}^2k_\mathrm{T}\, \langle N|\underbrace{b^\dagger_{k,\alpha} b_{k,\alpha}(\xi p^+, k_\mathrm{T}, \alpha)}_{\text{number operator}}|N\rangle$$

How quarks and gluons are distributed?

$$\xi = \frac{k^+}{P^+}$$

Momentum fraction



$$f_i(\xi) \sim \sum_{\alpha} \int \mathrm{d}^2 k_{\mathrm{T}} \, \langle N | \underbrace{b^{\dagger}_{k,\alpha} b_{k,\alpha} (\xi p^+, k_{\mathrm{T}}, \alpha)}_{\text{number operator}} | N \rangle$$

What do we mean by "hadronization" ? (1D)

$$\zeta = \frac{p_h^+}{k^+}$$

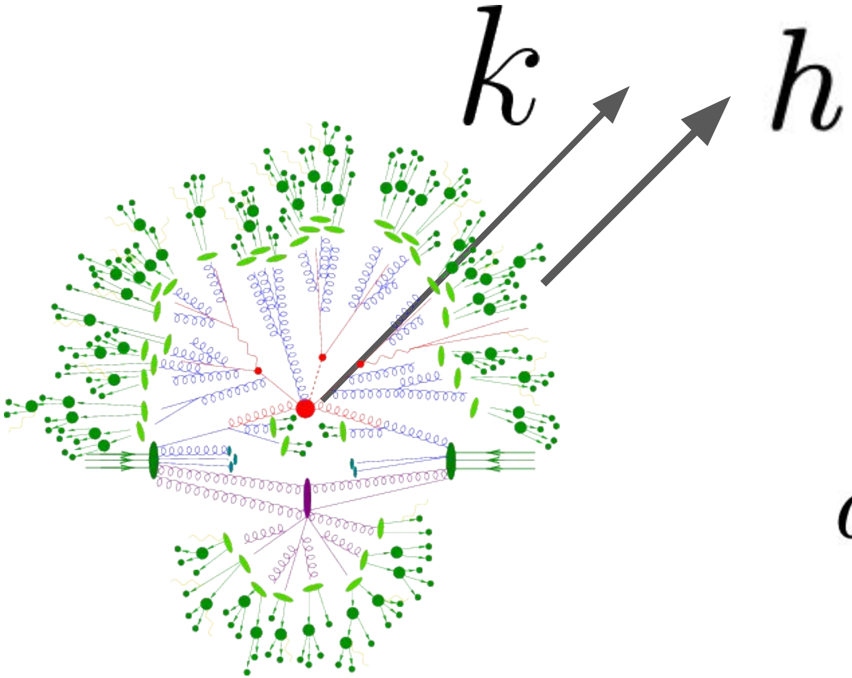
hadron momentum fraction relative to parent parton

$$d_{h/j}(\zeta) = \frac{\mathrm{Tr}_{\text{color,Dirac}}}{4N_{c,j}} \sum_X \zeta \int \frac{dw^+}{2\pi} e^{i(p_h^-/\zeta)w^+}$$
$$\times \gamma^- \langle 0 | \bar{\psi}_j(0, w^+, \mathbf{0}_\mathrm{T}) \, |p_h, X\rangle\langle p_h, X| \, \psi_j(0) | 0\rangle$$

Fragmentation functions (FFs)

X = all states except detected hadron ***h***

How quarks and gluons are distributed?



$$\zeta = \frac{p_h^+}{k^+}$$

$$d_{h/j}(\zeta)$$

Number density of hadrons from parent parton

Hadron structure in interacting theory

Definition of PDFs in field theory requires renormalization

PDFs will depend on renormalization scale and its RGEs are the famous DGLAP equations

UV singularity when the field separation is zero

$$f_i(\xi) \stackrel{!}{=} \int \frac{\mathrm{d}w^-}{4\pi} e^{-i\xi p^+ w^-} \langle N|\bar{\psi}_i(0, w^-, \mathbf{0}_\mathrm{T})\gamma^+\psi_i(0)|N\rangle$$

Renormalization

$$f = Z_F \otimes f_{\mathrm{bare}}$$

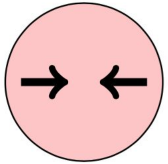
$$f(\xi) \to f(\xi, \mu)$$

Dokshitzer–**G**ribov–**L**ipatov–**A**ltarelli–**P**arisi

$$\frac{\mathrm{d} f_i(\xi,\mu^2)}{\mathrm{d}\ln\mu^2} = \sum_j \int_\xi^1 \frac{\mathrm{d}y}{y} P_{ij}(\xi,\mu^2)\, f_j\left(\frac{y}{\xi},\mu^2\right)$$

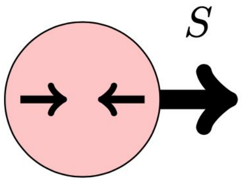
aka **DGLAP**

Spin structures



$$f = f_{\rightarrow} + f_{\leftarrow}$$

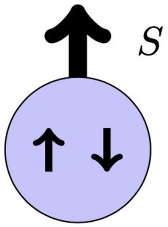
$$\langle N|\bar{\psi}_i(0, w^-, \mathbf{0}_\mathrm{T})\gamma^+\psi_i(0)|N\rangle$$



$$\Delta f = f_{\rightarrow} - f_{\leftarrow}$$

Helicity distribution

$$\langle N|\bar{\psi}_i(0, w^-, \mathbf{0}_\mathrm{T})\gamma^+\gamma_5\psi_i(0)|N\rangle$$



$$\delta_\mathrm{T} f = f_{\uparrow} - f_{\downarrow}$$

Transversity

$$\langle N|\bar{\psi}_i(0, w^-, \mathbf{0}_\mathrm{T})\gamma^+\gamma_\perp\gamma_5\psi_i(0)|N\rangle$$

Extensions to 3D

$$\xi = \frac{k^+}{P^+}$$

Longitudinal momentum fraction

$$f(\xi)$$

PDFs

$$f(\xi, k_{\mathrm{T}})$$

Transverse momentum distribution -> **TMDs**

$$f(\xi, b_{\mathrm{T}})$$

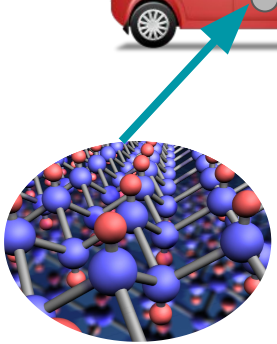
Impact parameter distribution -> **GPDs**

So how do we get hadron structure from experimental data?

What part of this is the "internal structure"?

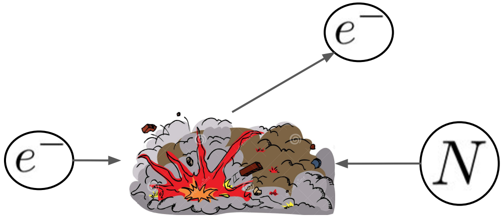
Factorization

Want to see
internal structure

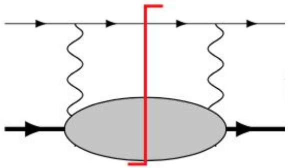


But we only see debris

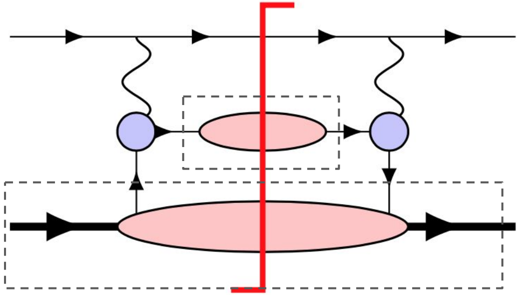
Factorization in deep-inelastic scattering (DIS)

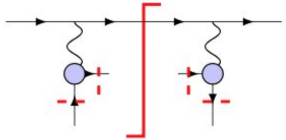


$$E' \frac{\mathrm{d}\sigma}{\mathrm{d}^3 l'}$$

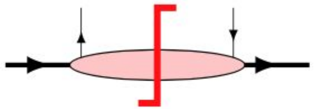


Approximations





Collision dependent factor



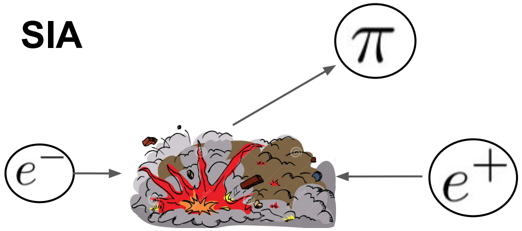
Internal structure

$$\sum_i \int_x^1 \frac{\mathrm{d}\xi}{\xi} H_i(\xi)\, f_i\left(\frac{x}{\xi}\right) + \mathcal{O}\left(\frac{m^2}{Q^2}\right)$$

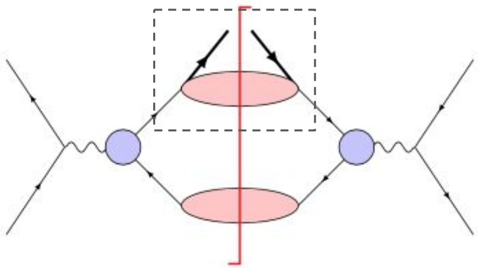
Error of approximations

Factorization in other reactions

SIA

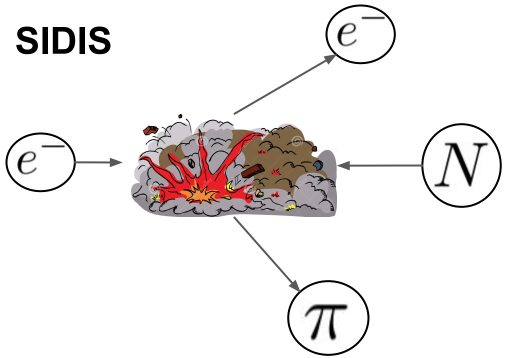


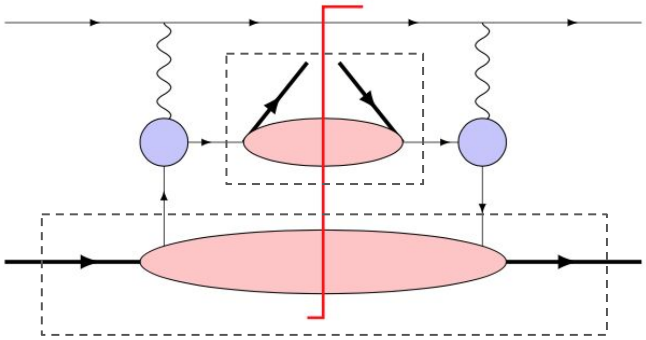
hadronization



$$d\sigma = \sum_i H_i^{\mathrm{SIA}} \otimes d_i$$

SIDIS



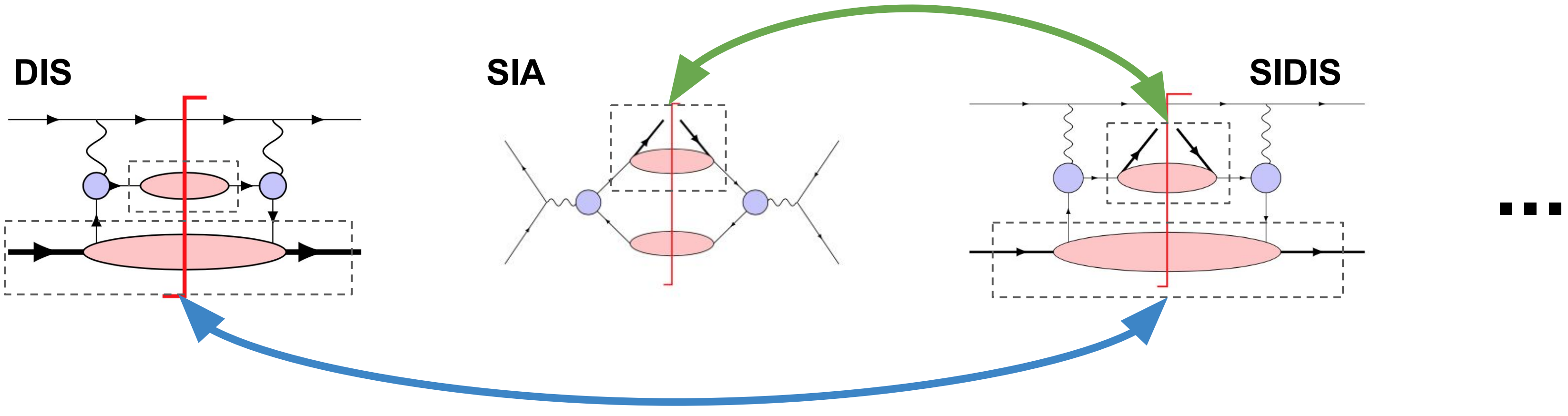


$$d\sigma = \sum_{ij} H_{ij}^{\mathrm{SIDIS}} \otimes f_i \otimes d_j$$

structure + hadronization

..and many more

Universality



cross sections described by **universal non-perturbative** functions, e.g. PDFs, FFs

The Bayesian inference

Experiments = theory + errors

$$d\sigma_{\mathrm{DIS}} = \sum_i H_i^{\mathrm{DIS}} \otimes f_i$$

$$d\sigma_{\mathrm{DY}} = \sum_{ij} H_{ij}^{\mathrm{DY}} \otimes f_i \otimes f_j$$

$$d\sigma_{\mathrm{SIA}} = \sum_i H_i^{\mathrm{SIA}} \otimes d_i$$

$$d\sigma_{\mathrm{SIDIS}} = \sum_{ij} H_{ij}^{\mathrm{SIDIS}} \otimes f_i \otimes d_j$$

Hadron Structure

Hadronization

RGE boundary conditions

$$f_i(\xi, \mu_0^2) = N_i \xi^{a_i}(1-\xi)^{b_i}(1+\ldots)$$

$$d_i(\zeta, \mu_0^2) = N_i \zeta^{a_i}(1-\zeta)^{b_i}(1+\ldots)$$

$$\mathbf{a} = (N_i, a_i, b_i, \ldots)$$

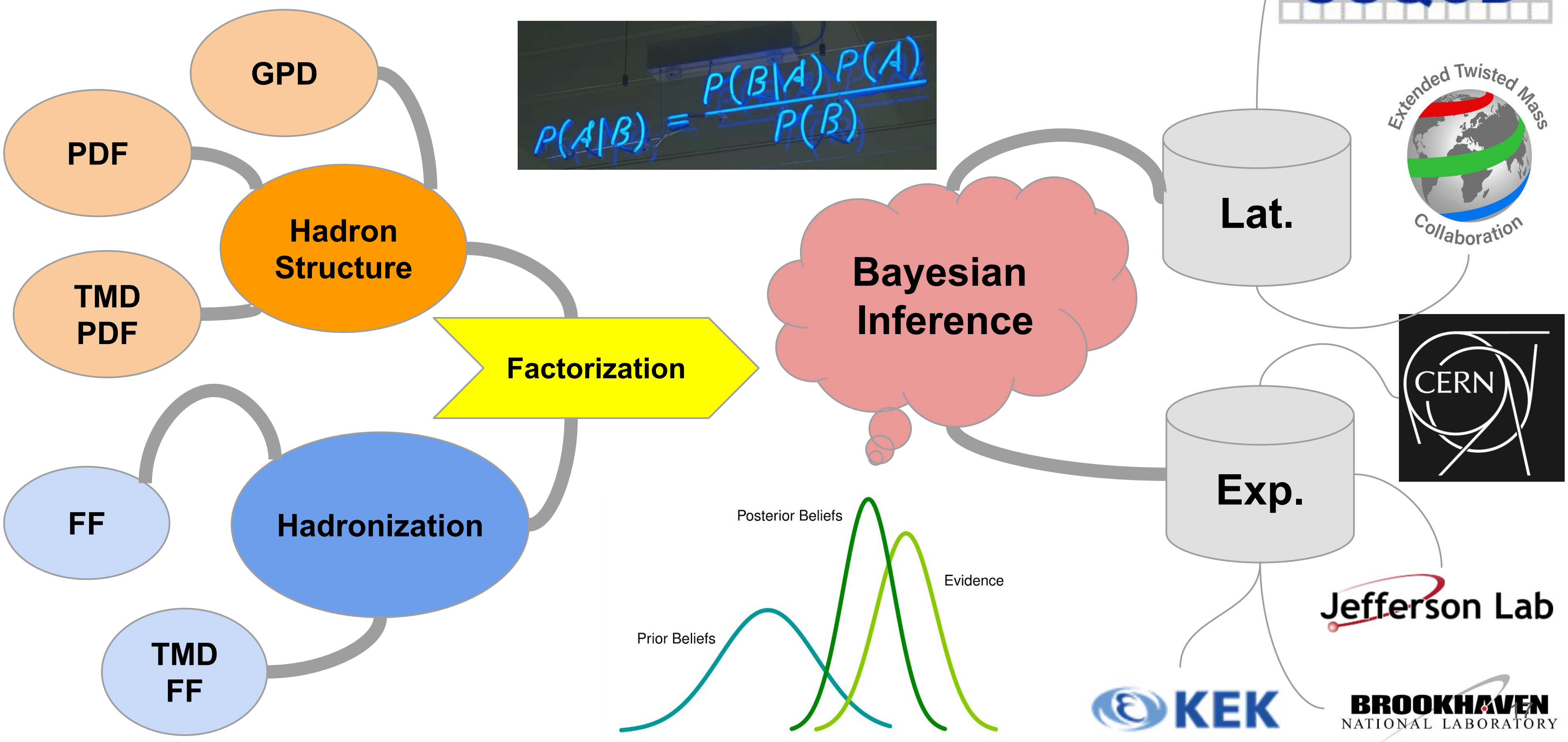
$$\underbrace{\rho(\mathbf{a}|\mathrm{data})}_{\text{Posterior distribution}} \sim \underbrace{\mathcal{L}(\mathbf{a}, \mathrm{data})}_{\text{Likelihood}}\,\underbrace{\pi(\mathbf{a})}_{\text{Prior distribution}}$$

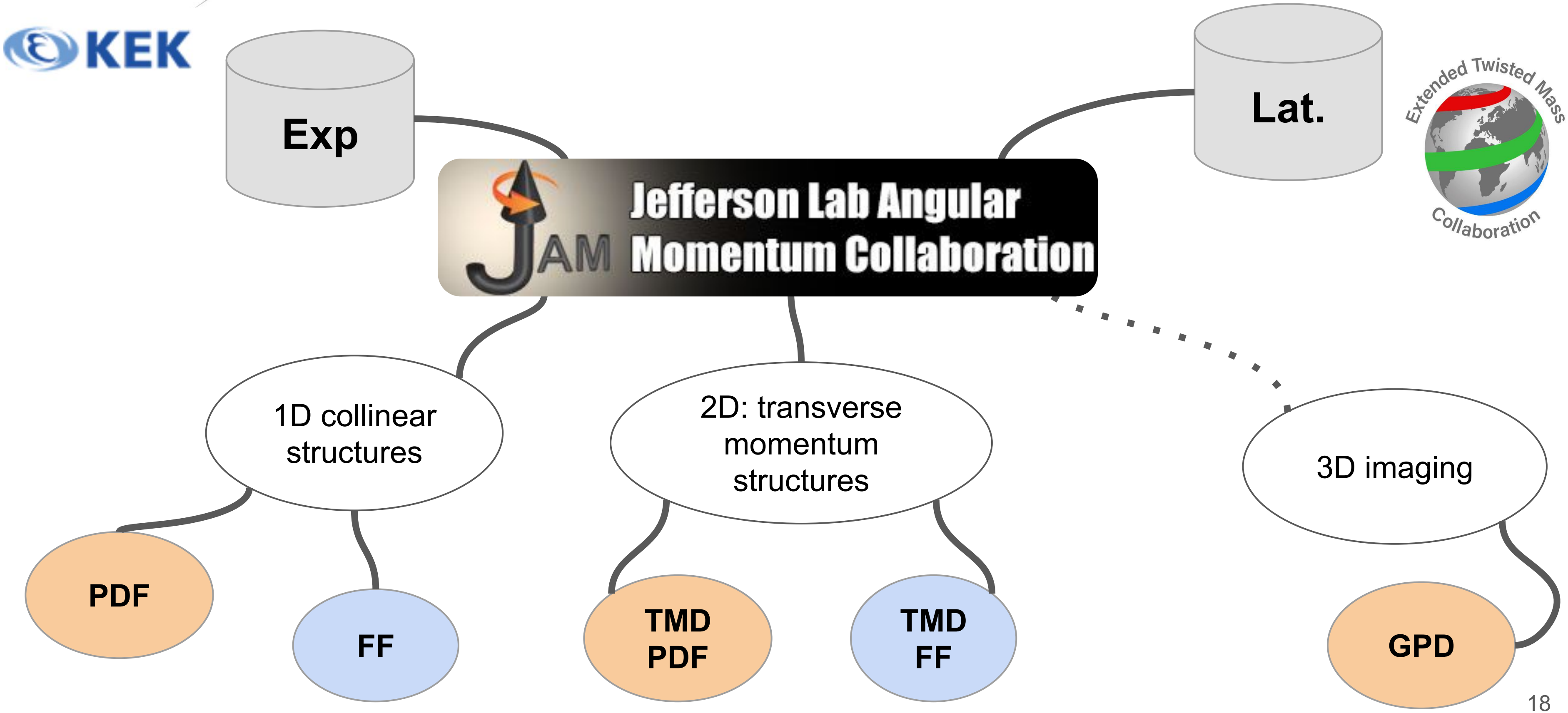
$$\mathcal{L}(\boldsymbol{a}, \mathrm{data}) = \exp\left[-\frac{1}{2}\chi^2(\boldsymbol{a}, \mathrm{data})\right]$$

$$\mathrm{E}[f_i(\xi, \mu^2)] = \int d^n\mathbf{a}\; \rho(\mathbf{a}|\mathrm{data}) f_i(\xi, \mu^2; \mathbf{a})$$

$$\mathrm{V}[f_i(\xi, \mu^2)] = \int d^n\mathbf{a}\; \rho(\mathbf{a}|\mathrm{data}) \left[f_i(\xi, \mu^2; \mathbf{a}) - \mathrm{E}[f_i(\xi, \mu^2)]\right]^2$$

The QCD global analysis paradigm

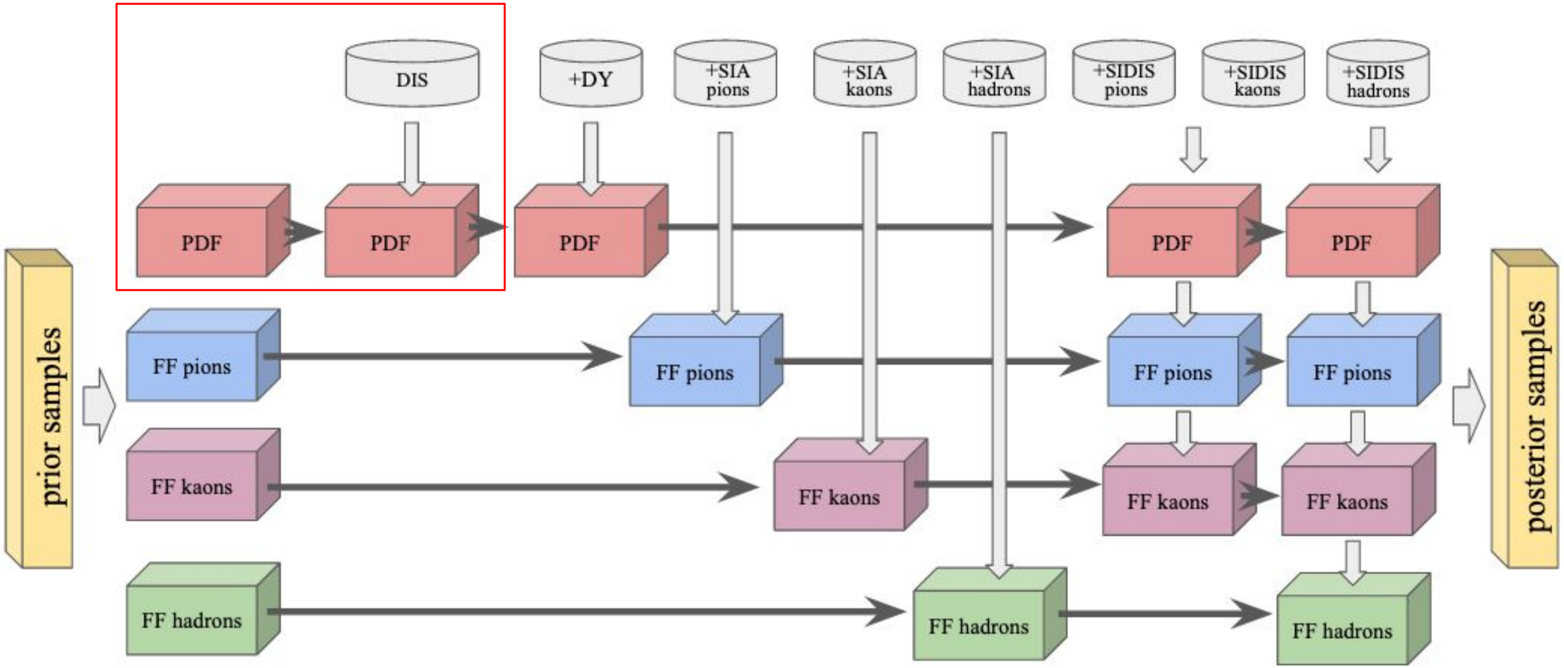



Exp
Lat.
Jefferson Lab Angular Momentum Collaboration
1D collinear structures
2D: transverse momentum structures
3D imaging
PDF
FF
TMD PDF
TMD FF
GPD

An example: JAM20-SIDIS

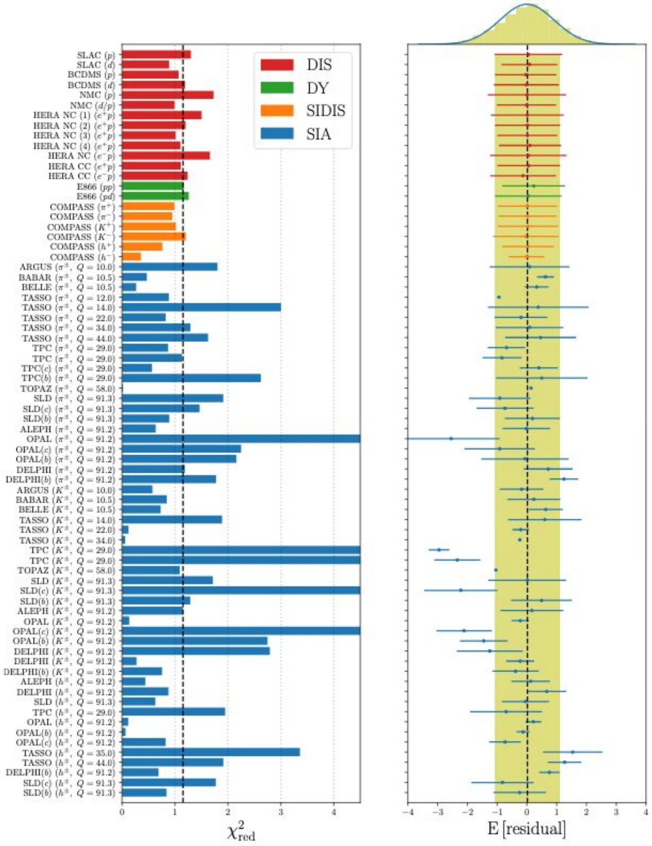
Moffat, Melnitchouk, Rogers, NS
[arXiv:2101.04664](https://arxiv.org/abs/2101.04664)

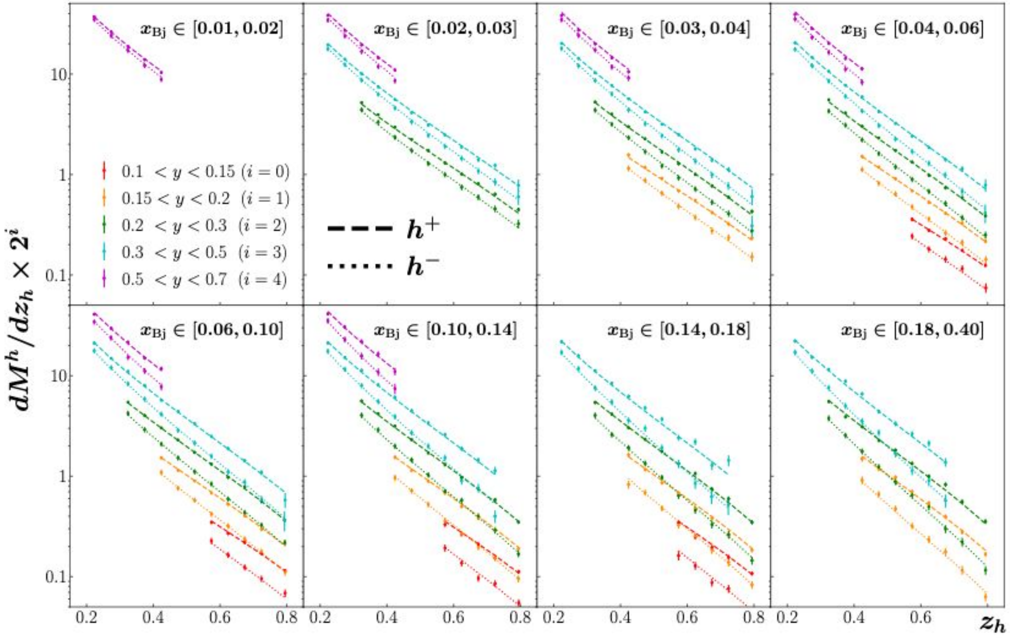
These lectures



An example: JAM20-SIDIS

Moffat, Melnitchouk, Rogers, NS
[arXiv:2101.04664](https://arxiv.org/abs/2101.04664)

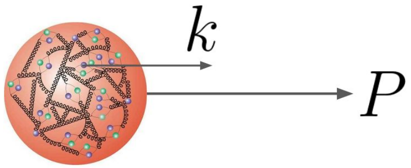


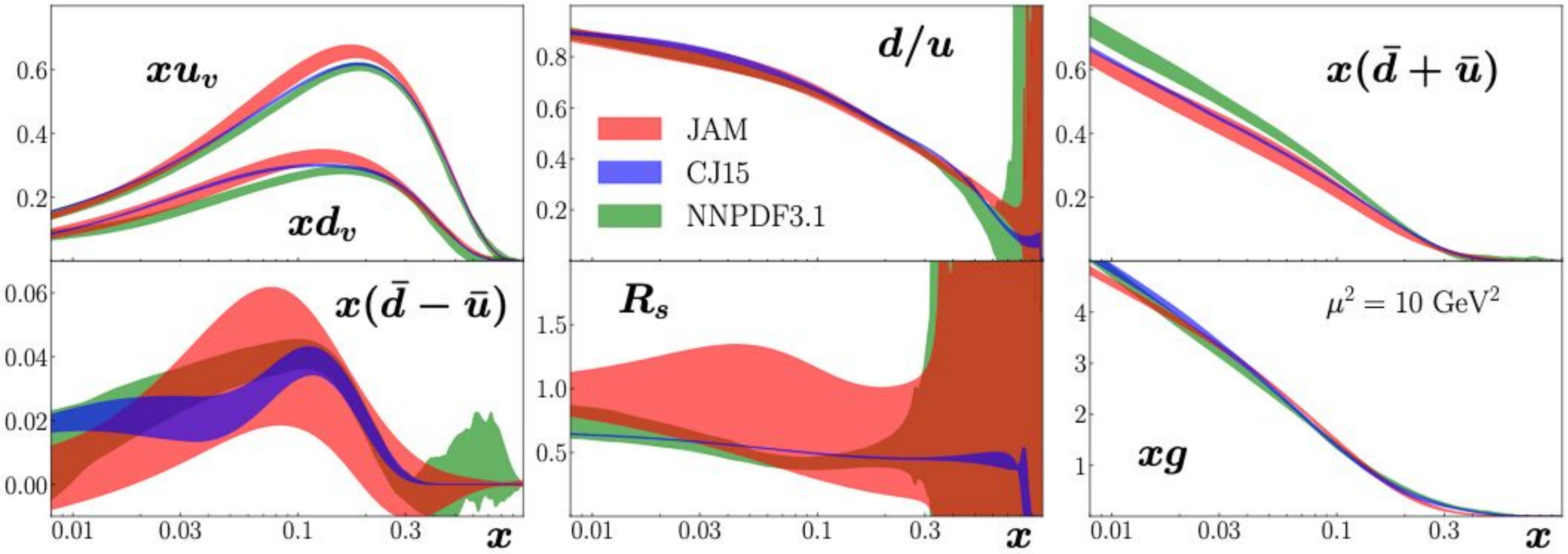


An example: JAM20-SIDIS

Moffat, Melnitchouk, Rogers, NS
arXiv:2101.04664

PDFs

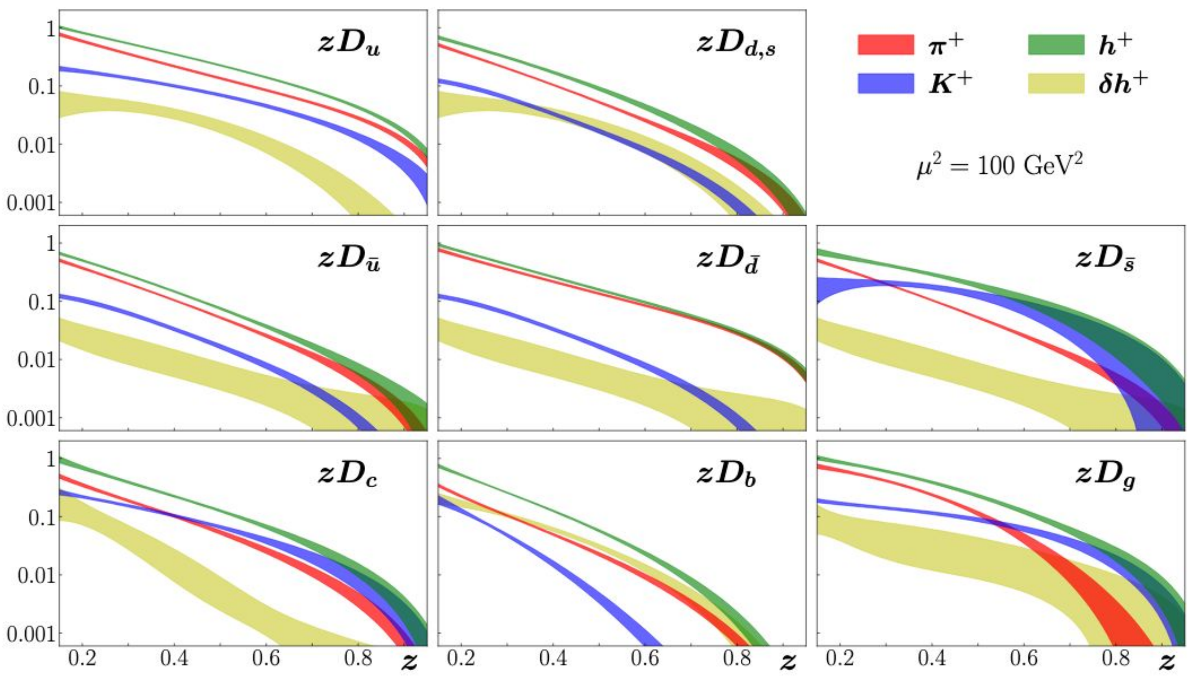


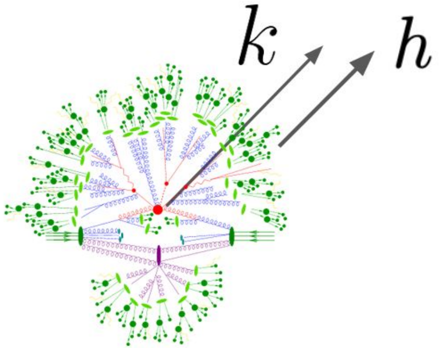


An example: JAM20-SIDIS

Moffat, Melnitchouk, Rogers, NS
arXiv:2101.04664

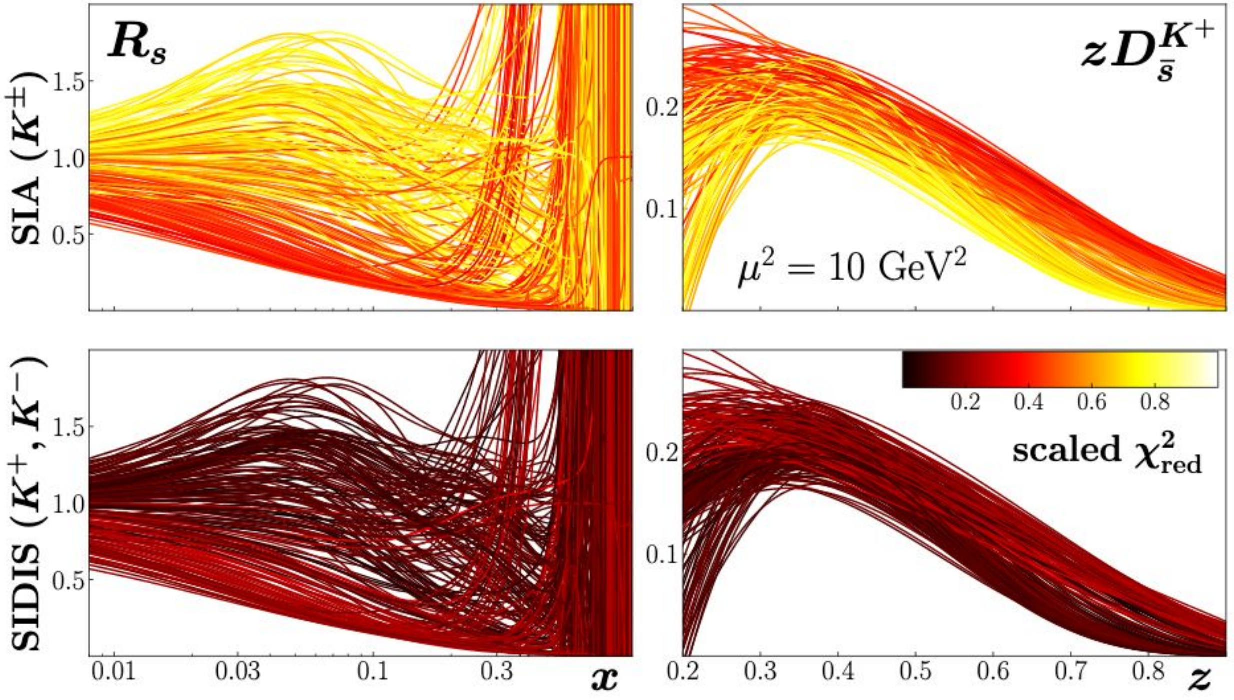
FFs





An example: JAM20-SIDIS

Moffat, Melnitchouk, Rogers, NS
arXiv:2101.04664

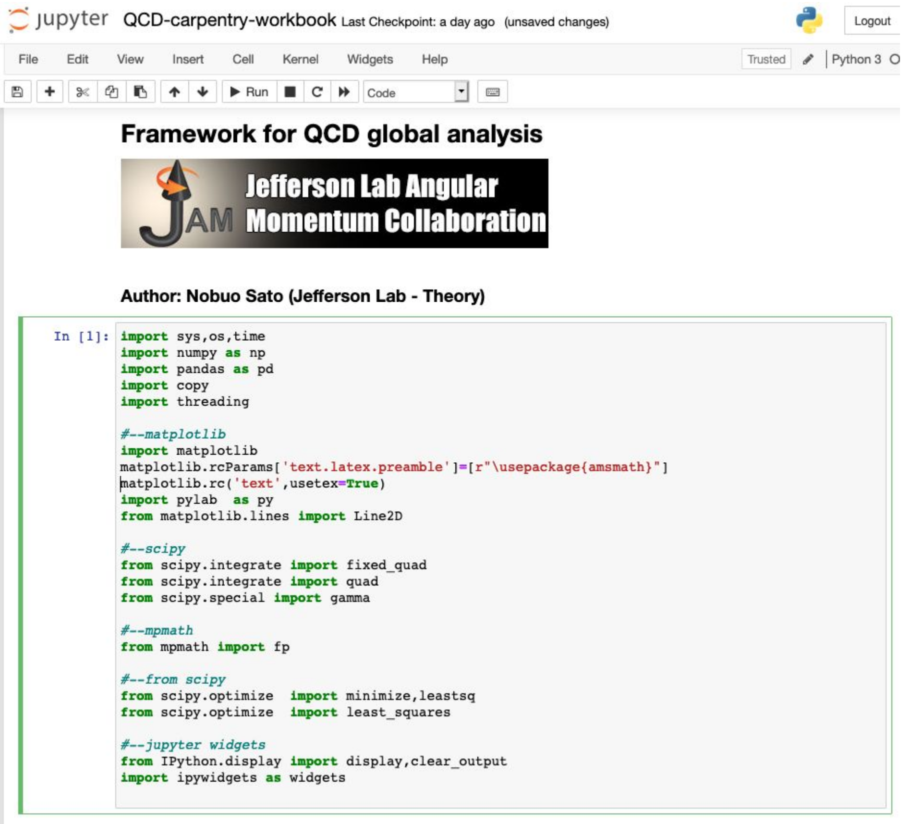


$$R_s = \frac{s + \bar{s}}{\bar{u} + \bar{d}}$$

The simultaneous fit of PDFs and FFs provides new insights on nucleon strangeness

QCD carpentry in python

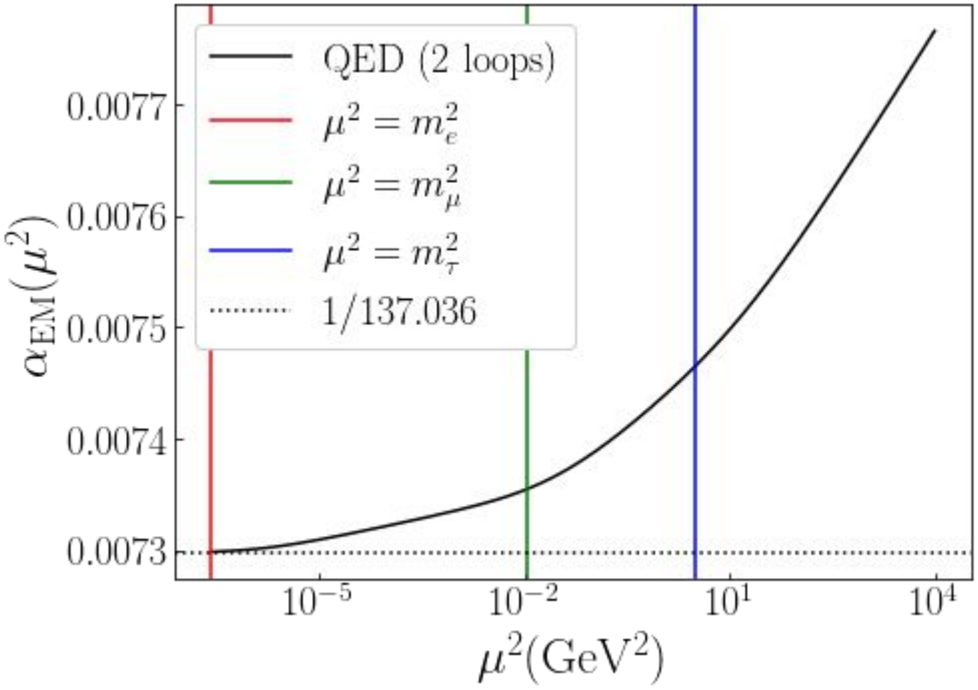
- We will use a jupyter-notebook available at https://github.com/QCDHUB/qcdcarpentry
- The lectures involves several exercises. I will give few minutes to work on them
- You need to have jupyter notebook available in your computer. All the required dependencies are listed. Use $pip install xyz to get libraries you don't have.
- Ok, let's take a look the notebook

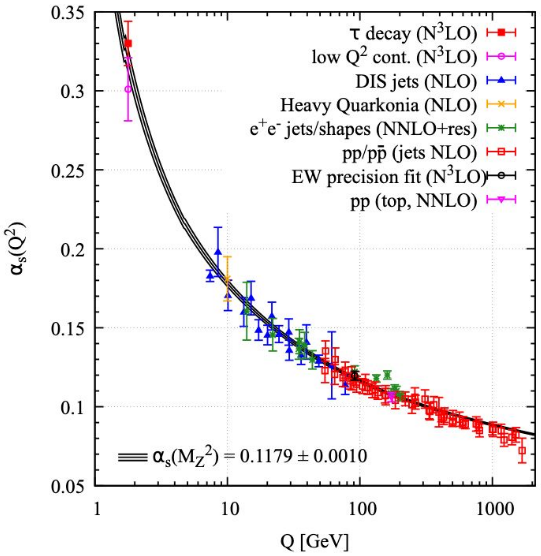


Continue to Jupyter notebook

Exercise 1 (time: 5 mins)

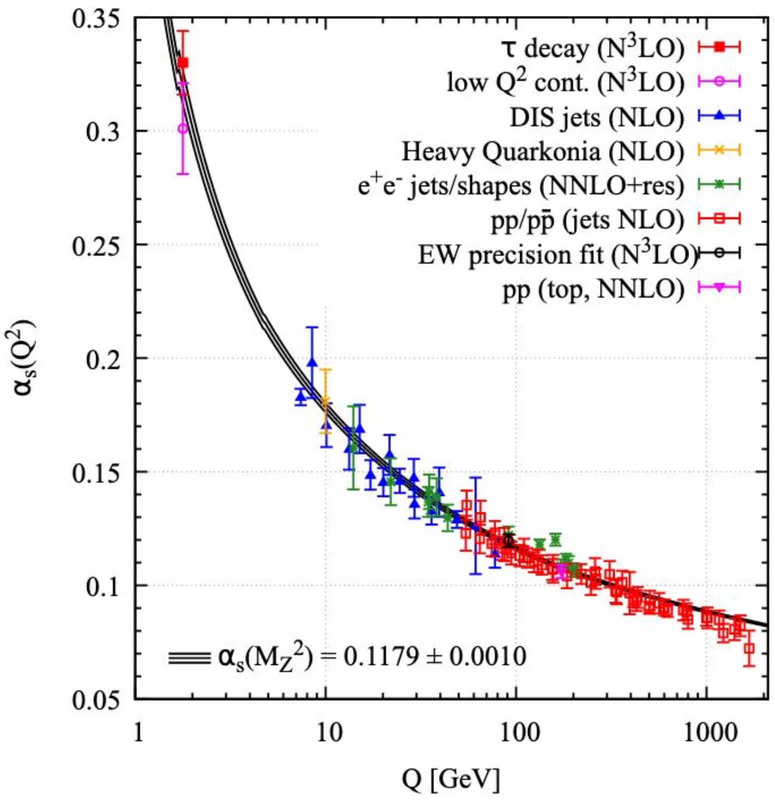
- plot $\alpha_{\rm EM}$ as a function of $\mu^2 \in (m_e^2, 10^4)$
- include vertical lines indicating the mass thresholds $m_e^2,\ m_\mu^2,\ m_\tau^2$





Solving QCD's beta function

The running of the strong coupling



$$a_S(\mu^2) = \frac{\alpha_S(\mu^2)}{4\pi}$$

$$\frac{d a_S}{d \ln \mu^2} = \beta(a_s) = -\left(\beta_0 a_S^2 + \beta_1 a_S^3 + \ldots\right)$$

$$\beta_0 = 11 - \frac{2}{3} N_f \qquad \beta_1 = 102 - \frac{38}{3} N_f$$

We need a boundary condition to solve the RGE

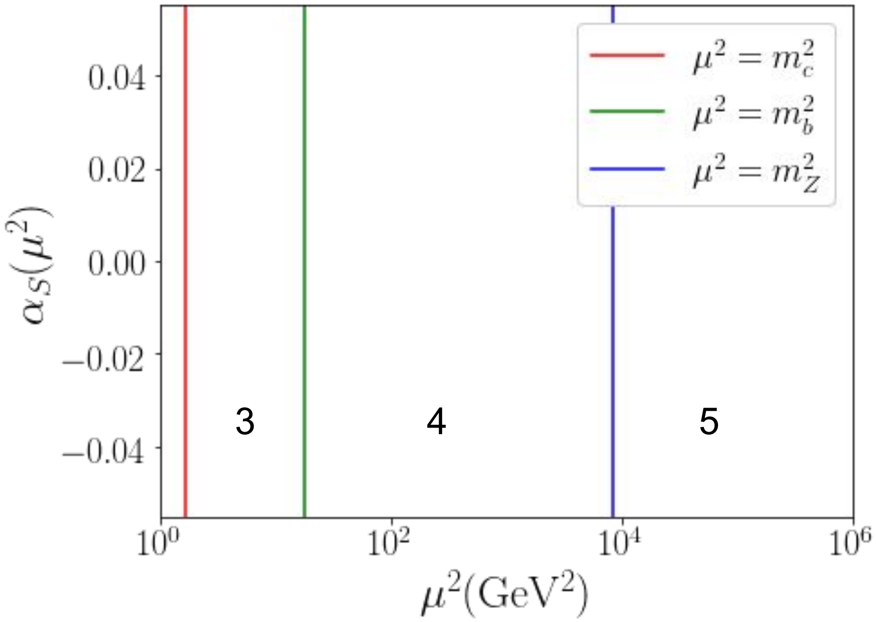
The beta function is discontinuous

Solving the QCD beta function

scale	N_f	active flavors
$\mu < m_c$	3	u, d, s
$m_c \leq \mu < m_b$	4	u, d, s, c
$m_b \leq \mu$	5	u, d, s, c, b

$$\frac{da_S}{d\ln\mu^2} = \beta(a_s) = -\left(\beta_0 a_S^2 + \beta_1 a_S^3 + \ldots\right)$$

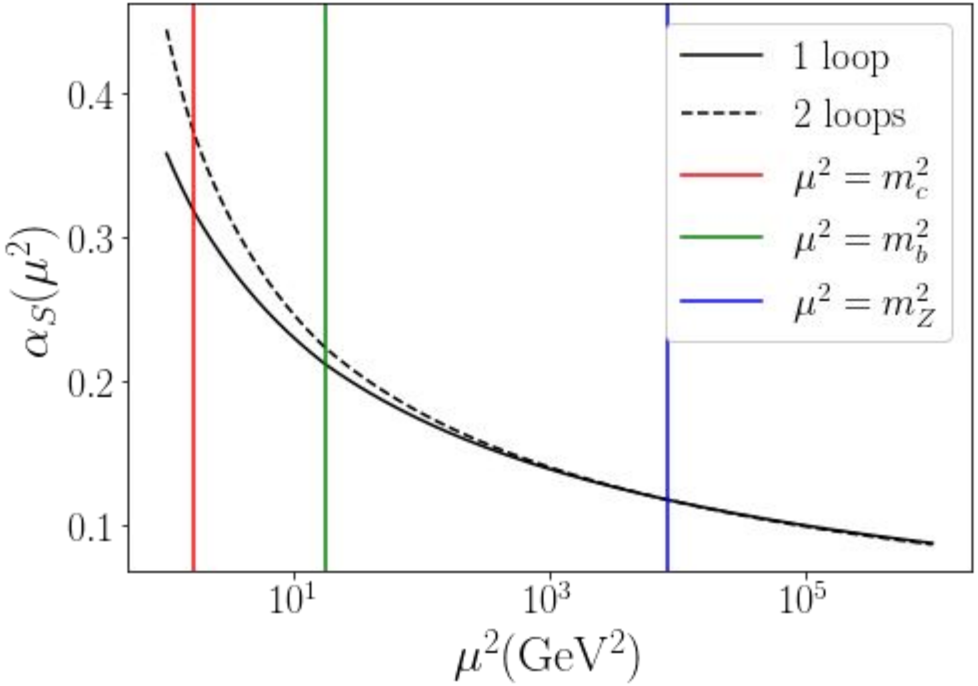
To solve the RGE at any scale we need boundary conditions for 3,4,5 flavors



Continue to Jupyter notebook

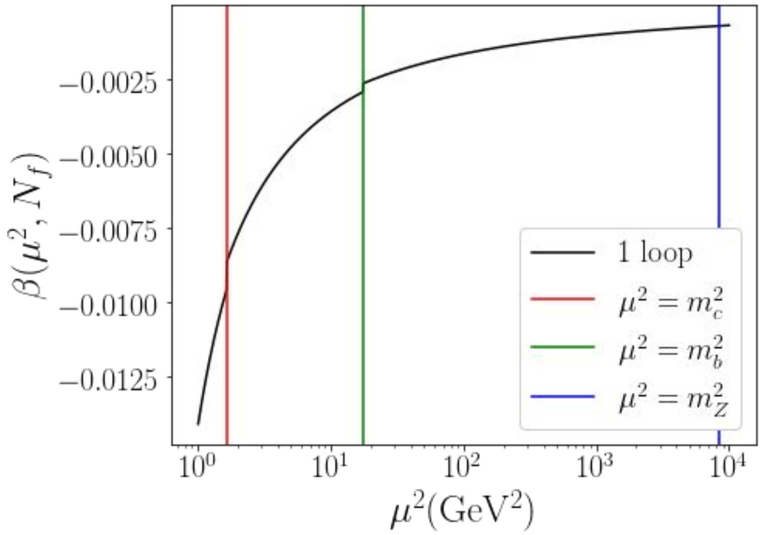
Exercise 2.A (time: 5 mins)

- plot α_S as a function of $\mu^2 \in (1, 10^4)$
- make the plot using 1-loop and 2-loops
- include vertical lines indicating the mass thresholds $m_c^2,\ m_b^2$ and m_Z^2



Exercise 2.B (time: 5 mins)

- plot $\beta(\mu^2)$ as a function of $\mu^2 \in (1, 10^4)$
- using 2-loops (order=1)
- include vertical lines indicating the mass thresholds $m_c^2,\ m_b^2$ and m_Z^2
- **Hint**:
  - for a given μ^2, compute `a` via `get_a` and `Nf` via `get_Nf`
  - use the function `beta_func(a,Nf)` to get the numerical value of the beta function



Outline

Lecture 1

- Motivations
- QCD carpentry setup
- Solving QCD's beta function

Lecture 2

- Mellin transforms
- Solving DGLAP
- Modeling input scale PDFs

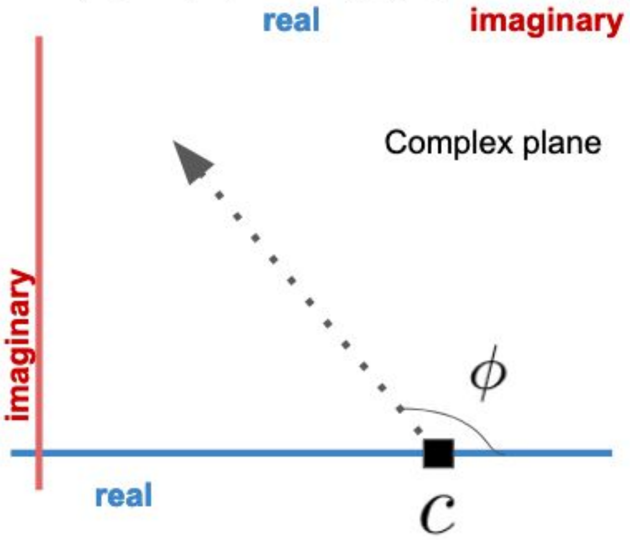
Lecture 3

- DIS theory
- World DIS data
- The chi2 function
- Global analysis

Lecture 4

- Bayesian inference
  - Maximum likelihood
  - MC methods
- JAM history
- Machine learning

$$N(z) = [c + z\cos(\phi)] + i[z\sin(\phi)]$$



Mellin transforms

Mellin transforms

Mellin transform of f(x)

$$F(N) = \int_0^1 dx x^{N-1} f(x)$$

Can be done numerically

$$f(x) = \frac{1}{2\pi i} \int_c dN x^{-N} F(N)$$

Inverse Mellin transform

Complex contour integration

Numerical implementation

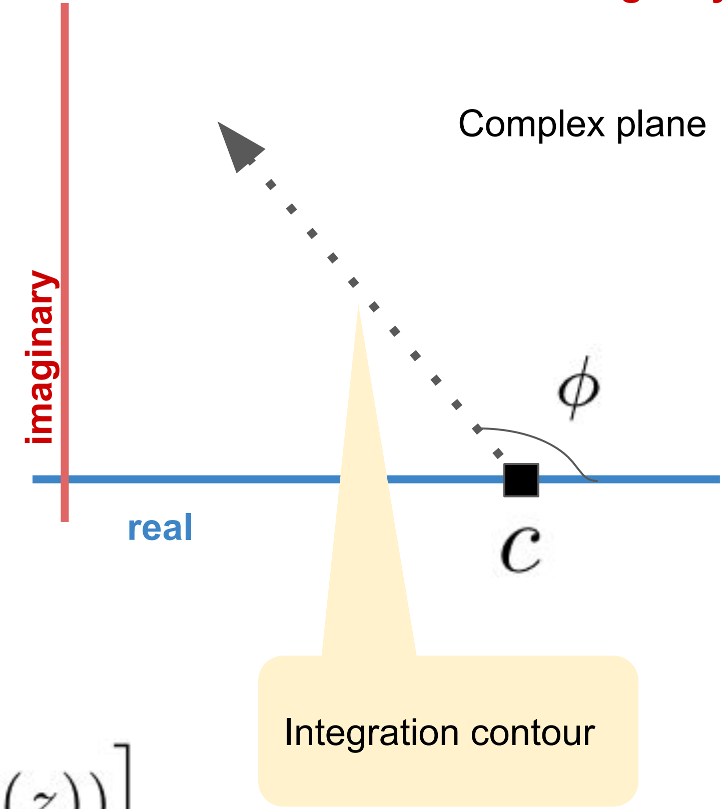
$$N(z) = [c + z\cos(\phi)] + i[z\sin(\phi)]$$

real imaginary

$$f(x) = \frac{1}{2\pi i}\int_c dN x^{-N} F(N)$$

$$N(z) = c + ze^{i\phi}$$

$$f(x) = \frac{1}{\pi}\int_0^\infty dz \mathrm{Im}\left[e^{i\phi} x^{-N(z)} F(N(z))\right]$$



Example

$$f(x) = x$$

$$F(N) = \int_0^1 dx x^{N-1} f(x)$$

$$F(N) = \frac{1}{N+1} x^{N+1}\Big|_0^1 = \frac{1}{N+1}$$

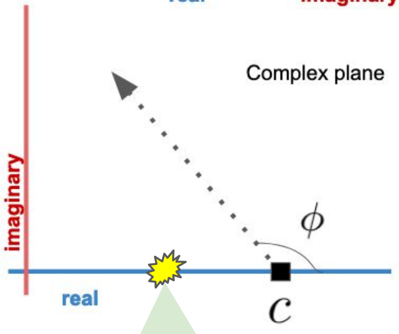
Pole at N = -1

$$x = \frac{1}{\pi}\int_0^\infty dz \mathrm{Im}\left[e^{i\phi} x^{-N(z)} \frac{1}{N(z)+1}\right]$$

C is chosen so that all the poles are in the left of C

Phi is chosen to be greater than pi/2

$$N(z) = \underbrace{[c + z\cos(\phi)]}_{\text{real}} + i\underbrace{[z\sin(\phi)]}_{\text{imaginary}}$$



Right most Pole of F(N)

Gaussian Quadrature

Only for range -1 to 1

$$\int_{-1}^{1} dx\ g(x) \approx \sum_{i=1}^{n} w_i g(x_i)$$

For arbitrary range

$$\int_{a}^{b} dz\ g(z) \approx \frac{b-a}{2} \sum_{i=1}^{n} w_i\ g\left(\frac{1}{2}(b-a)x_i + \frac{1}{2}(a+b)\right)$$

Inverse mellin transform with Gaussian Quadrature

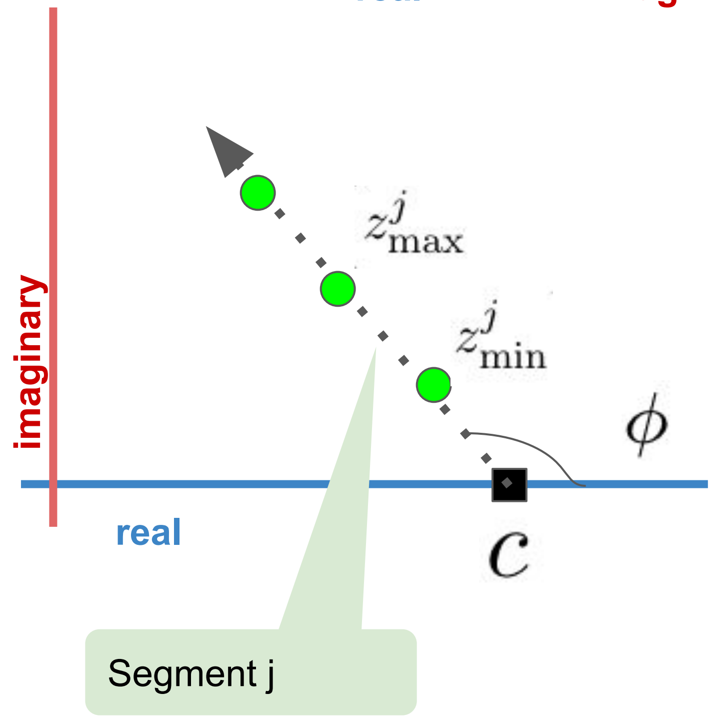
$$f(x) = \frac{1}{\pi} \int_0^{\infty} dz \mathrm{Im} \left[e^{i\phi} x^{-N(z)} F(N(z)) \right]$$

$$\Downarrow$$

$$f(x) \approx \frac{1}{\pi} \boxed{\sum_{j=1}^{k}} \frac{1}{2} \left(z^j_{\max} - z^j_{\min} \right) \sum_i w_i \mathrm{Im} \left[e^{i\phi} x^{-N(z_i^j)} F(N(z_i^j)) \right]$$

We only need to know F at the segment gaussian points

$$N(z) = \underbrace{[c + z\cos(\phi)]}_{\text{real}} + i\underbrace{[z\sin(\phi)]}_{\text{imaginary}}$$



Mellin convolutions

Definition of a convolution of two functions

$$\sigma(z) = \int_z^1 \frac{d\xi}{\xi} h(\xi) f\left(\frac{z}{\xi}\right) \Rightarrow \Sigma(N) = \int_0^1 dz\, z^{N-1} \sigma(z)$$

$$\Sigma(N) = H(N)\, F(N)$$

$$H(N) = \int_0^1 dy\, y^{N-1} h(y)$$

$$F(N) = \int_0^1 dy\, y^{N-1} f(y)$$

Mellin transform makes a convolution an ordinary product

Why Mellin transforms?

The kernels are known analytically. They are called “splitting functions”

Continue to Jupyter notebook

PDFs obeys a system of integro differential equations (DGLAP)

This is a “mellin convolution”

$$\frac{\partial}{\partial \ln \mu^2} f_{j/H}(\xi, \mu) = \sum_{j'} \int_{\xi}^{1} \frac{dz}{z} P_{jj'}(z, g) f_{j'/H}(\xi/z, \mu)$$

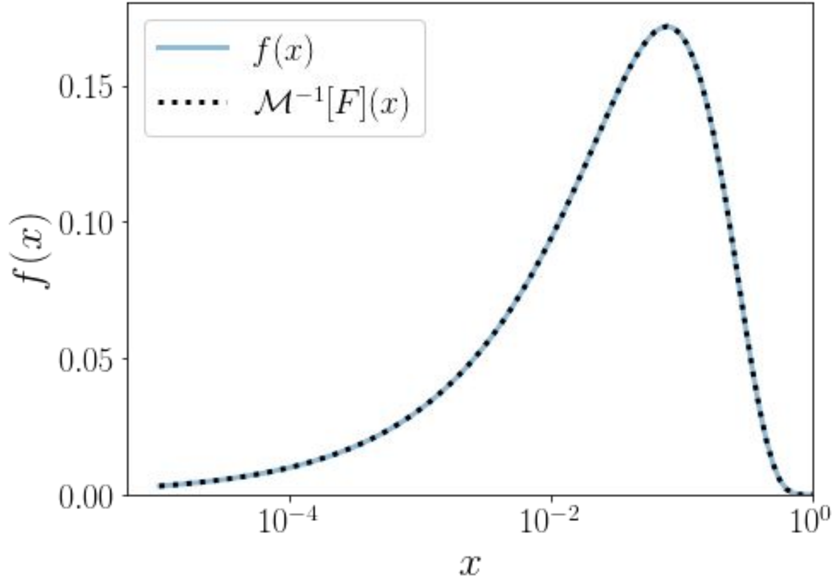
$$\Downarrow$$

This is a matrix equation

$$\boxed{\frac{\partial}{\partial \ln \mu^2} F_{j/H}(N, \mu) = \sum_{j'} P_{jj'}(N, \mu) \; F_{j'/H}(N, \mu)}$$

Exercise 3.A (time: 5 mins)

- Lets try $f(x) = x^a(1-x)^b$ with $a = -0.5$ and $b = 3$.
- Plot $xf(x)$ vs. x and the inverse mellin transform for $0 < x < 1$ (use log scale for the x axis)
- **Hint**: the mellin transform of f is $F(N) = \Gamma(N+a)\Gamma(b+1)/\Gamma(N+a+b+1)$
- **Hint**: Gamma function is available via `gamma(...)`
- **Attention**: the pole of F is at $N = -a$. Choose $c > -a$



Exercise 3.B (time: 10 mins)

Consider the convolution

$$\sigma(z) = \int_z^1 \frac{dx}{x} f(x) g\left(\frac{z}{x}\right)$$

The mellin transform is

$$\Sigma(N) = F(N)G(N)$$

with

$$F(N) = \int_0^1 x^{N-1} f(x)$$

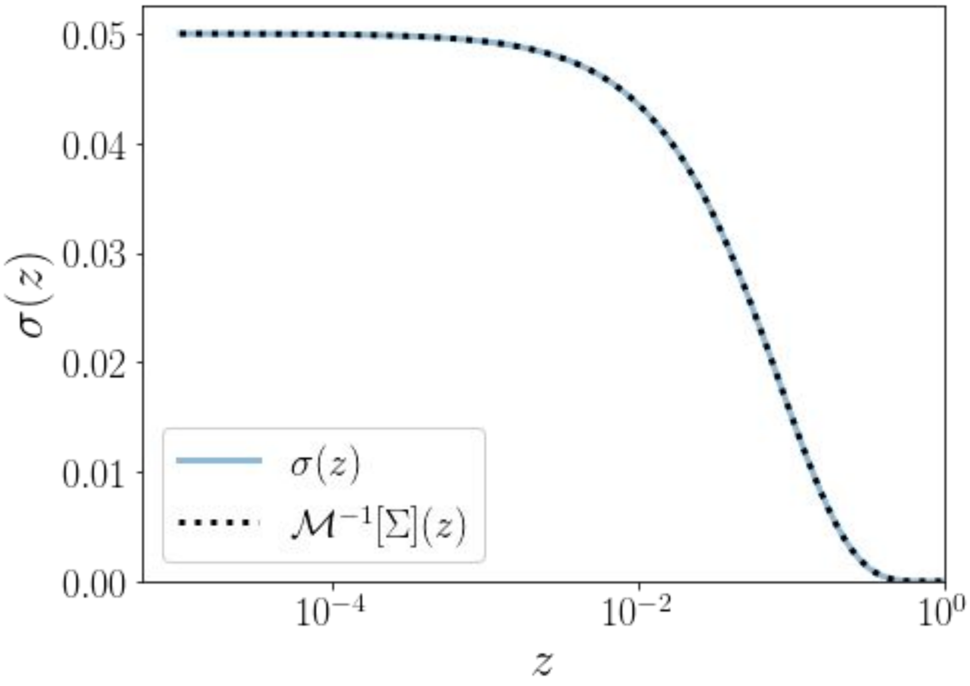
$$G(N) = \int_0^1 x^{N-1} g(x)$$

Using

$f(x) = x^a(1-x)^b$ with $a = -0.5$ and $b = 3$.

$g(x) = x^c(1-x)^d$ with $c = 1.0$ and $d = 3$

- Plot: $z\sigma(z)$ vs. z
- Plot: $z\mathcal{M}^{-1}(\Sigma)$ vs. z
- use $0 < z < 1$



$$\frac{\partial}{\partial \ln \mu^2} f_{j/H}(\xi, \mu) = \sum_{j'} \int_{\xi}^{1} \frac{dz}{z} P_{jj'}(z, g) f_{j'/H}(\xi/z, \mu)$$

Solving DGLAP

DGLAP in Mellin space

Splitting kernels

$$\frac{\partial}{\partial \ln \mu^2} f_{j/H}(\xi, \mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} P_{jj'}(z, g) f_{j'/H}(\xi/z, \mu)$$

System of integro-differential equations

$$\downarrow$$

$$\frac{\partial}{\partial \ln \mu^2} F_{j/H}(N, \mu) = \sum_{j'} P_{jj'}(N, \mu)\ F_{j'/H}(N, \mu)$$

Ordinary system of differential equations

Can be solved Analytically!

Flavor composition

$$\frac{\partial}{\partial \ln \mu^2} F_{j/H}(N,\mu) = \sum_{j'} P_{jj'}(N,\mu)\ F_{j'/H}(N,\mu)$$

11 equations for 5 active quark flavors + glue

$$F_{j/H}(N) = F_j$$

$$F_{q^\pm} = F_q \pm F_{\bar{q}}$$

$$\begin{aligned} F_{\pm 3} &= F_{u^\pm} - F_{d^\pm}, \\ F_{\pm 8} &= F_{u^\pm} + F_{d^\pm} - 2F_{s^\pm}, \\ F_{\pm 15} &= F_{u^\pm} + F_{d^\pm} + F_{s^\pm} - 3F_{c^\pm}, \\ F_{\pm 24} &= F_{u^\pm} + F_{d^\pm} + F_{s^\pm} + F_{c^\pm} - 4F_{b^\pm}, \end{aligned}$$

$$F_{\pm} = F_{u^\pm} + F_{d^\pm} + F_{s^\pm} + F_{c^\pm} + F_{b^\pm}$$

$$F_{j/H}(N) \Rightarrow \boxed{F_{\pm j}}\ \boxed{F_{\pm}}\ \boxed{F_g}$$

Just linear transformations

11 = 8 + 2 + 1

Flavor singlet and non-singlet evolution

$$\frac{\partial}{\partial \ln \mu^2} F_{j/H}(N,\mu) = \sum_{j'} P_{jj'}(N,\mu)\; F_{j'/H}(N,\mu)$$

Non singlet combinations decouples from glue

$$\boxed{\frac{\partial F_{\pm j}}{\partial \ln \mu^2} = P^{\pm}_{\rm NS} F_{\pm j}} \qquad \boxed{\frac{\partial F_{-}}{\partial \ln \mu^2} = P^{-}_{\rm NS} F_{-}}$$

Non-singlet evolution

$$\boxed{\frac{\partial}{\partial \ln \mu^2} \begin{pmatrix} F_+ \\ F_g \end{pmatrix} = \begin{pmatrix} P_{qq} & P_{qg} \\ P_{gq} & P_{gg} \end{pmatrix} \begin{pmatrix} F_+ \\ F_g \end{pmatrix}}$$

Singlet evolution

Solving the non-singlet evolution equations

$$\boxed{\frac{\partial F_{\pm j}}{\partial \ln \mu^2} = P_{\rm NS}^{\pm} F_{\pm j}}$$

$$P_{ij}(a_S) = \sum_{m=0}^{\infty} a_S^{m+1}(\mu) P_{ij}^{(m)}$$

Splitting functions depend only on alphaS

$$\Downarrow$$

$$\frac{da_S}{d\ln \mu^2} = \beta(a_s)$$

$$\Downarrow$$

$$\boxed{\frac{\partial F_{\pm j}}{\partial a_S} = -\frac{1}{\beta_0 a_S} P_{\rm NS}^{\pm(0)} F_{\pm j}}$$

$$\Rightarrow$$

alphaS at the final scale

$$\boxed{F_{\pm j}(a_S) = \left(\frac{a_S}{a_0}\right)^{-P_{\rm NS}^{\pm(0)}/\beta_0} F_{\pm j}(a_0)}$$

alphaS at the input scale

Solving the singlet evolution equations

Eigenvalue decomposition

$$\frac{\partial}{\partial \ln \mu^2}\begin{pmatrix} F_+ \\ F_g \end{pmatrix} = \begin{pmatrix} P_{qq} & P_{qg} \\ P_{gq} & P_{gg} \end{pmatrix}\begin{pmatrix} F_+ \\ F_g \end{pmatrix}$$

$$\Downarrow$$

$$\frac{\partial}{\partial a_S}\begin{pmatrix} F_+ \\ F_g \end{pmatrix} = \frac{-1}{\beta_0 a_S}\begin{pmatrix} P_{qq}^{(0)} & P_{qg}^{(0)} \\ P_{gq}^{(0)} & P_{gg}^{(0)} \end{pmatrix}\begin{pmatrix} F_+ \\ F_g \end{pmatrix}$$

$$\boldsymbol{R}_0 = \frac{1}{\beta_0}\begin{pmatrix} P_{qq}^{(0)} & P_{qg}^{(0)} \\ P_{gq}^{(0)} & P_{gg}^{(0)} \end{pmatrix} = r_- \boldsymbol{e}_- + r_+ \boldsymbol{e}_+$$

$$\boldsymbol{e}_\pm = \frac{1}{r_\pm - r_\mp}\left[\boldsymbol{R}_0 - r_\mp \boldsymbol{I}\right]$$

$$r_\pm = \frac{1}{2\beta_0}\left[P_{qq}^{(0)} + P_{gg}^{(0)} \pm \sqrt{\left(P_{qq}^{(0)} - P_{gg}^{(0)}\right)^2 + 4P_{qg}^{(0)}P_{gq}^{(0)}}\right]$$

$$\boxed{\begin{pmatrix} F_+(a_S) \\ F_g(a_S) \end{pmatrix} = \left[\boldsymbol{e}_-\left(\frac{a_S}{a_0}\right)^{-r_-} + \boldsymbol{e}_+\left(\frac{a_S}{a_0}\right)^{-r_+}\right]\begin{pmatrix} F_+(a_0) \\ F_g(a_0) \end{pmatrix}}$$

Flavor decomposition

$$\boxed{F_{\pm j}} \quad \boxed{F_{\pm}} \quad \boxed{F_g}$$

$$F_{b^\pm} = (F_- - F_{\pm 24})/5$$

$$F_{c^\pm} = F_{b^\pm} + (F_{\pm 24} - F_{\pm 15})/4$$

$$F_{s^\pm} = F_{c^\pm} + (F_{\pm 15} - F_{\pm 8})/3$$

$$F_{d^\pm} = F_{s^\pm} + (F_{\pm 8} - F_{\pm 3})/2$$

$$F_{u^\pm} = F_{s^\pm} + (F_{\pm 8} + F_{\pm 3})/2$$

$$F_q = \frac{1}{2}(F_{q^+} + F_{q^+})$$

$$F_{\bar{q}} = \frac{1}{2}(F_{q^+} - F_{q^-})$$

Evolution flow

Evolved PDFs

μ^2

$\boxed{F_{\pm j}}$ $\boxed{F_{\pm}}$ $\boxed{F_g}$

$$F_{b^\pm} = (F_- - F_{\pm 24})/5$$
$$F_{c^\pm} = F_{b^\pm} + (F_{\pm 24} - F_{\pm 15})/4$$
$$F_{s^\pm} = F_{c^\pm} + (F_{\pm 15} - F_{\pm 8})/3$$
$$F_{d^\pm} = F_{s^\pm} + (F_{\pm 8} - F_{\pm 3})/2$$
$$F_{u^\pm} = F_{s^\pm} + (F_{\pm 8} + F_{\pm 3})/2$$

$$F_{j/H}(N)$$

$$F_{\pm j}(a_S) = \left(\frac{a_S}{a_0}\right)^{-P_{\rm NS}^{\pm(0)}/\beta_0} F_{\pm j}(a_0)$$

$$\begin{pmatrix} F_+(a_S) \\ F_g(a_S) \end{pmatrix} = \left[\boldsymbol{e}_- \left(\frac{a_S}{a_0}\right)^{-r_-} + \boldsymbol{e}_+ \left(\frac{a_S}{a_0}\right)^{-r_+} \right] \begin{pmatrix} F_+(a_0) \\ F_g(a_0) \end{pmatrix}$$

μ_0^2

Input scale PDFs

$$F_{\pm 3} = F_{u^\pm} - F_{d^\pm},$$
$$F_{\pm 8} = F_{u^\pm} + F_{d^\pm} - 2F_{s^\pm},$$
$$F_{\pm 15} = F_{u^\pm} + F_{d^\pm} + F_{s^\pm} - 3F_{c^\pm},$$
$$F_{\pm 24} = F_{u^\pm} + F_{d^\pm} + F_{s^\pm} + F_{c^\pm} - 4F_{b^\pm},$$
$$F_{\pm} = F_{u^\pm} + F_{d^\pm} + F_{s^\pm} + F_{c^\pm} + F_{b^\pm}$$

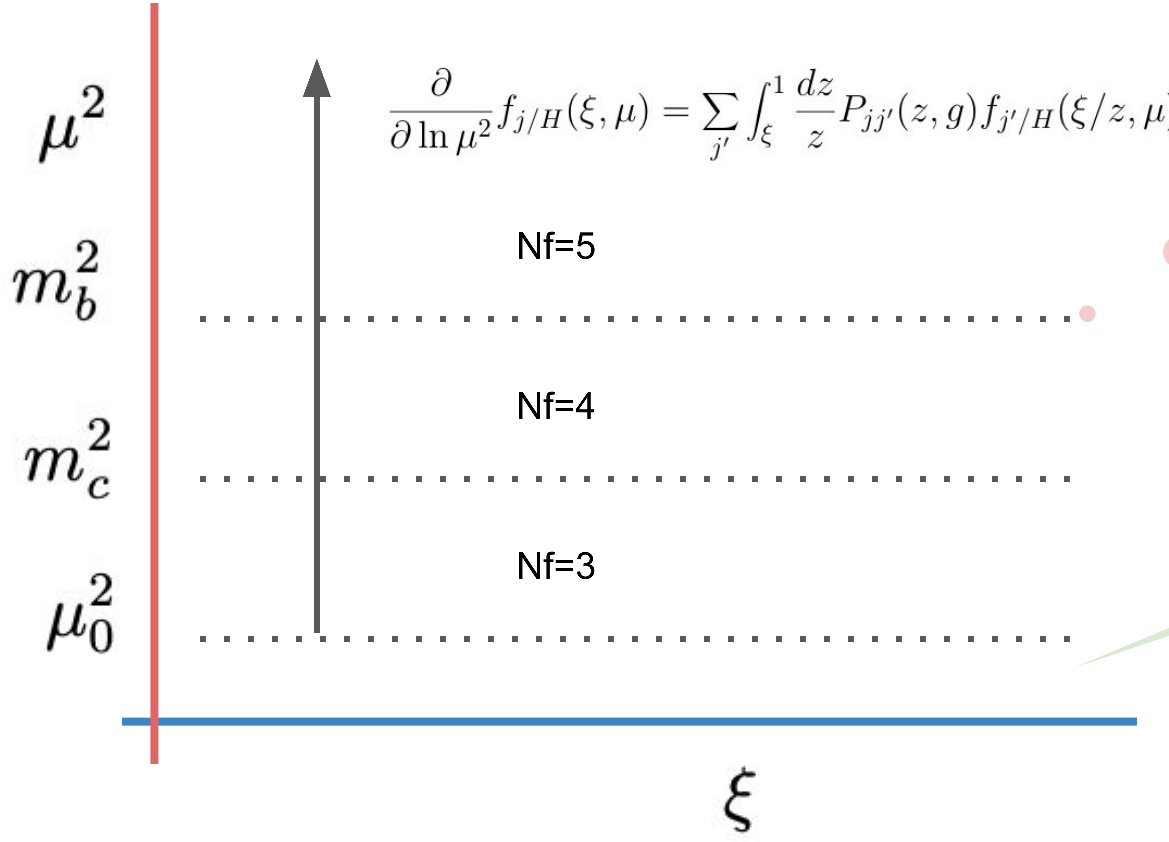
$$F_{j/H}(N)$$

$\boxed{F_{\pm j}}$ $\boxed{F_{\pm}}$ $\boxed{F_g}$

ξ

Boundary conditions

$$\frac{\partial}{\partial \ln \mu^2} f_{j/H}(\xi,\mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} P_{jj'}(z,g) f_{j'/H}(\xi/z,\mu)$$



We need boundary conditions at each mass threshold

Scales where DGLAP is not applicable

Continue to Jupyter notebook

Exercise 4.A (time: 5 mins)

- Set N=1 in the mellin class via `conf['mellin'].N=np.array([1])`
- check the valence number sum rule at LO
- **Hint:** $\frac{\partial q_v}{\partial \ln \mu^2}(N) = P^-_{NS}(N)q_v(N) = 0$ for $N = 1$

```
Nf=4
Q2ini=conf['aux'].mc2
Q2fin=conf['aux'].mb2
output = dglap.evolve(BC,Q2ini,Q2fin,Nf)
print('um=',output['um'])
print('dm=',output['dm'])
```

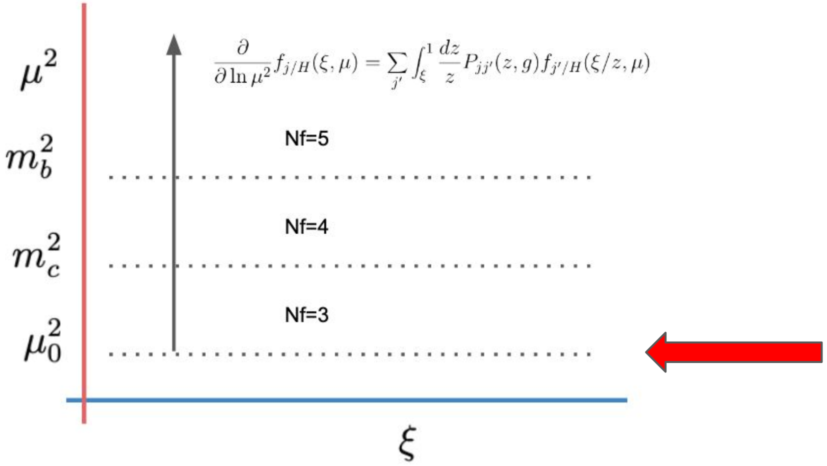
```
um= [2.+0.j]
dm= [1.+0.j]
```

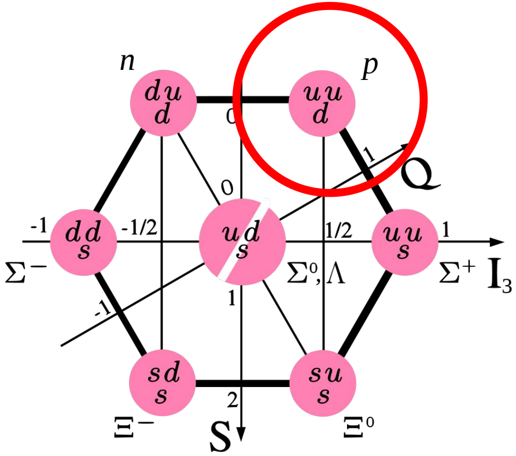
Exercise 4.B (time: 5 mins)

- Set N=2 in the mellin class via `conf['mellin'].N=np.array([2])`
- check the momentum sum rule at LO
- **Hint:** $\frac{\partial(\Sigma+G)}{\partial \ln \mu^2}(N) = (P_{qq} + P_{gq})\Sigma + (P_{qg} + P_{gg})G = 0$ for $N = 2$

```
Nf=4
Q2ini=conf['aux'].mc2
Q2fin=conf['aux'].mb2
output = dglap.evolve(BC,Q2ini,Q2fin,Nf)
print('sigma+g (mu) =',output['g']+output['sigma'])
```

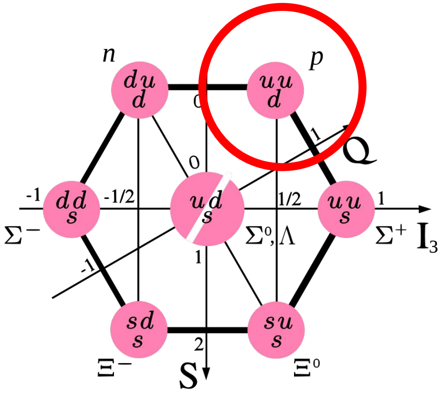
```
sigma+g (mu) = [0.99999997+0.j]
```





Modeling **input** scale PDFs

Sum rules for proton PDFs

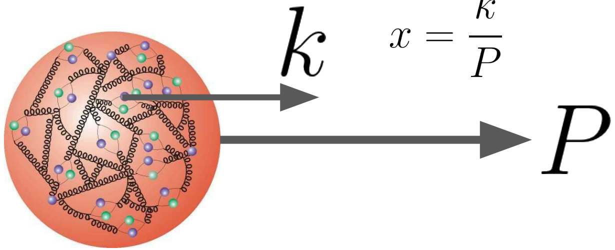


Valence number sum rules

$$\int_0^1 dx\ [f_{u/p}(x) - f_{\bar{u}/p}(x)] = 2$$

$$\int_0^1 dx\ [f_{d/p}(x) - f_{\bar{d}/p}(x)] = 1$$

$$\int_0^1 dx\ [f_{s/p}(x) - f_{\bar{s}/p}(x)] = 0$$

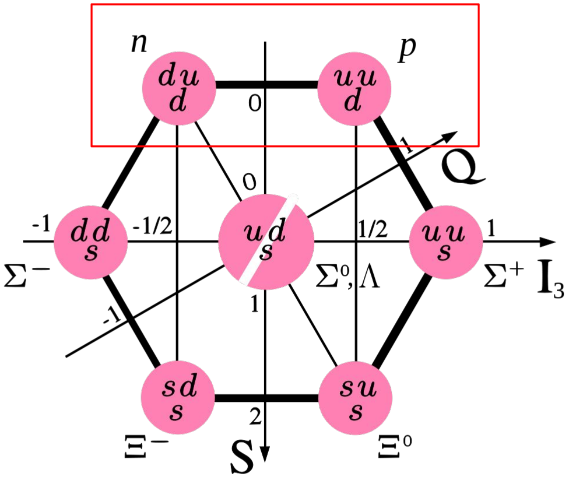


$$x = \frac{k}{P}$$

Momentum sum rule

$$\int_0^1 dx\ x\ [f_g + f_{u^+} + f_{d^+} + f_{s^+} + f_{c^+} + f_{b^+}] = 1$$

neutron PDFs ?



$$m_p \simeq m_n$$

Isospin symmetry

$$f_{u/n} = f_{d/p}$$

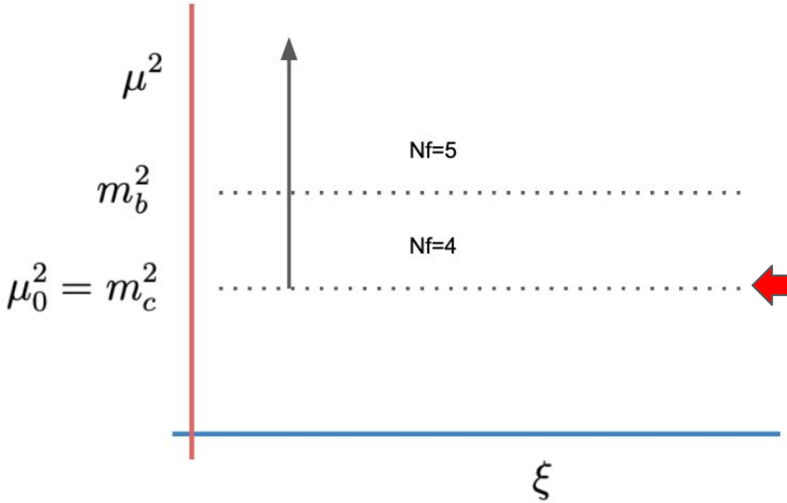
$$f_{d/n} = f_{u/p}$$

$$f_{\bar{u}/n} = f_{\bar{d}/p}$$

$$f_{\bar{d}/n} = f_{\bar{u}/p}$$

PDF parametrization (workbook setup)

Continue to Jupyter notebook



$$u(\xi, \mu_0^2) = u_v(\xi, \mu_0^2) + 2\bar{u}(\xi, \mu_0^2)$$

$$d(\xi, \mu_0^2) = d_v(\xi, \mu_0^2) + 2\bar{d}(\xi, \mu_0^2)$$

$$\bar{u}(\xi, \mu_0^2) = S_1(\xi, \mu_0^2) + \bar{u}_0(\xi, \mu_0^2)$$

$$\bar{d}(\xi, \mu_0^2) = S_1(\xi, \mu_0^2) + \bar{d}_0(\xi, \mu_0^2)$$

$$s(\xi, \mu_0^2) = S_2(\xi, \mu_0^2) + s_0(\xi, \mu_0^2)$$

$$\bar{s}(\xi, \mu_0^2) = S_2(\xi, \mu_0^2) + \bar{s}_0(\xi, \mu_0^2)$$

Generic template function

$$T(\xi; \boldsymbol{a}) = \mathcal{M} \frac{\xi^{\alpha} (1-\xi)^{\beta} \left(1 + \gamma\sqrt{\xi} + \delta\xi\right)}{\int_0^1 \mathrm{d}\xi\, \xi^{\alpha+1} (1-\xi)^{\beta} \left(1 + \gamma\sqrt{\xi} + \delta\xi\right)} \rightarrow u_v,\ d_v,\ \bar{u}_0,\ \bar{d}_0,\ s_0,\ \bar{s}_0,\ S_1,\ S_2$$

Exercise 5.A (time: 5 mins)

- Physical proton pdfs will need $\int_0^1 \; dx\, u_v(x) = 2$ and $\int_0^1 \; dx\, d_v(x) = 1$
- Modify the `BC` we use above and show that the valence number sum rules don't evolve
- **hint**:
  - Use the function `dglap.evolve(BC,Q2ini,Q2fin,Nf)` to evolve
  - Set N=1 in the mellin class via `conf['mellin'].N=np.array([1])`
  - Set `um = np.array([2])`
  - Set `dm = np.array([1])`
  - Printe the output values `um` and `dm`

```
Nf=4
Q2ini=conf['aux'].mc2
Q2fin=conf['aux'].mb2
output = dglap.evolve(BC,Q2ini,Q2fin,Nf)
print('um=',output['um'])
print('dm=',output['dm'])
```

```
um= [2.+0.j]
dm= [1.+0.j]
```

Exercise 5.B (time: 5 mins)

- A physical proton pdfs will need $\int_0^1 dx\; x\; [\Sigma(x) + g(x)] = 1$
- Modify the `BC` we use above and show that the momentum sum rules don't evolve
- **hint**:
  - Set N=2 in the mellin class via `conf['mellin'].N=np.array([2])`
  - Set `g= 1-up-dp-sp`
  - Print the values of `output['g']+output['sigma'])`

```
Nf=4
Q2ini=conf['aux'].mc2
Q2fin=conf['aux'].mb2
output = dglap.evolve(BC,Q2ini,Q2fin,Nf)
print('sigma+g (mu) =',output['g']+output['sigma'])
```

```
sigma+g (mu) = [0.99999997+0.j]
```

Exercise 6 (time: 5 mins)

- Check the valence number and momentum sum rules at $\mu^2 = m_c^2, 10, 100, 1000$

```
mu^2=1.638400
uv-> (2.000000047183332, 2.4232824635816996e-08)
dv-> (0.9999999958918243, 3.46867490286229e-09)
sv-> (4.3347507561294274e-08, 2.1829166918001526e-08)
msr-> (0.9999999969010325, 6.567994703665647e-09)

mu^2=10.000000
```

```
/work/JAM/apps/anaconda3/envs/snakes3/lib/python3.6/site-pac
tegrationWarning: The integral is probably divergent, or slc
  """
```

```
uv-> (2.000000047183332, 2.4232824635816996e-08)
dv-> (0.9999999958918243, 3.46867490286229e-09)
sv-> (4.3347507561294274e-08, 2.1829166918001526e-08)
msr-> (0.9999999969010325, 6.567994703665647e-09)

mu^2=100.000000
uv-> (2.000000047183332, 2.4232824635816996e-08)
dv-> (0.9999999958918243, 3.46867490286229e-09)
sv-> (4.3347507561294274e-08, 2.1829166918001526e-08)
msr-> (0.9999999969010325, 6.567994703665647e-09)

mu^2=1000.000000
uv-> (2.000000047183332, 2.4232824635816996e-08)
dv-> (0.9999999958918243, 3.46867490286229e-09)
sv-> (4.3347507561294274e-08, 2.1829166918001526e-08)
msr-> (0.9999999969010325, 6.567994703665647e-09)
```

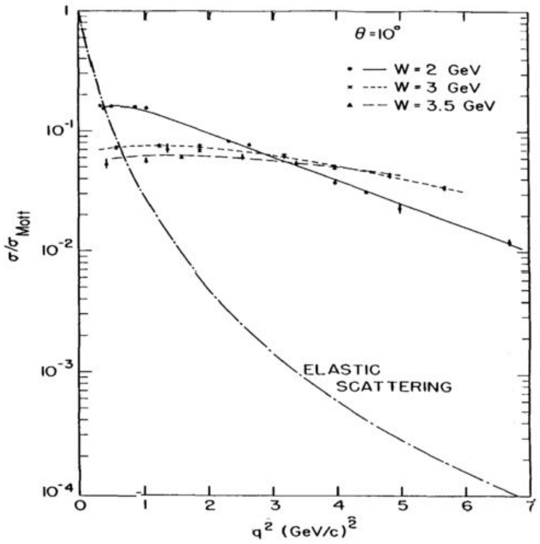
Outline

Lecture 1
- Motivations
- QCD carpentry setup
- Solving QCD's beta function

Lecture 2
- Mellin transforms
- Solving DGLAP
- Modeling input scale PDFs

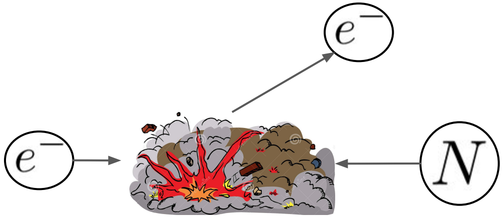
Lecture 3
- DIS theory
- World DIS data
- The chi2 function
- Global analysis

Lecture 4
- Bayesian inference
  - Maximum likelihood
  - MC methods
- JAM history
- Machine learning

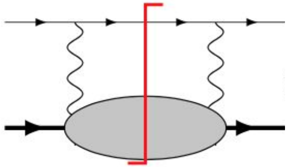


DIS theory

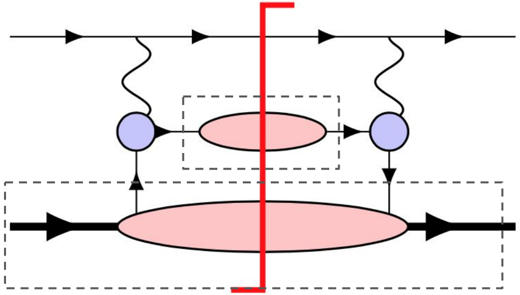
Factorization in deep-inelastic scattering (DIS)

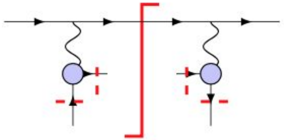


$$E' \frac{\mathrm{d}\sigma}{\mathrm{d}^3 l'}$$

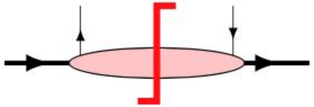


Approximations





Collision dependent factor

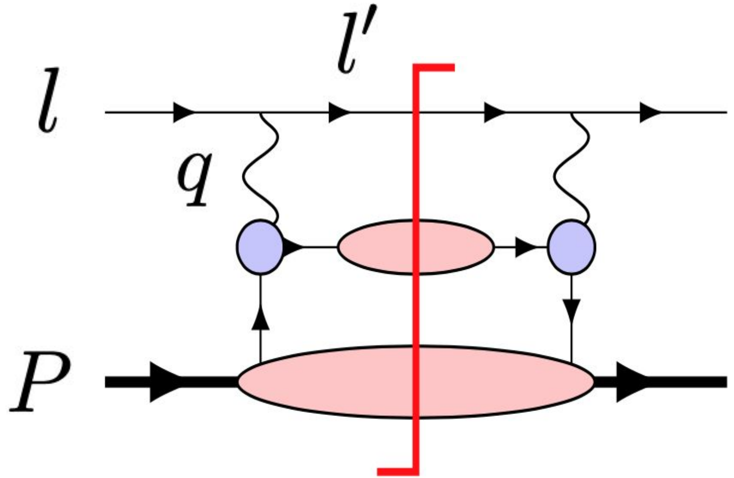


Internal structure

$$\sum_i \int_x^1 \frac{\mathrm{d}\xi}{\xi} H_i(\xi) \, f_i\left(\frac{x}{\xi}\right) + \mathcal{O}\left(\frac{m^2}{Q^2}\right)$$

Error of approximations

DIS kinematics



$$q = (l - l')$$

$$y = \frac{P \cdot q}{P \cdot l}$$

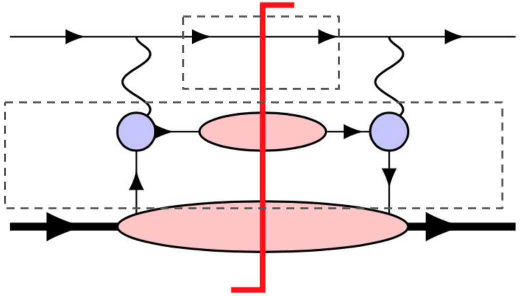
$$Q^2 = -q^2$$

$$x_{\rm bj} = \frac{Q^2}{2P \cdot q}$$

Think them as change of variables

$$l'(E', \theta', \phi') = l'(x_{\rm bj}, Q^2, \phi')$$

DIS factorization



$$\frac{d^2\sigma^i}{dx\,dy} = \frac{2\pi\alpha^2}{xyQ^2}\left(\left(Y_+ + 2x^2y^2\frac{M^2}{Q^2}\right)\boxed{F_2^i} - y^2\boxed{F_L^i} \mp Y_- x\boxed{F_3^i}\right)$$

$$F_i^p(x_{\mathrm{bj}}, Q^2) = \sum_q e_q^2 \int_{x_{\mathrm{bj}}}^1 \frac{d\xi}{\xi}\left[f_{q/p}(\xi,\mu^2) C_{q,i}\left(\frac{x_{\mathrm{bj}}}{\xi}, \frac{Q^2}{\mu^2}, \alpha_S(\mu^2)\right) + f_{g/p}(\xi,\mu^2) C_{g,i}\left(\frac{x_{\mathrm{bj}}}{\xi}, \frac{Q^2}{\mu^2}, \alpha_S(\mu^2)\right)\right]$$

Quark contributions

Gluon contributions

DIS in Mellin space

$$F_i^p(x_{\mathrm{bj}}, Q^2) = \sum_q e_q^2 \int_{x_{\mathrm{bj}}}^1 \frac{d\xi}{\xi} f_{q/p}(\xi, \mu^2) C_{q,i}\left(\frac{x_{\mathrm{bj}}}{\xi}, \frac{Q^2}{\mu^2}, \alpha_S(\mu^2)\right) + (q \to g)$$

$$\Big\downarrow$$

$$F_i^p(N, Q^2) = \sum_q e_q^2 \, f_{q/p}(N, \mu^2) C_{q,i}\left(N, \frac{Q^2}{\mu^2}, \alpha_S(\mu^2)\right) + (q \to g)$$

$$\Big\downarrow$$

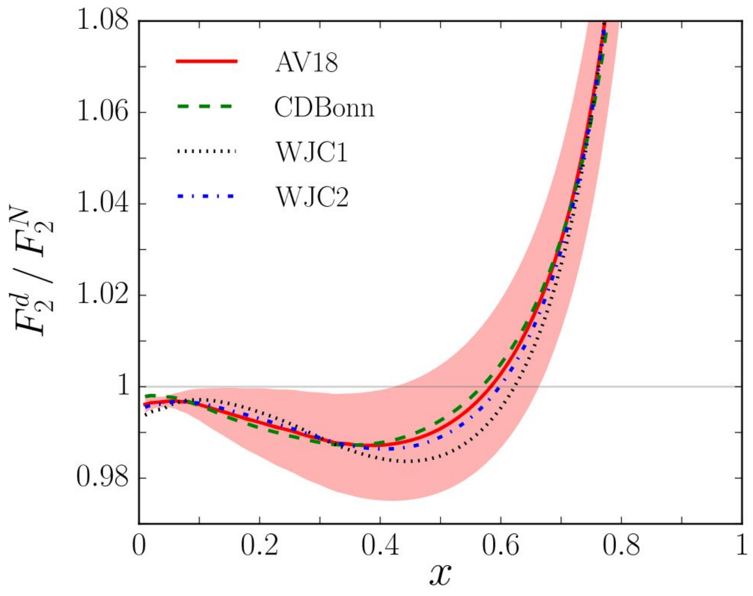
$$C_j(N) = C_j^{[0]}(N) + \frac{\alpha_S}{4\pi} C_j^{[1]}(N) + O(\alpha_S^2)$$

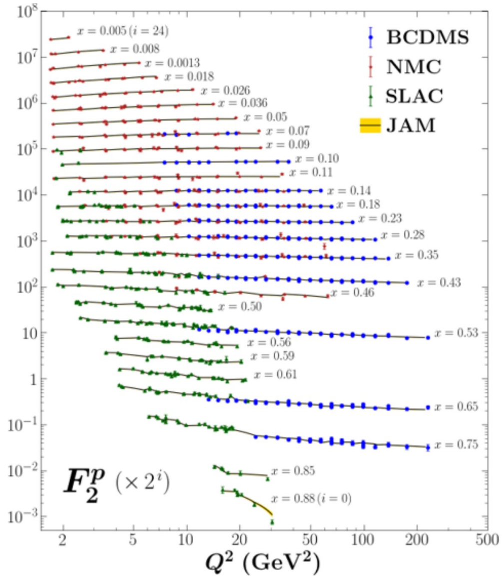
DIS with Deuteron target

$$F_i^d \approx \frac{1}{2}\left(F_i^p + F_i^n\right)$$

- This approximation ignores the "EMC" effect.
- If we ignore the large x_bj data, the approximation is ok

Accardi, Brady, Melnitchouk, Owens, NS

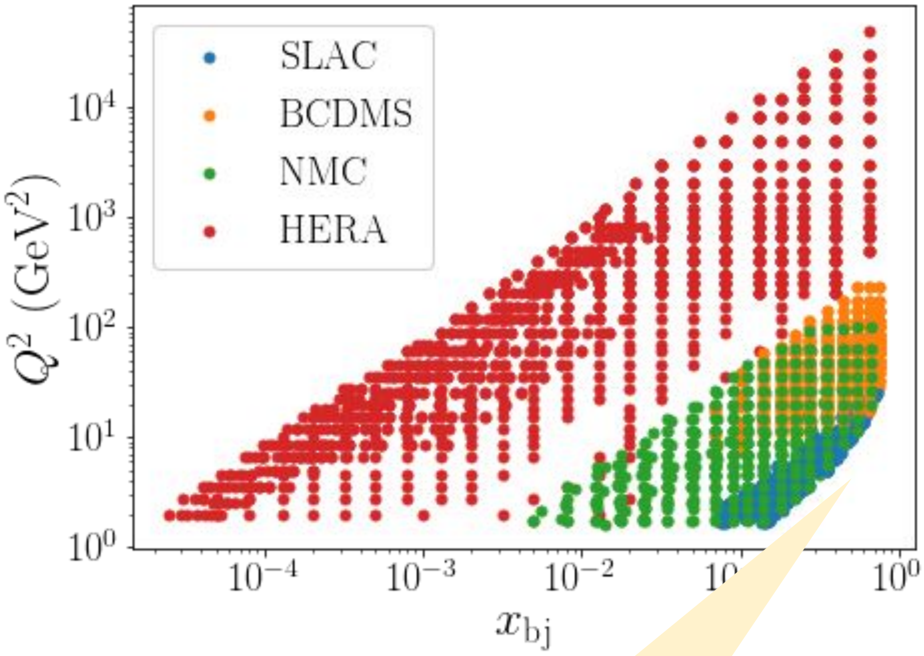




World DIS data

World DIS data

	idx	col	target	current	obs
0	10010	SLAC	p	NC	F2
1	10016	BCDMS	p	NC	F2
2	10020	NMC	p	NC	F2
3	10026	HERA II NC e+ (1)	p	NC	sig_r
4	10027	HERA II NC e+ (2)	p	NC	sig_r
5	10028	HERA II NC e+ (3)	p	NC	sig_r
6	10029	HERA II NC e+ (4)	p	NC	sig_r
7	10030	HERA II NC e-	p	NC	sig_r
8	10031	HERA II CC e+	p	CC	sig_r
9	10032	HERA II CC e-	p	CC	sig_r
10	10011	SLAC	d	NC	F2
11	10017	BCDMS	d	NC	F2
12	10021	NMC	d/p	NC	F2d/F2p



Jefferson Lab DIS data

DIS database

Continue to Jupyter notebook

QCDHUB / qcdcarpentry

Code | Issues | Pull requests | Actions | Proje

main · qcdcarpentry / database / idis / expdata /

nobuosato update

..	
10001.xlsx	update
10002.xlsx	update
10003.xlsx	update
10004.xlsx	update
10005.xlsx	update

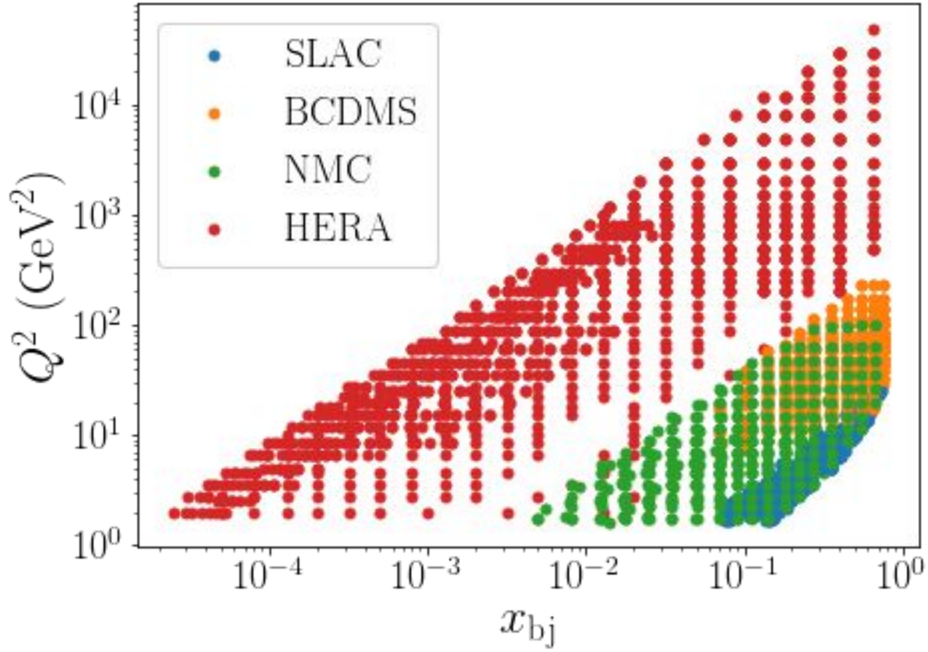
custminze loader for the experimental DIS data tables

```
class READER:

    def __init__(self):
        self.aux=conf['aux']

    def isnumeric(self,value):
        try:
            int(value)
            return True
        except:
            return False


    def get_X(self,tab):
        cols=tab.columns.values
        if any([c=='X' for c in cols])==False:
            if any([c=='W2' for c in cols]):
                tab['X']=pd.Series(tab['Q2']/(tab['W2']-self.aux.M2+ta
            elif any([c=='W' for c in cols]):
                tab['X']=pd.Series(tab['Q2']/(tab['W']**2-self.aux.M2+
            else:
                print('cannot retrive X values')
```

This class will load the excel files, add missing kinematic variables and transform the data into numpy arrays

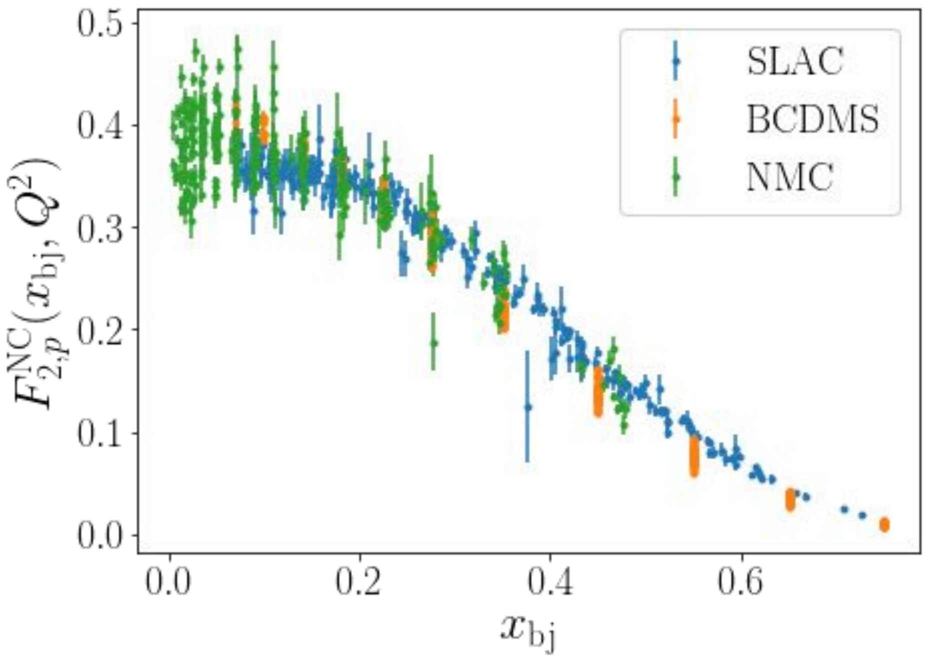
Exercise 7 (time: 5 mins)

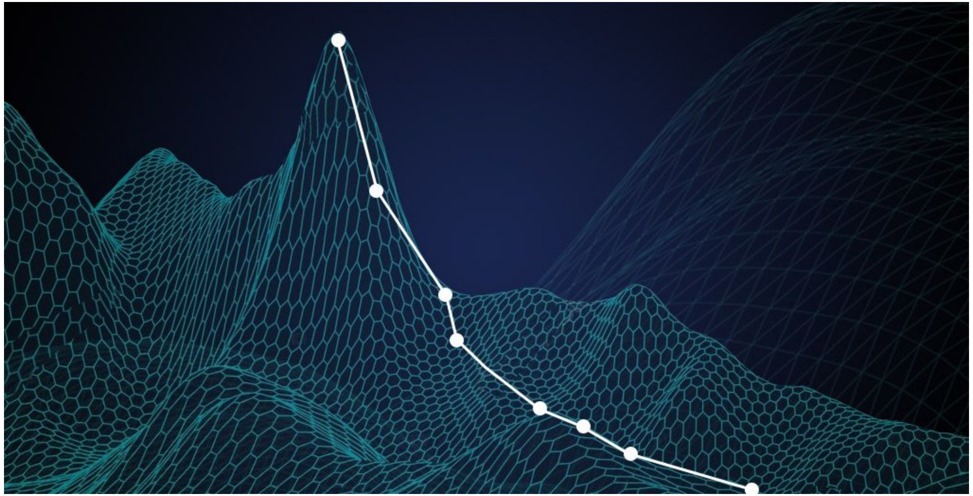
- plot the kinematics bins $x_{\mathrm{bj}},\ Q^2$ of the world DIS data sets
- **Hint**: use log scale for both axis



Exercise 8 (time: 5 mins)

- plot the values of $F_{2,p}^{\mathrm{NC}}$ as a function of x_{bj} from all the data sets
- **Hint**: use the colums `X`, `value`, `alpha` from `loss.tabs`





The Loss function

Anatomy of Chi2 function

Experimental data point

Point-by-point correlated systematic uncertainties

Theory with pdf parameters **a**

$$\chi^2(\boldsymbol{a}) = \sum_{i,e} \left(\frac{d_{i,e} - \sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}} \right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1 - N_e}{\delta N_e} \right)^2$$

Uncorrelated uncertainties added in quadrature

Overall normalization uncertainty

Anatomy of Chi2 function

Nuisance fitting parameter

Penalties for the Nuisance parameters

$$\chi^2(\boldsymbol{a}) = \sum_{i,e}\left(\frac{d_{i,e} - \sum_k \underline{r_e^k}\beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/\underline{N_e}}{\alpha_{i,e}}\right)^2 + \sum_k \left(\underline{r_e^k}\right)^2 + \left(\frac{1-\underline{N_e}}{\delta N_e}\right)^2$$

$$\frac{\partial\chi^2}{\partial r_e^k} = 0 \Rightarrow \boxed{\chi^2(\boldsymbol{a}) = \sum_{i,e}\left(\frac{d_{i,e} - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}}\right)^2 - \sum_{k,k'} B_{k,e}A^{-1}_{kk',e}B_{k',e}}$$

$$B_e^k = \sum_i \frac{\beta_{i,e}^k(d_{i,e} - T_{i,e}/N_e)}{\alpha_i^2} \qquad A_{kk',e} = \delta_{kk'} + \sum_i \frac{\beta_{i,e}^k\beta_{i,e}^{k'}}{\alpha_i^2}$$

Anatomy of Chi2 function

$$\chi^2(\boldsymbol{a}) = \sum_{i,e} \left(\frac{d_{i,e} - \boxed{\sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}}{\alpha_{i,e}} \right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1 - N_e}{\delta N_e} \right)^2$$

$$T_{i,e}^{\text{eff.}}(\boldsymbol{a}) = \sum_k r_e^k \beta_{i,e}^k + T_{i,e}(\boldsymbol{a})/N_e$$

We allow additive and multiplicative distortions to the theory to match the data

Exercise 9 (time: 5 mins)

- Include the HERA datasets via `get_datasets(Q2cut=1.27**2, W2cut=10, ihera=True)`
- Print the lenght of `res` , `rres` , `nres` and interpret these numbers
- Print the summary via `residuals.gen_report`

reaction: unpol DIS

filters: Q2>1.612900

filters: W2>10.000000

reaction: unpol DIS

idx	col	obs	tar	npts	chi2	chi2/npts	rchi2	nchi2
10010	SLAC	F2	p	222.00	3446.99	15.53	0.00	2.92
10016	BCDMS	F2	p	348.00	1248.40	3.59	392.95	0.14
10020	NMC	F2	p	274.00	8859.09	32.33	1240.31	1.69
10026	HERA	sig_r	p	402.00	15581.21	38.76	6738.11	0.00
10027	HERA	sig_r	p	75.00	3681.48	49.09	970.09	0.00
10028	HERA	sig_r	p	259.00	1418.46	5.48	411.72	0.00
10029	HERA	sig_r	p	209.00	1690.17	8.09	422.45	0.00
10030	HERA	sig_r	p	159.00	542.90	3.41	266.06	0.00
10031	HERA	sig_r	p	39.00	183.09	4.69	60.78	0.00
10032	HERA	sig_r	p	42.00	42.09	1.00	6.19	0.00
10011	SLAC	F2	d	231.00	6505.54	28.16	0.00	3.37
10017	BCDMS	F2	d	254.00	2112.29	8.32	330.38	0.28
10021	NMC	F2d/F2p	d/p	174.00	1657.92	9.53	924.72	0.00

Loss function

$$\chi^2(\boldsymbol{a}) = \sum_{i,e}\left(\frac{d_{i,e} - \sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}}\right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1-N_e}{\delta N_e}\right)^2$$

Observables

$$F_i^p(x_{\mathrm{bj}}, Q^2) = \sum_q e_q^2 \int_{x_{\mathrm{bj}}}^1 \frac{d\xi}{\xi}\, f_{q/p}(\xi, \mu^2) C_{q,i}\left(\frac{x_{\mathrm{bj}}}{\xi}, \frac{Q^2}{\mu^2}, \alpha_S(\mu^2)\right) + (q \to g)$$

Reset boundary conditions

$$\frac{\partial}{\partial \ln \mu^2} f_{j/H}(\xi, \mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} P_{jj'}(z, g) f_{j'/H}(\xi/z, \mu)$$

parametrization

$$T(\xi; \boldsymbol{a}) = \mathcal{M} \frac{\xi^\alpha (1-\xi)^\beta \left(1 + \gamma\sqrt{\xi} + \delta\xi\right)}{\int_0^1 \mathrm{d}\xi\, \xi^{\alpha+1} (1-\xi)^\beta \left(1 + \gamma\sqrt{\xi} + \delta\xi\right)}$$

Managing Parameters

Setting up parameters

```
def setup_params():

    conf['params'] = {}

    conf['params']['pdf'] = {}

    conf['params']['pdf']['g N']    ={'value':   3.09994e-01, 'min': None, 'max': None, 'fixed': True }
    conf['params']['pdf']['g a']    ={'value':  -5.20900e-01, 'min': -1.9, 'max':    1, 'fixed': False}
    conf['params']['pdf']['g b']    ={'value':   4.29360e+00, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['uv N']   ={'value':   3.25322e-01, 'min': None, 'max': None, 'fixed': True }
    conf['params']['pdf']['uv a']   ={'value':  -2.14402e-01, 'min': -0.6, 'max':    1, 'fixed': False}
    conf['params']['pdf']['uv b']   ={'value':   3.04406e+00, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['dv N']   ={'value':   1.06672e-01, 'min': None, 'max': None, 'fixed': True }
    conf['params']['pdf']['dv a']   ={'value':  -3.45404e-01, 'min': -0.6, 'max':    1, 'fixed': False}
    conf['params']['pdf']['dv b']   ={'value':   4.48193e+00, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['db N']   ={'value':   3.65346e-02, 'min':    0, 'max':    1, 'fixed': False}
    conf['params']['pdf']['db a']   ={'value':  -9.35028e-01, 'min':   -1, 'max':    1, 'fixed': False}
    conf['params']['pdf']['db b']   ={'value':   4.48545e+00, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['ub N']   ={'value':   1.70043e-02, 'min':    0, 'max':    1, 'fixed': False}
    conf['params']['pdf']['ub a']   ={'value':  -1.00000e+00, 'min':   -1, 'max':    1, 'fixed': False}
    conf['params']['pdf']['ub b']   ={'value':   1.00000e+01, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['s N']    ={'value':   9.91077e-02, 'min':    0, 'max':    1, 'fixed': True}
    conf['params']['pdf']['s a']    ={'value':   1.00000e+00, 'min': -0.6, 'max':    1, 'fixed': False}
    conf['params']['pdf']['s b']    ={'value':   4.43290e+00, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['sb N']   ={'value':   2.96987e-02, 'min':    0, 'max':    1, 'fixed': False}
    conf['params']['pdf']['sb a']   ={'value':  -6.00000e-01, 'min': -0.6, 'max':    1, 'fixed': False}
    conf['params']['pdf']['sb b']   ={'value':   3.56087e+00, 'min':    0, 'max':   10, 'fixed': False}

    conf['params']['pdf']['sea N']  ={'value':   3.68792e-03, 'min':    0, 'max':    1, 'fixed': False}
    conf['params']['pdf']['sea a']  ={'value':  -1.87906e+00, 'min': -1.9, 'max':   -1, 'fixed': False}
    conf['params']['pdf']['sea b']  ={'value':   8.07746e+00, 'min':    0, 'max':   10, 'fixed': False}
```

Set limits

Define the **free** parameters

PARMAN - interface to setup parameters

Continue to Jupyter notebook

```
In [61]: class PARMAN:

    def __init__(self):
        self.get_ordered_free_params()

    def get_ordered_free_params(self):
        self.par=[]
        self.order=[]
        self.pmin=[]
        self.pmax=[]

        if 'check lims' not in conf: conf['check lims']=True

        for k in conf['params']:
            for kk in conf['params'][k]:
                if  conf['params'][k][kk]['fixed']==False:
                    p=conf['params'][k][kk]['value']
                    pmin=conf['params'][k][kk]['min']
                    pmax=conf['params'][k][kk]['max']
                    self.pmin.append(pmin)
                    self.pmax.append(pmax)
                    if p<pmin or p>pmax:
                        if conf['check lims']: raise ValueError(
```

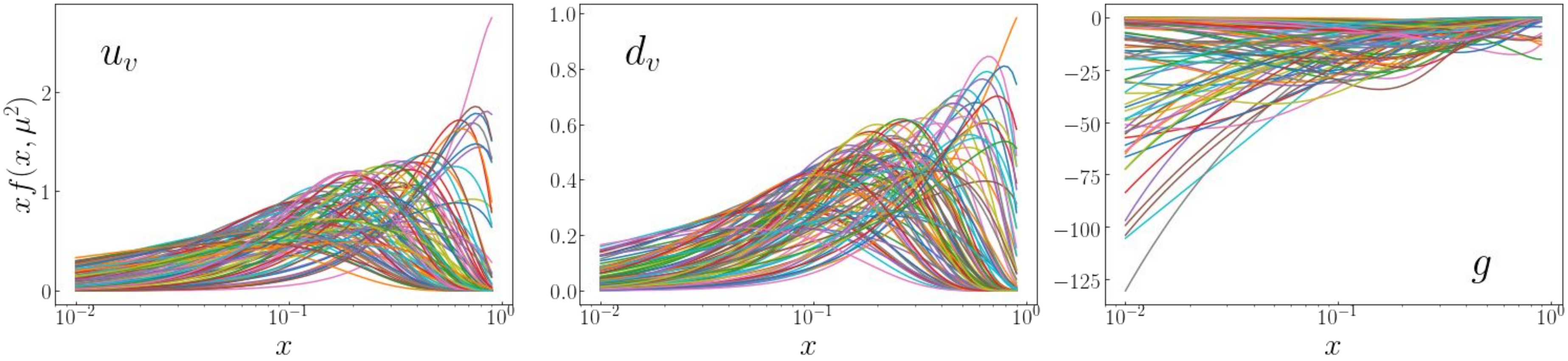
Check the limits

Updates the PDF class

Handle internally parameter ordering

Exercise 10 (time: 5 mins)

- Use the method `gen_flat` to scan 100 random parameter vectors within the hyperbox and plot the resulting pdf conbinations $xu_v,\ xd_v,\ xg$ at the input scale $mu^2 = 1.27 * *2$.
- **Hint**: use the x-range `X = 10**np.linspace(-2,np.log10(0.9),100)`



Managing Residuals

Loss function

$$\chi^2(\boldsymbol{a}) = \sum_{i,e}\left(\frac{d_{i,e} - \sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}}\right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1-N_e}{\delta N_e}\right)^2$$

Observables

$$F_i^p(x_{\mathrm{bj}},Q^2) = \sum_q e_q^2 \int_{x_{\mathrm{bj}}}^1 \frac{d\xi}{\xi}\, f_{q/p}(\xi,\mu^2)\, C_{q,i}\left(\frac{x_{\mathrm{bj}}}{\xi},\frac{Q^2}{\mu^2},\alpha_S(\mu^2)\right) + (q\to g)$$

Reset boundary conditions

$$\frac{\partial}{\partial \ln\mu^2} f_{j/H}(\xi,\mu) = \sum_{j'}\int_\xi^1 \frac{dz}{z} P_{jj'}(z,g)\, f_{j'/H}(\xi/z,\mu)$$

parametrization

$$T(\xi;\boldsymbol{a}) = \mathcal{M}\frac{\xi^\alpha(1-\xi)^\beta\left(1+\gamma\sqrt{\xi}+\delta\xi\right)}{\int_0^1 \mathrm{d}\xi\,\xi^{\alpha+1}(1-\xi)^\beta\left(1+\gamma\sqrt{\xi}+\delta\xi\right)}$$

RESMAN - interface to query residuals

```
In [75]: class RESMAN:

             def __init__(self):

                 conf['aux']      = AUX()
                 conf['mellin'] = MELLIN(npts=4)
                 conf['alphaS'] = ALPHAS()
                 conf['eweak']  = EWEAK()
                 conf['pdf']      = PDF()

                 self.parman=PARMAN()
                 self.setup_idis()


             def setup_idis(self):

                 self.idis_tabs=READER().load_data_sets('idis')
                 conf['idis tabs'] = self.idis_tabs
                 self.idis_thy=THEORY()
                 conf['idis'] = self.idis_thy
                 self.idis_res=RESIDUALS()

             def get_residuals(self,par, initial = False):

                 self.parman.set_new_params(par, initial = initial)

                 data = conf['datasets']

                 #--compute residuals
                 res,rres,nres=[],[],[]
                 if 'idis' in conf['datasets']:
                     out=self.idis_res.get_residuals()
                     res=np.append(res,out[0])
                     rres=np.append(rres,out[1])
                     nres=np.append(nres,out[2])
                 return res,rres,nres
```

Loads all the parts that we need, including parman

Collect all the residuals

$$\chi^2(\boldsymbol{a}) = \sum_{i,e}\left(\frac{d_{i,e} - \sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}}\right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1-N_e}{\delta N_e}\right)^2$$

Continue to Jupyter notebook

Exercise 11 (time: 5 mins)

- Use `RESMAN` to compute 10 times the residuals for different parameters generated by `parman.gen_flat()`
- For each run print the first 3 entries of the current parameters as well a the total χ^2

```
par= [0.19197712 1.03448414 0.34214713]  chi2= 13419609.489305
par= [-0.20511792  6.35590864  0.64983866]  chi2= 81859552.43154643
par= [-1.1957957   7.33717188 -0.43530517]  chi2= 76402443.43800555
par= [ 0.3481756   3.25387064 -0.54117908]  chi2= 18069861.530045442
par= [0.91721661 2.12831572 0.24350342]  chi2= 24218875.047675647
par= [-1.33526148  5.43868399  0.00556536]  chi2= 17724081.19952482
par= [-0.4772018   0.96316372  0.33593136]  chi2= 60638297.4801399
par= [-1.66225909  7.95801622  0.26728382]  chi2= 51754451.24467237
par= [0.45288714 6.74492207 0.99792619]  chi2= 12098310.192077495
par= [-1.74669965  5.36573854 -0.11400083]  chi2= 27450829.072440065
```

Loss function

$$\chi^2(\boldsymbol{a}) = \sum_{i,e} \left(\frac{d_{i,e} - \sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}}\right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1-N_e}{\delta N_e}\right)^2$$

⇧

Observables

$$F_i^p(x_{\mathrm{bj}},Q^2) = \sum_q e_q^2 \int_{x_{\mathrm{bj}}}^1 \frac{d\xi}{\xi}\, f_{q/p}(\xi,\mu^2)\, C_{q,i}\left(\frac{x_{\mathrm{bj}}}{\xi},\frac{Q^2}{\mu^2},\alpha_S(\mu^2)\right) + (q\to g)$$

⇧

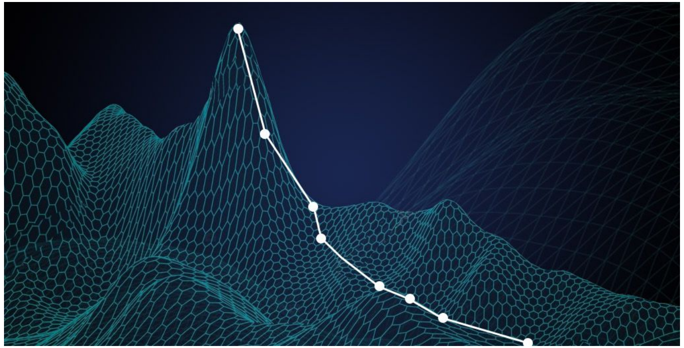
Reset boundary conditions

$$\frac{\partial}{\partial \ln\mu^2} f_{j/H}(\xi,\mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} P_{jj'}(z,g)\, f_{j'/H}(\xi/z,\mu)$$

⇧

parametrization

$$T(\xi;\boldsymbol{a}) = \mathcal{M}\frac{\xi^\alpha(1-\xi)^\beta\left(1+\gamma\sqrt{\xi}+\delta\xi\right)}{\int_0^1 \mathrm{d}\xi\, \xi^{\alpha+1}(1-\xi)^\beta\left(1+\gamma\sqrt{\xi}+\delta\xi\right)}$$



Maximum Likelihood

MAXLIKE - interface of maximum likelihood

```
In [83]: class MAXLIKE:

             def __init__(self):

                 self.resman=RESMAN()
                 self.parman=self.resman.parman
                 self.set_display()


             def get_residuals(self,par):
                 res,rres,nres=self.resman.get_residuals(par)
                 self.cnt+=1
                 self.print_status(res,rres,nres)
                 if len(rres)!=0: res=np.append(res,rres)
                 if len(nres)!=0: res=np.append(res,nres)
                 return res

             def checklimits(self):

                 for k in conf['params']:
                     for kk in conf['params'][k]:
                         if conf['params'][k][kk]['fixed']!=False:
                         p=conf['params'][k][kk]['value']
                         pmin=conf['params'][k][kk]['min']
                         pmax=conf['params'][k][kk]['max']
                         if  p<pmin or p>pmax:
                             print ('%s-%s out of limits. '%(k,kk))
                             sys.exit()

                 for k in conf['datasets']:
                     for kk in conf['datasets'][k]['norm']:
                         p=conf['datasets'][k]['norm'][kk]['value']
```

Loads resman

SciPy.org

SciPy.org | Docs | SciPy v1.6.0 Reference Guide | Optimization and root finding (scipy.optimize)

scipy.optimize.least_squares¶

scipy.optimize.**least_squares**(*fun, x0, jac='2-point', bounds=- inf, inf, method='trf', ftol=1e-08, xtol=1e-08, gtol=1e-08, x_scale=1.0, loss='linear', f_scale=1.0, diff_step=None, tr_solver=None, tr_options={}, jac_sparsity=None, max_nfev=None, verbose=0, args=(), kwargs={}*) [source]

Solve a nonlinear least-squares problem with bounds on the variables.

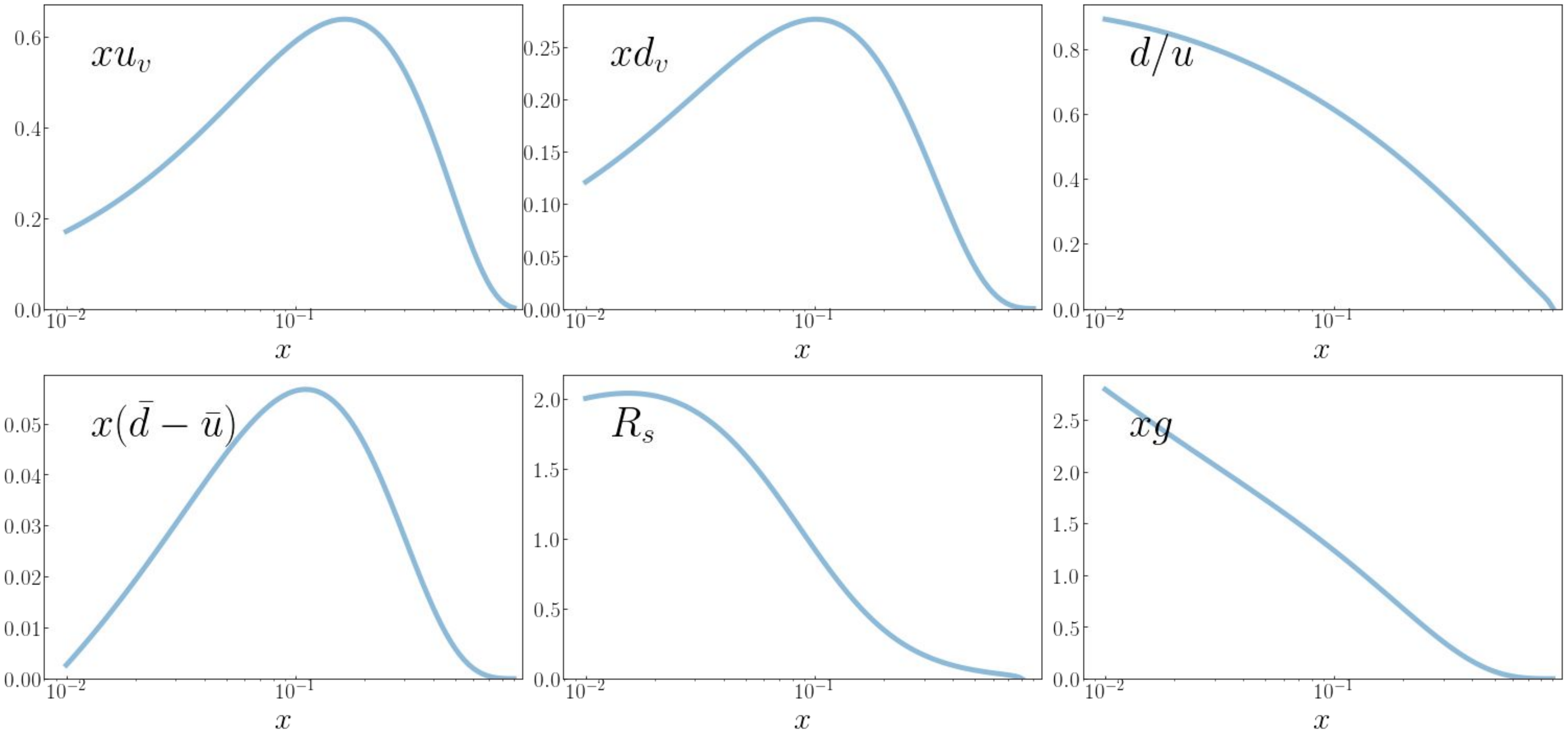
Given the residuals f(x) (an m-D real function of n real variables) and the loss function rho(s) (a sca

least_squares finds a local minimum of the cost function F(x):

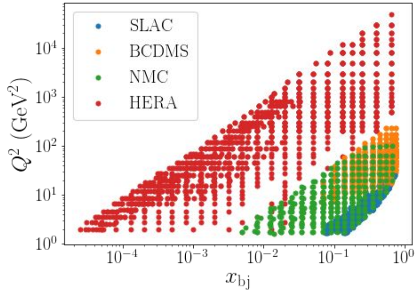
Continue to Jupyter notebook

Exercise 12.A (time: 5 mins)

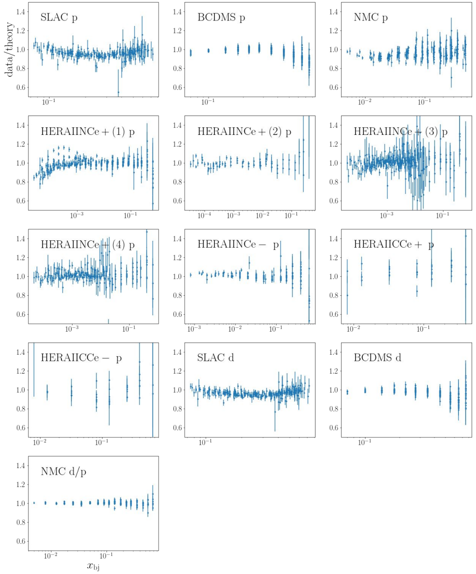
- Plot xu_v, xd_v, d/u, $x(\bar{d}+\bar{u})$,$x(\bar{d}-\bar{u})$, $R_s = (\bar{d}+\bar{u})/(s+\bar{s})$, xg
- **hint**: use the method `conf['pdf'].get_xf(x,mu2,flav)`

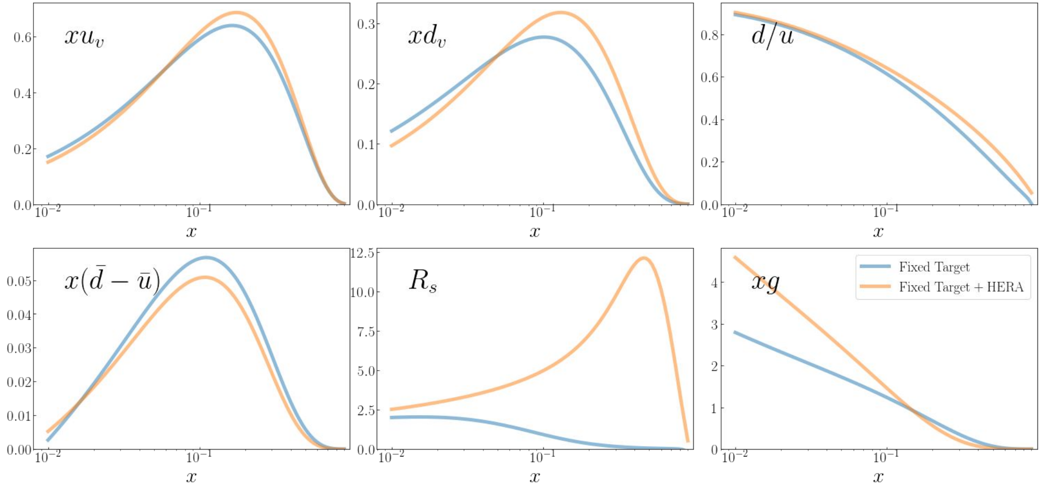


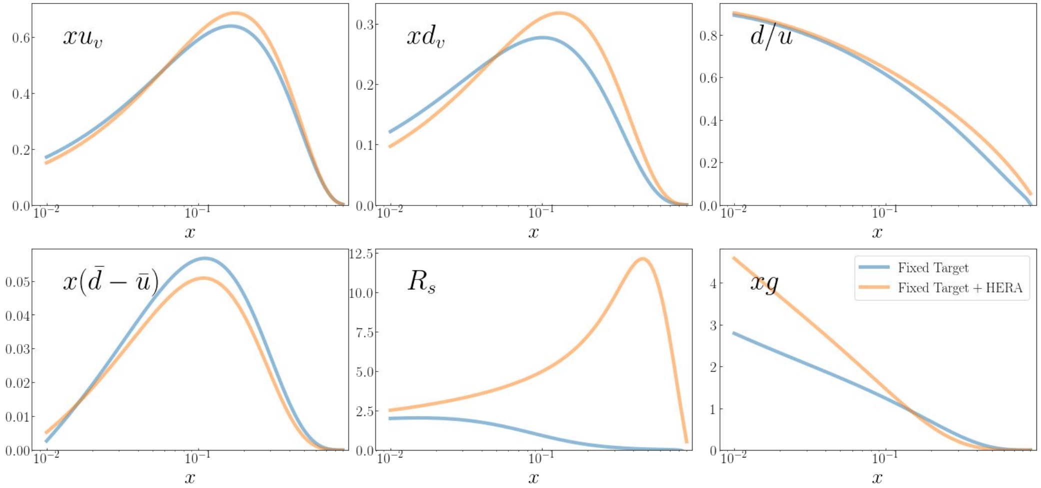
Exercise 12.B (time: 10 mins)



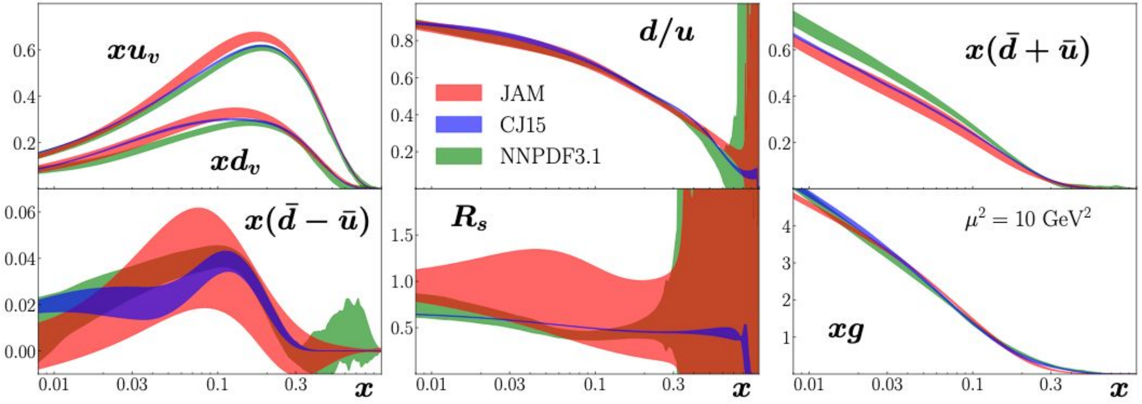
- Include the HERA data sets and run the fitter
- Plot data/theory for all experiments
- Plot the pdfs (as above) from the previous run and the new run
- **hint:**
  - use `get_datasets(Q2cut=1.27**2, W2cut=10, ihera=True)`
  - assuming you have the two sets of parameters stored as par1 and par2, when making the pdfs plots, use the method `resman.parman.set_new_params(par)` to update the `PDF` class







Moffat, Melnitchouk, Rogers, NS
arXiv:2101.04664



Outline

Lecture 1
- Motivations
- QCD carpentry setup
- Solving QCD's beta function

Lecture 2
- Mellin transforms
- Solving DGLAP
- Modeling input scale PDFs

Lecture 3
- DIS theory
- World DIS data
- The chi2 function
- Global analysis

Lecture 4
- Bayesian inference
  - Maximum likelihood
  - MC methods
- JAM history
- Machine learning

Bayesian inference

The Bayes theorem

$$\mathcal{L}(\boldsymbol{a}, \mathrm{data}) = \exp\left[-\frac{1}{2}\chi^2(\boldsymbol{a}, \mathrm{data})\right]$$

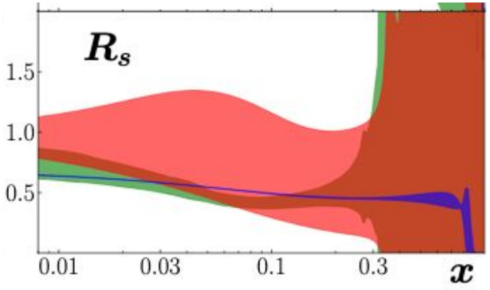
This is a choice

Min, Max, penalties, regulators etc

$$\rho(\boldsymbol{a}|\mathrm{data}) \sim \mathcal{L}(\boldsymbol{a}, \mathrm{data})\pi(\boldsymbol{a})$$

$$\Downarrow$$

This is impractical



$$\mathrm{E}[\mathcal{O}] = \int d^n a \; \rho(\boldsymbol{a}|\mathrm{data}) \; \mathcal{O}(\boldsymbol{a})$$

$$\mathrm{V}[\mathcal{O}] = \int d^n a \; \rho(\boldsymbol{a}|\mathrm{data}) \; \left[\mathcal{O}(\boldsymbol{a}) - \mathrm{E}[\mathcal{O}]\right]^2$$

How do we deal with the curse of dimensionality ?

$$\mathrm{E}[\mathcal{O}] = \int d^n a \; \rho(\boldsymbol{a}|\mathrm{data}) \; \mathcal{O}(\boldsymbol{a})$$

$$\mathrm{V}[\mathcal{O}] = \int d^n a \; \rho(\boldsymbol{a}|\mathrm{data}) \; \left[\mathcal{O}(\boldsymbol{a}) - \mathrm{E}[\mathcal{O}]\right]^2$$

Option 1: Maximum likelihood

Asummes symmetric likelihood, unique solution

$$\mathrm{E}[\mathcal{O}] \simeq \mathcal{O}(\boldsymbol{a}_0)$$

Assumes Gaussian behavior around ML

$$\mathrm{V}[\mathcal{O}] = \mathrm{Hessian,\ \ Lagrange\ multipliers}$$

How do we deal with the curse of dimensionality ?

$$\mathrm{E}[\mathcal{O}] = \int d^n a \; \rho(\boldsymbol{a}|\mathrm{data}) \; \mathcal{O}(\boldsymbol{a})$$

$$\mathrm{V}[\mathcal{O}] = \int d^n a \; \rho(\boldsymbol{a}|\mathrm{data}) \; \left[\mathcal{O}(\boldsymbol{a}) - \mathrm{E}[\mathcal{O}]\right]^2$$

Option 2: MC approach

Build an MC ensemble ($$$)

$$\mathrm{E}[\mathcal{O}] \simeq \frac{1}{N}\sum_k \mathcal{O}(\boldsymbol{a}_k)$$

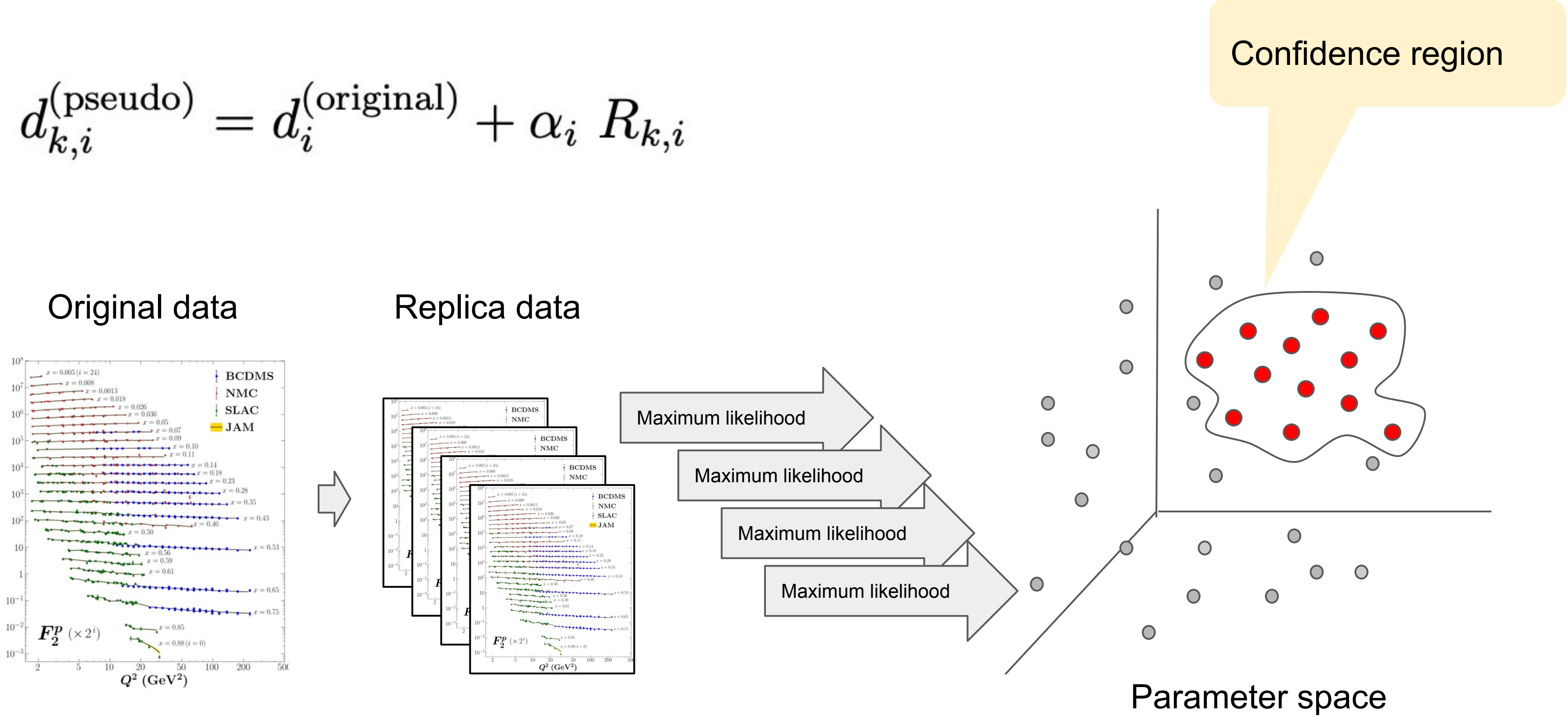
$$\mathrm{V}[\mathcal{O}] =\simeq \frac{1}{N}\sum_k \left[\mathcal{O}(\boldsymbol{a}_k) - \mathrm{E}[\mathcal{O}]\right]^2$$

Many algorithms

- MCMC
- HMC
- Data resampling
- ...

Data resampling

$$d_{k,i}^{(\mathrm{pseudo})} = d_i^{(\mathrm{original})} + \alpha_i \, R_{k,i}$$



Jefferson Lab Angular Momentum Collaboration

Staff / Faculty

W. Melnitchouk (JLab), T. Rogers (ODU/JLab), A. Prokudin (PSU), D. Pitonyak (LVC), L. Gamberg (PSU), Z. Kang (UCLA) J. Qiu (JLab), A. Accardi (Hampton/JLab), A. Metz (Temple), C.-R. Ji (NCSU), M. Constantinou (Temple), F. Steffens (Bonn), M. White (Adelaide) , ...

Students / Postdocs

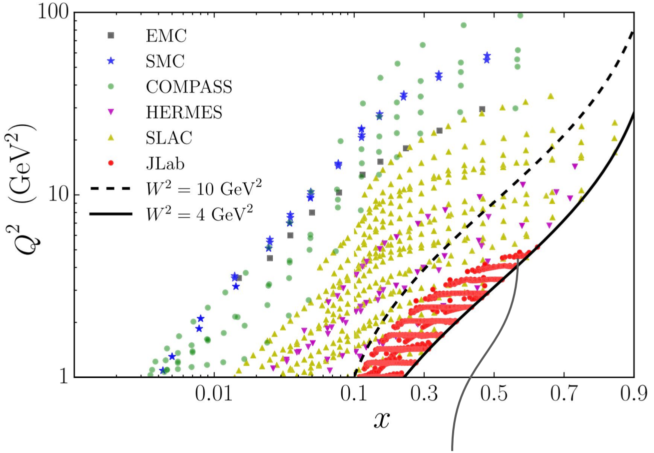
C. Cocuzza (Temple), Y. Zhou (W&M), P. Barry (NCSU), E. Moffat (ODU), J. Bringewatt (UMD), J. Ethier (Nikhef), C. Andres (JLab), F. Delcarro (JLab), A. Hiller-Blin (JLab), Z. Searle (Adelaide)

JAM history

JAM'15 (1D spin-PDFs)

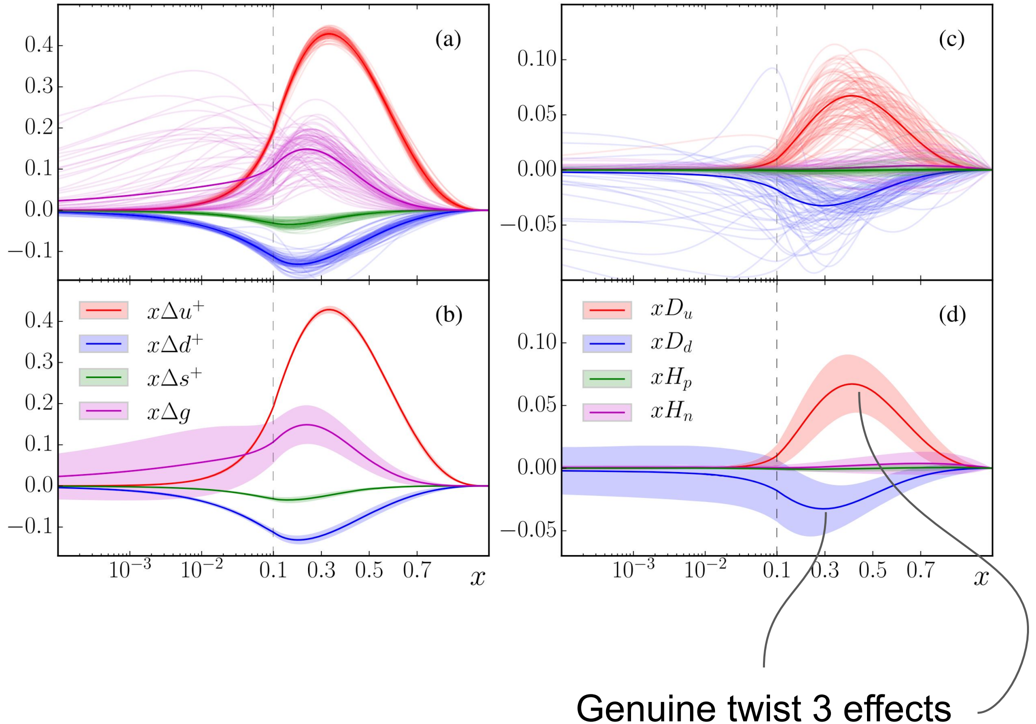
Bayesian MC framework

$$A_\parallel = \frac{\sigma^{\downarrow\Uparrow} - \sigma^{\uparrow\Uparrow}}{\sigma^{\downarrow\Uparrow} + \sigma^{\uparrow\Uparrow}} \qquad A_\perp = \frac{\sigma^{\downarrow\Rightarrow} - \sigma^{\uparrow\Rightarrow}}{\sigma^{\downarrow\Rightarrow} + \sigma^{\uparrow\Rightarrow}}$$



High-x data

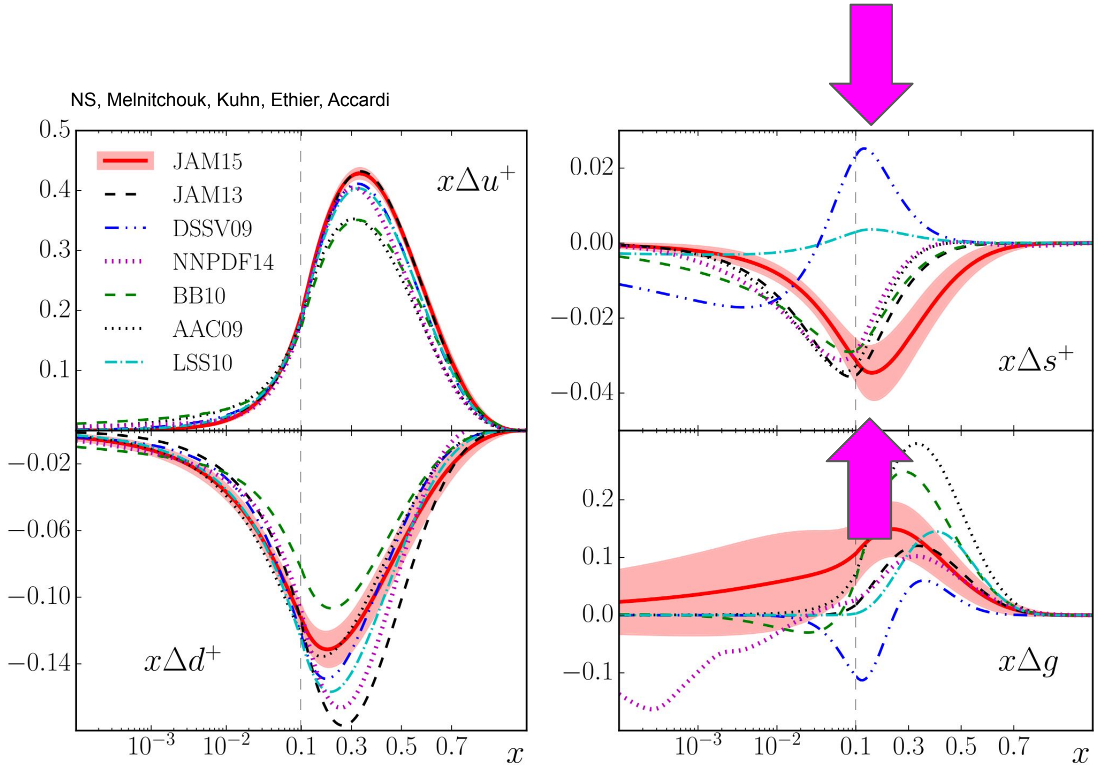
NS, Melnitchouk, Kuhn, Ethier, Accardi



Genuine twist 3 effects

JAM'15 (1D spin-PDFs)

NS, Melnitchouk, Kuhn, Ethier, Accardi



A Possible Resolution of the Strange Quark Polarization Puzzle ?

Elliot Leader, Alexander V. Sidorov, Dimiter B. Stamenov

The strange quark polarization puzzle, i.e. the contradiction between the negative polarized strange quark density obtained from analyses of inclusive DIS data and the positive values obtained from combined analyses of inclusive and semi-inclusive SIDIS data using de Florian et. al. (DSS) fragmentation functions, is discussed. To this end the results of a new combined NLO QCD analysis of the polarized inclusive and semi-inclusive DIS data, using the Hirai et. al. (HKNS) fragmentation functions, are presented. It is demonstrated that the polarized strange quark density is very sensitive to the kaon fragmentation functions, and if the set of HKNS fragmentation functions is used, the polarized strange quark density obtained from the combined analysis turns out to be negative and well consistent with values obtained from the pure DIS analyses.

"...It is demonstrated that the polarized strange quark density is very sensitive to Kaon FF."

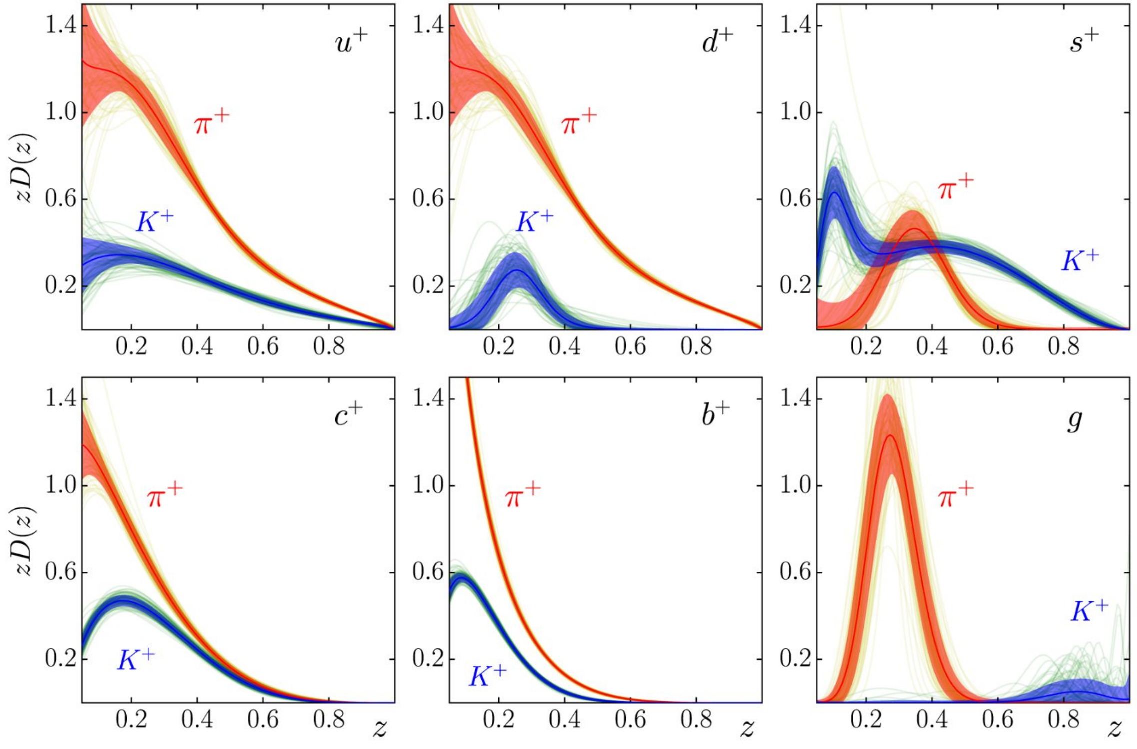
SU(3) constraints:

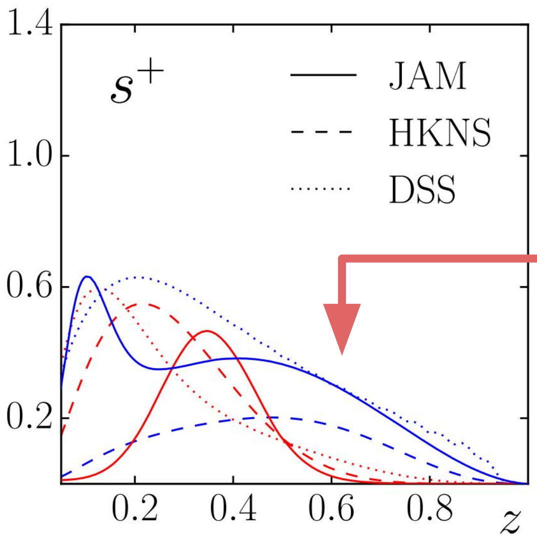
$$\Delta u^+(1,Q^2) + \Delta d^+(1,Q^2) - 2\Delta s^+(1,Q^2) = a_8,$$

Role of SIDIS and SIA ?

JAM'16 (1D FFs)

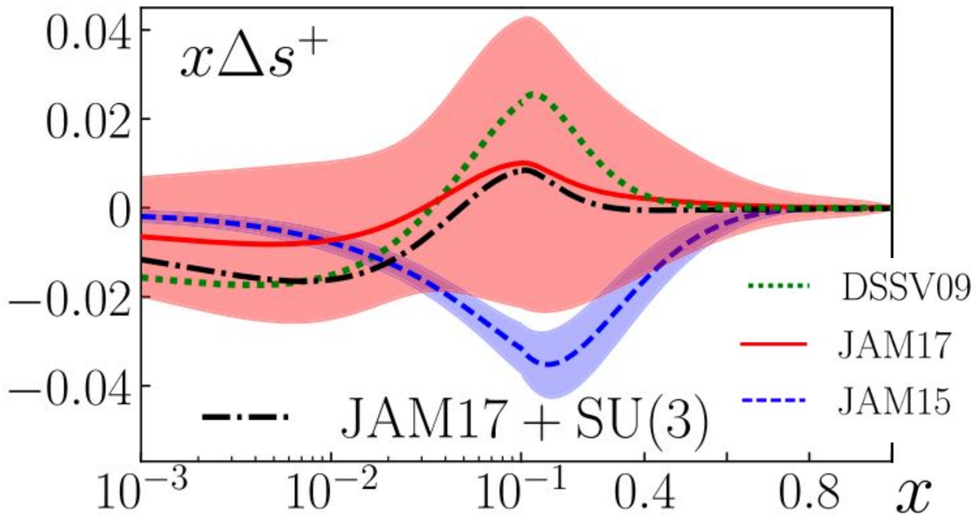
NS, Ethier, Melnitchouk, Hirai, Kumano



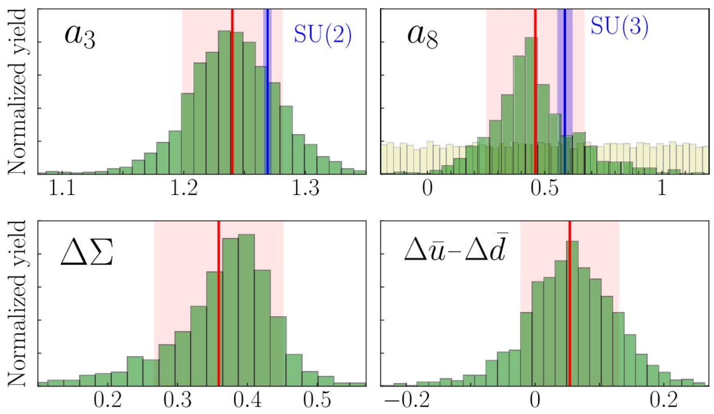


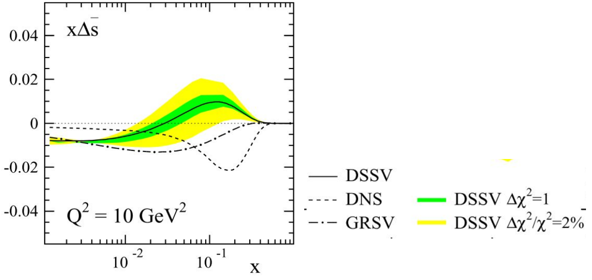
FF kaon:
JAM closer to DSS at large z

JAM’17 (1D simultaneous extraction of spin PDFs and FFs)



Ethier, NS, Melnitchouk



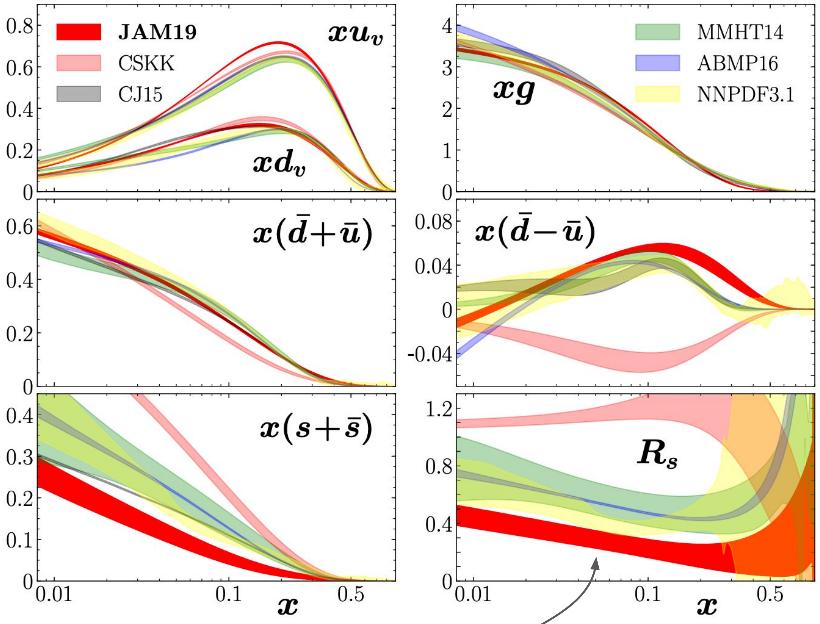


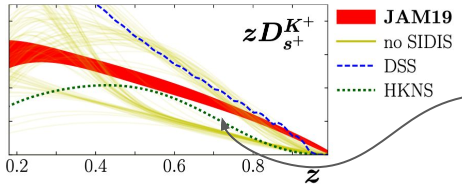
Florian, Sassot, Stratmann, Vogelsang

- Use of pol. DIS, SIDIS and SIA
- No SU(2) nor SU(3) constraints
- Empirical evidence of g_3 ~ g_A 2%
- **No strange puzzle - need more data**

JAM'19 (1D simul. extraction of spin-averaged PDFs and FFs)

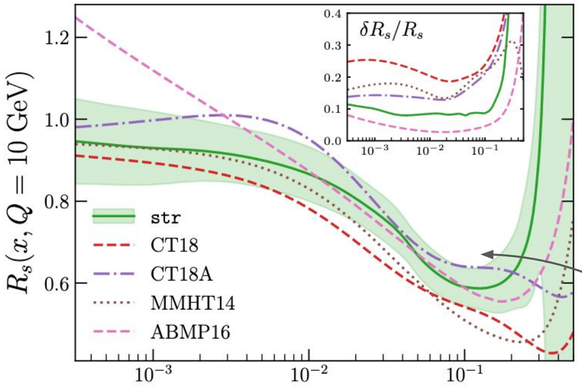
NS, Andres, Melnitchouk





SIA (kaon) prefers large strange -> Kaon FF

Faura, Iranipour, Nocera, Rojo, Ubiali

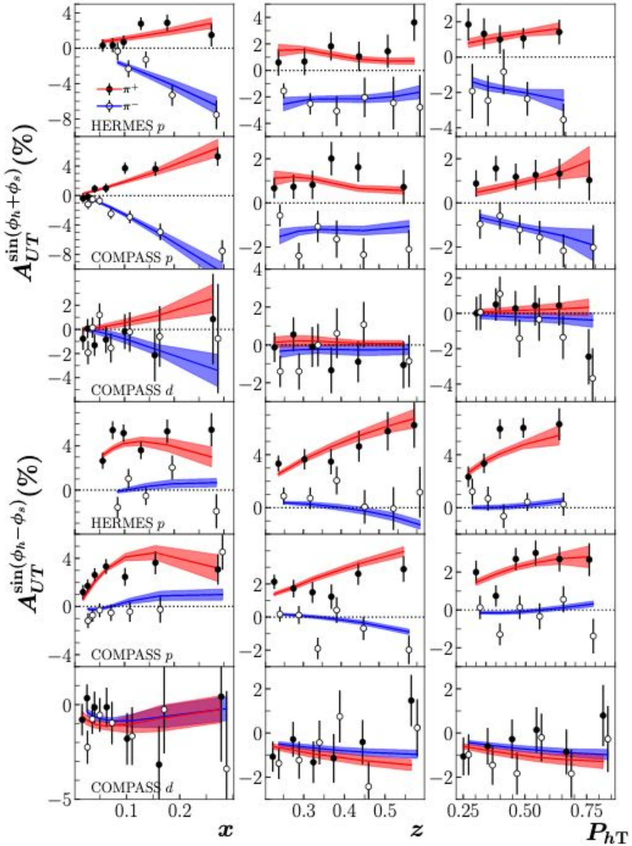


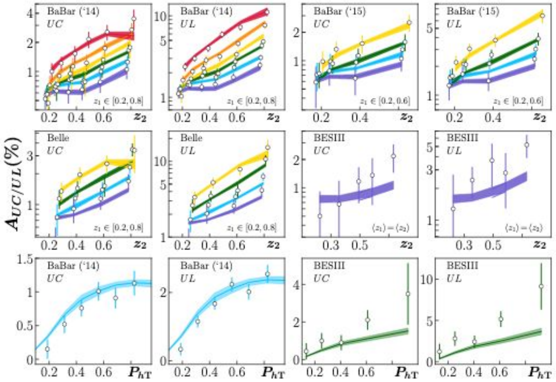
SIDIS (kaon) suppresses strange PDF

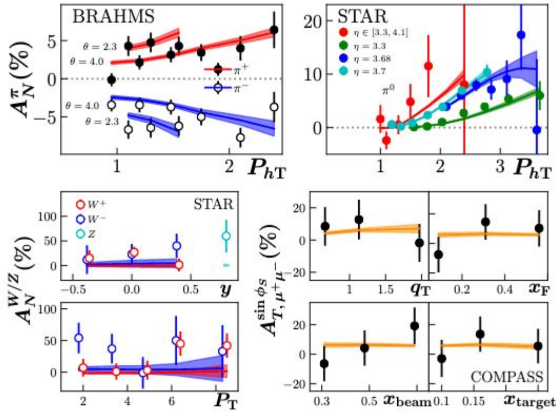
Strange suppression disfavored by collider W+c data

JAM’20 (3D zoo of correlation functions)

Cammarota, Gamberg, Kang, Miller, Pitonyak, Prokudin, Rogers, NS

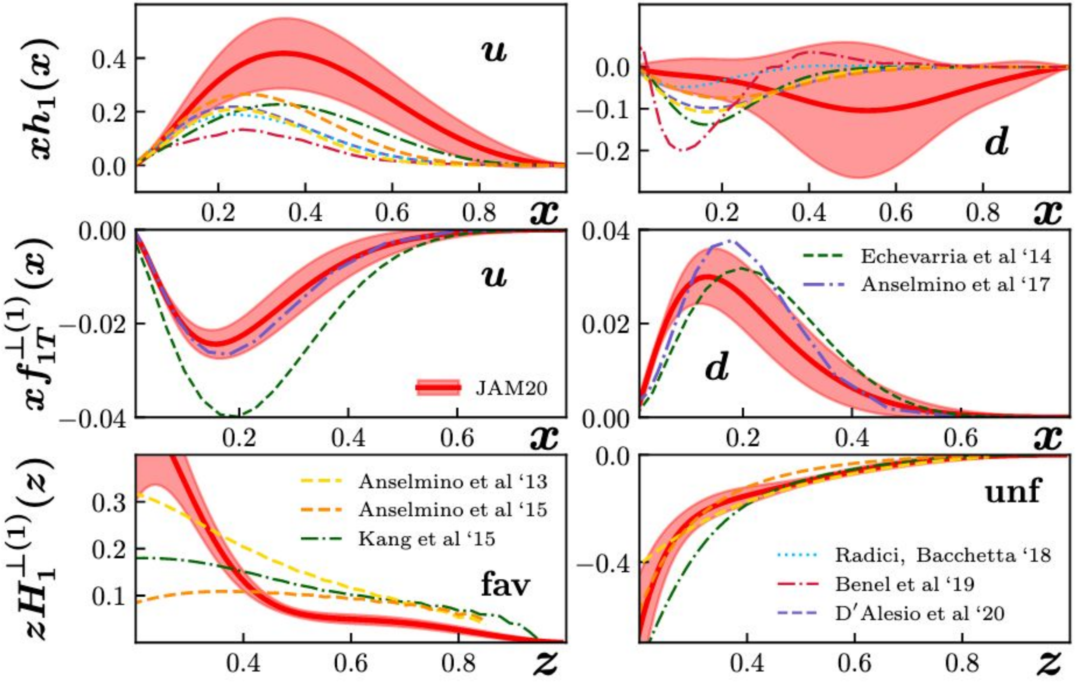




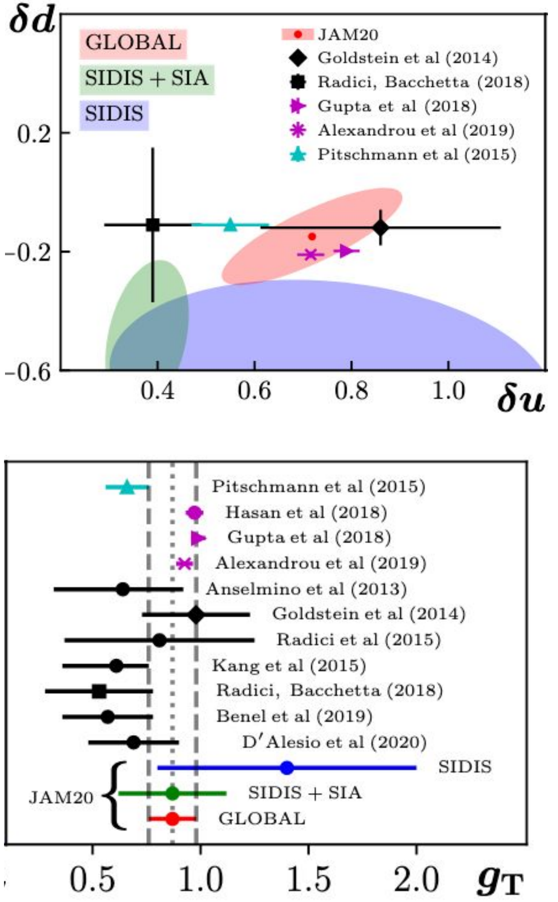


Observable	Reactions
$A_{\text{SIDIS}}^{\text{Siv}}$	$e+(p,d)^{\uparrow} \to e+(\pi^+,\pi^-,\pi^0)+X$
$A_{\text{SIDIS}}^{\text{Col}}$	$e+(p,d)^{\uparrow} \to e+(\pi^+,\pi^-,\pi^0)+X$
$A_{\text{SIA}}^{\text{Col}}$	$e^+ + e^- \to \pi^+\pi^-(UC,UL)+X$
$A_{\text{DY}}^{\text{Siv}}$	$\pi^- + p^{\uparrow} \to \mu^+\mu^- + X$
$A_{\text{DY}}^{\text{Siv}}$	$p^{\uparrow}+p \to (W^+,W^-,Z)+X$
A_N^h	$p^{\uparrow}+p \to (\pi^+,\pi^-,\pi^0)+X$

JAM'20 (3D zoo of correlation functions)

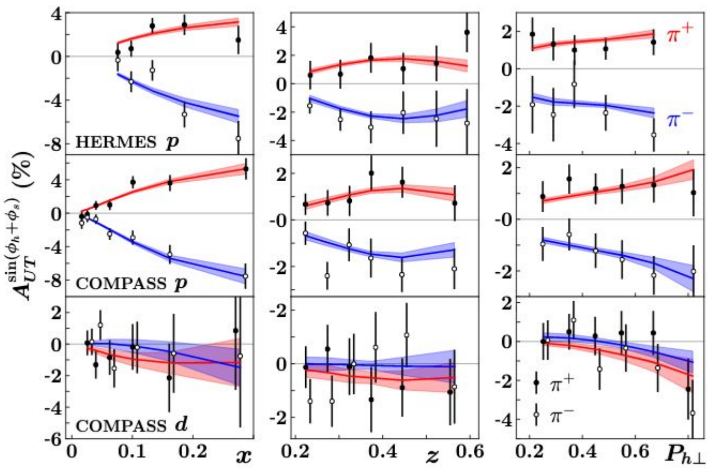


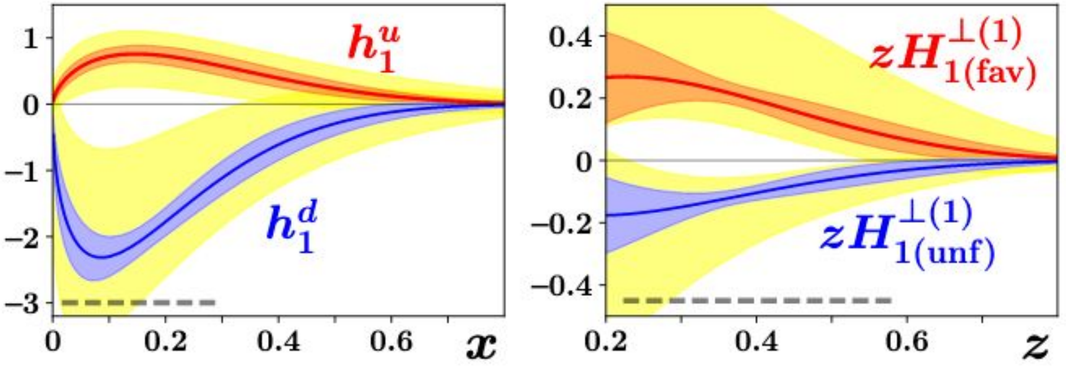
Cammarota, Gamberg, Kang, Miller, Pitonyak, Prokudin, Rogers, NS

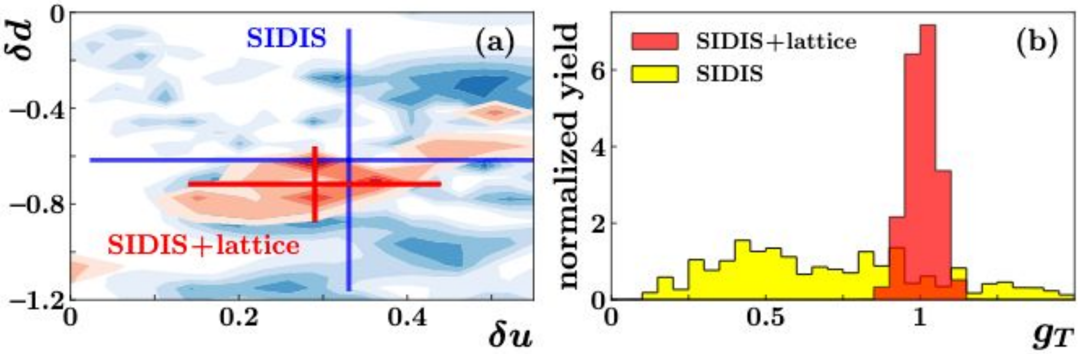


JAM'18 (3D experiment + lattice QCD: gT moment)

Lin, Melnitchouk, Prokudin, NS, Shows



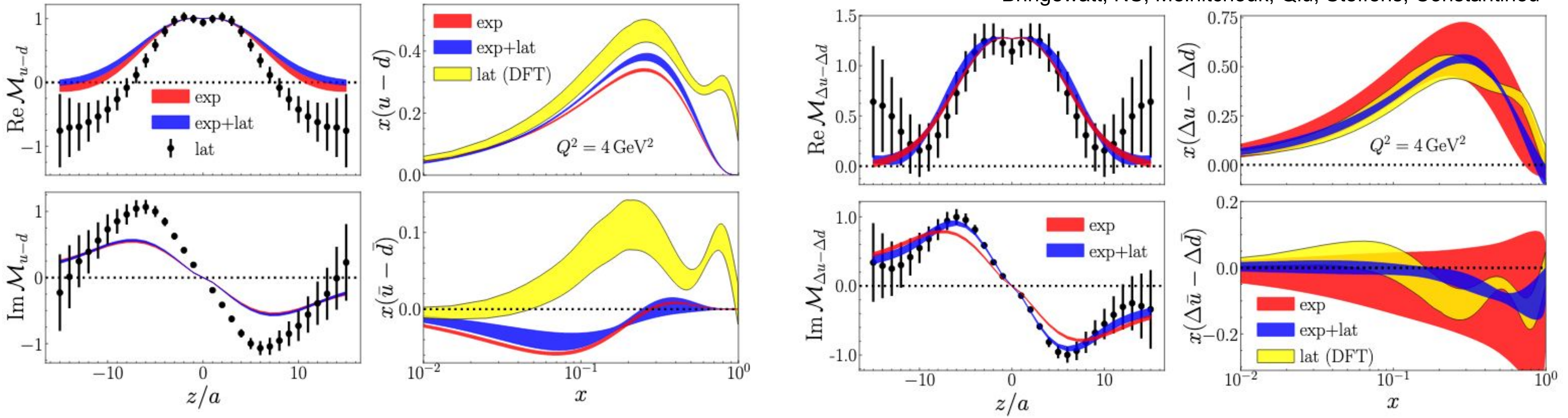




Inclusion of gT as Bayesian prior can complement experimental data

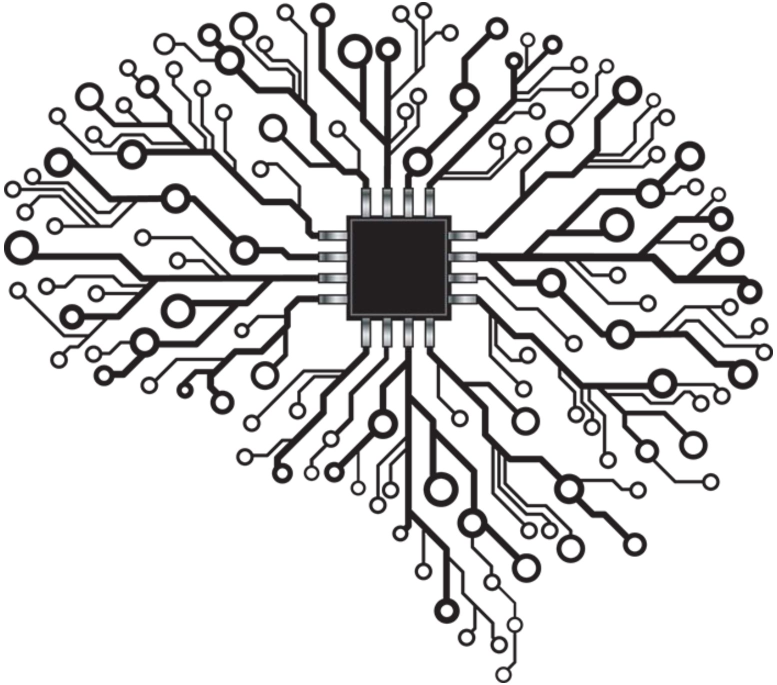
JAM'20 (1D experiment + lattice QCD: quasi-PDFs)

Bringewatt, NS, Melnitchouk, Qiu, Steffens, Constantinou



$$\mathcal{M}_q(z,\mu) = \int_{-\infty}^{\infty} dx\, e^{-ixP_3 z} \int_{-1}^{1} \frac{d\xi}{|\xi|} C_q\left(\frac{x}{\xi}, \frac{\mu}{\xi P_3}\right) f_q(\xi,\mu)$$

$$\mathcal{M}_{\Delta q}(z,\mu) = \int_{-\infty}^{\infty} dx\, e^{-ixP_3 z} \int_{-1}^{1} \frac{d\xi}{|\xi|} C_{\Delta q}\left(\frac{x}{\xi}, \frac{\mu}{\xi P_3}\right) \Delta f_q(\xi,\mu)$$



Machine Learning

The inverse problem

Parametrization d.o.f.

Forward mapping

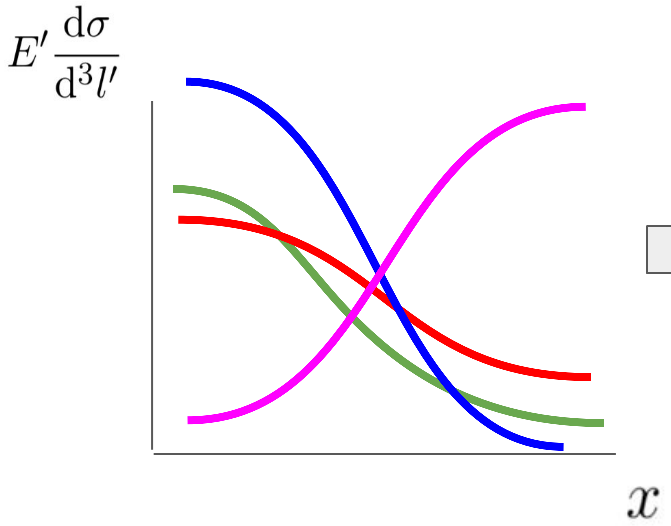
$$E' \frac{\mathrm{d}\sigma^{\mathbf{DIS}}}{\mathrm{d}^3 l'} = \sum_i \int_x^1 \frac{\mathrm{d}\xi}{\xi} H_i(\xi) f_i\left(\frac{x}{\xi}; \boldsymbol{a}\right)$$

Experimental observable

Backward mapping

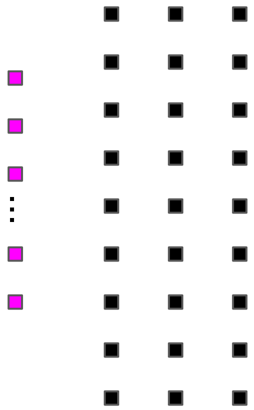
An idea: parametrize the **inverse function**

Observable space



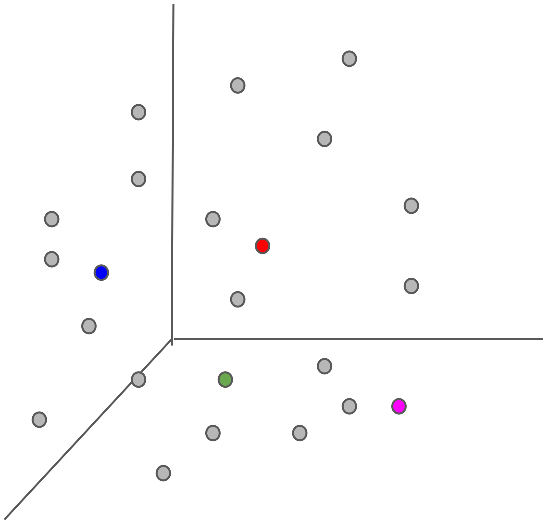
Theory

$$E' \frac{\mathrm{d}\sigma}{\mathrm{d}^3 l'} = \sum_i \int_x^1 \frac{\mathrm{d}\xi}{\xi} H_i(\xi) f_i \left(\frac{x}{\xi} ; \boldsymbol{a} \right)$$



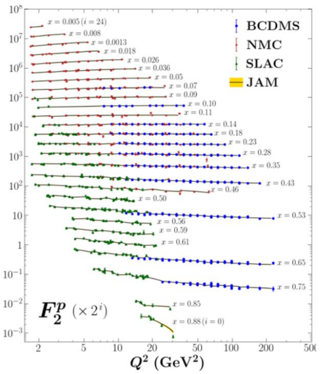
Neural Nets

Parameter space

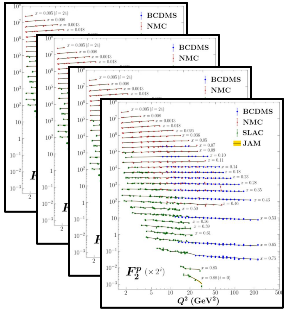


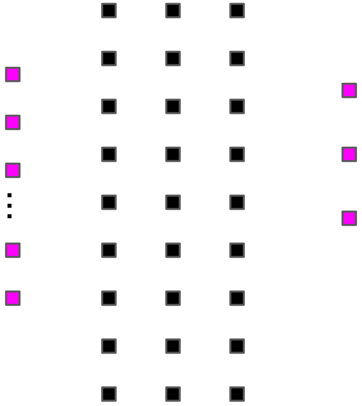
Parameter inference

Original data

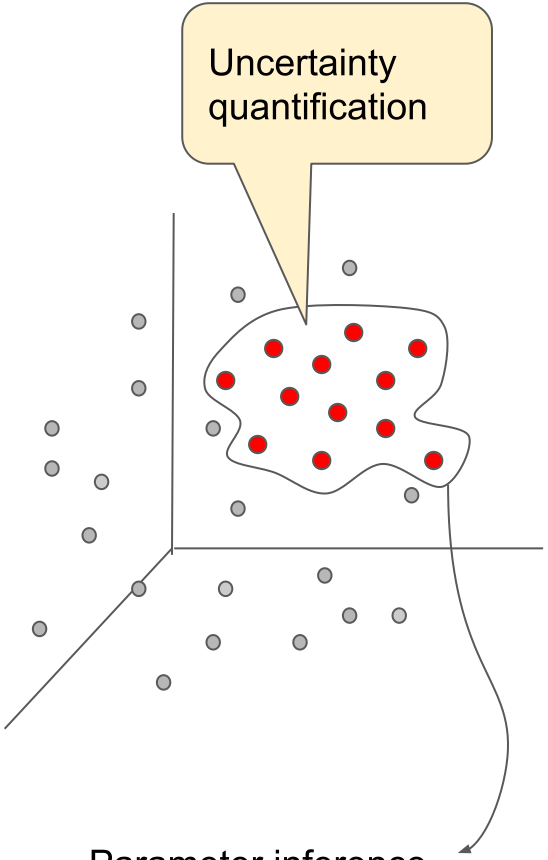


Replica data



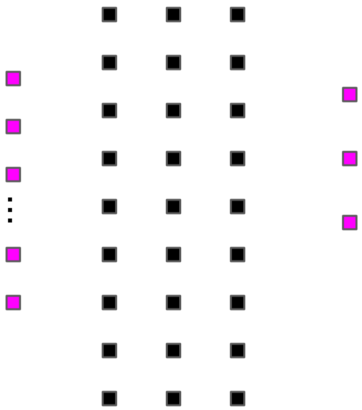


Trained inverse mapper



Parameter inference

$$\rho(\mathbf{a}|\text{data}) \sim \mathcal{L}(\mathbf{a}, \text{data})\pi(\mathbf{a})$$

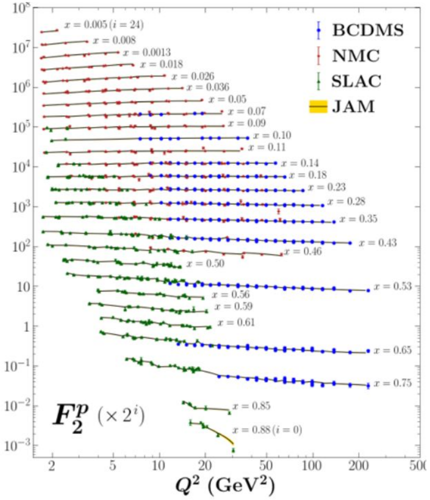


So why do we need **inverse mappers**?

1) Manipulate data input

Collecting MC samples is too expensive

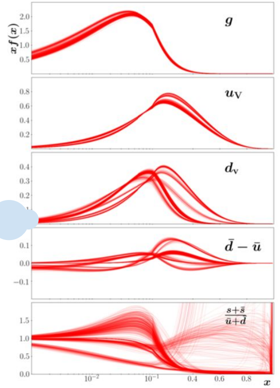
$$\rho(\mathbf{a}|\mathrm{data}) \sim \mathcal{L}(\mathbf{a}, \mathrm{data})\pi(\mathbf{a})$$



What happens if we remove … data ?

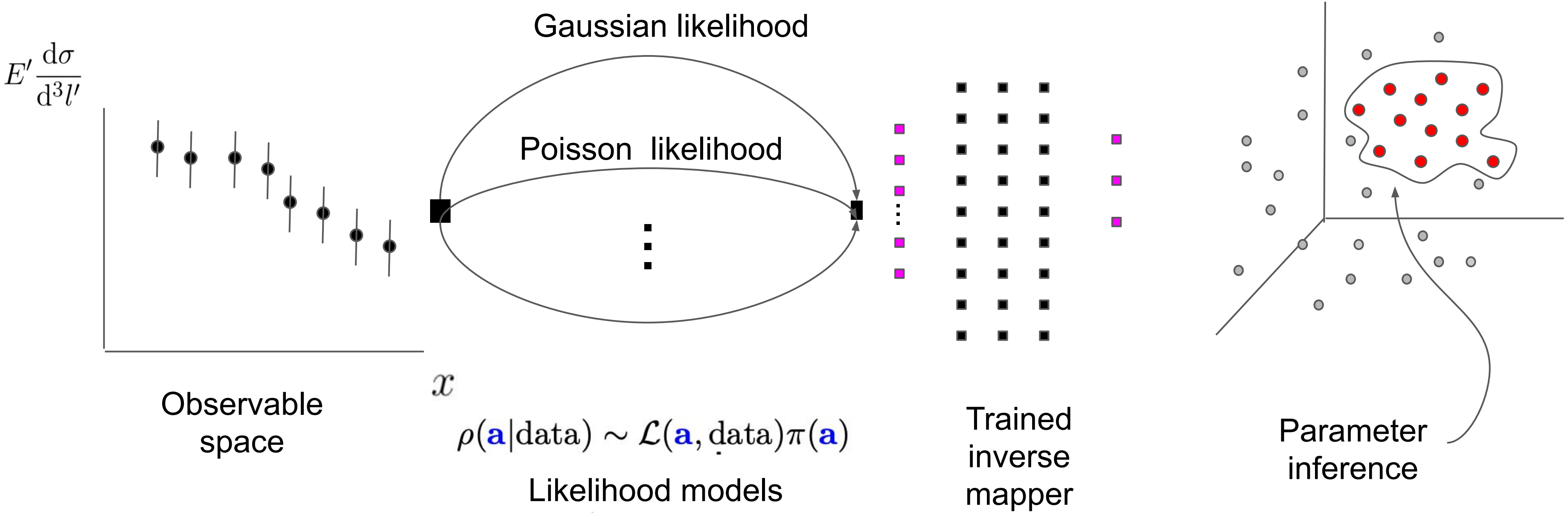
Where do we need more experiments?

“Global analysis is a kind of a sausage” … **how to unpack it**?

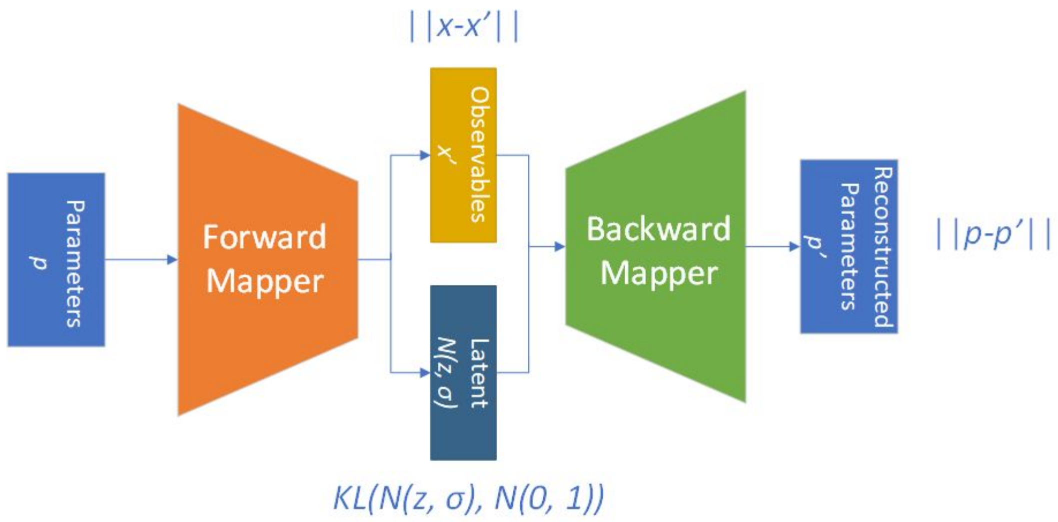


What data are forcing … to be …?

2) Bayesian inference modeling



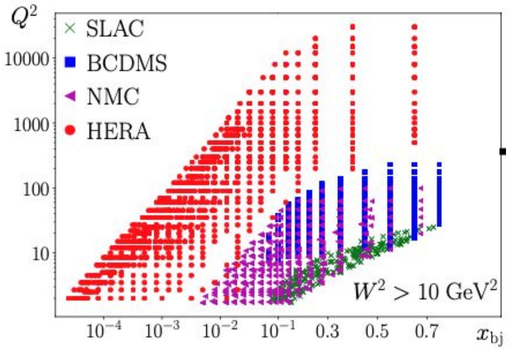
Existing methodologies are prohibitively expensive for such studies

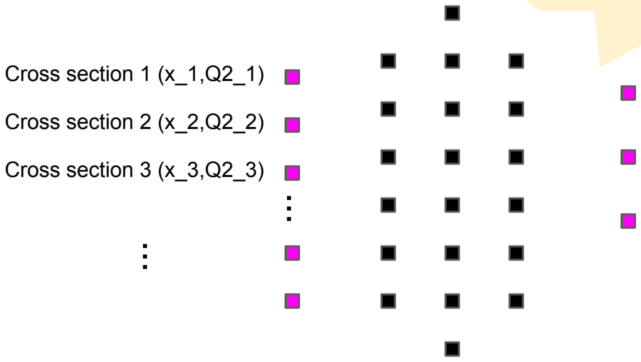


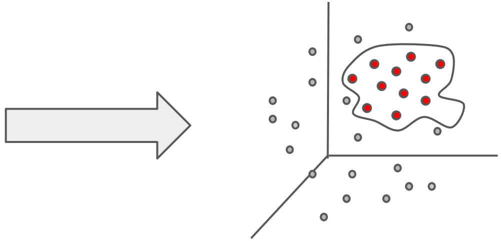
Inverse mapper architectures

Designing the inverse mappers

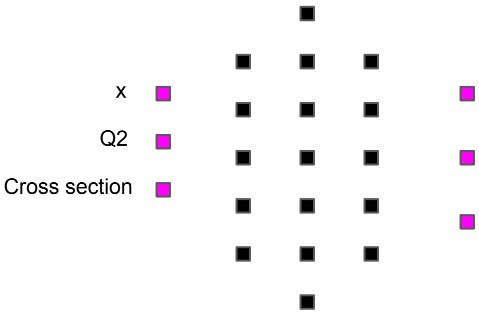
Kinematics dependent inverse mapper







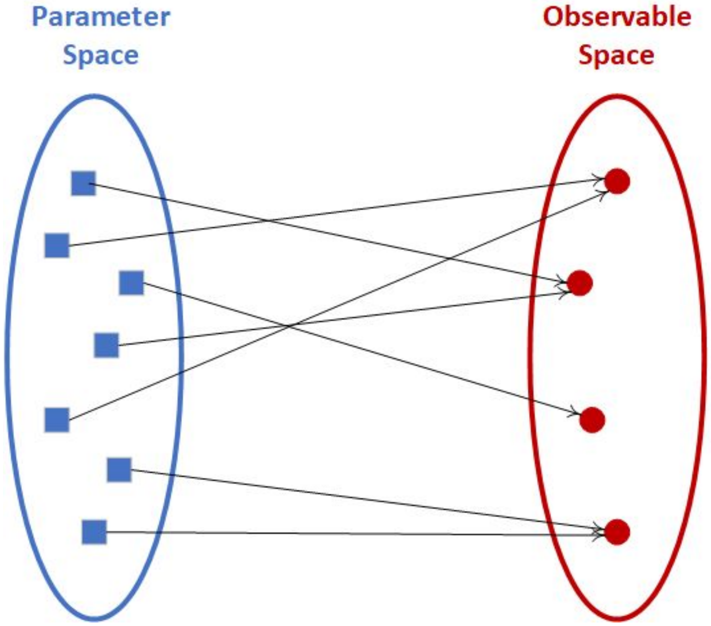
Parameter inference



Kinematics-independent inverse mapper

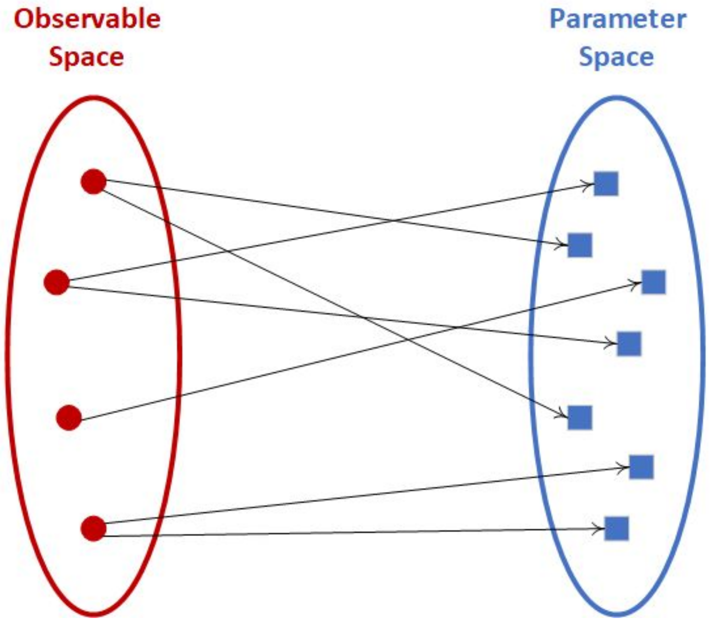
Ambiguity in inverse problems

Forward Mapper

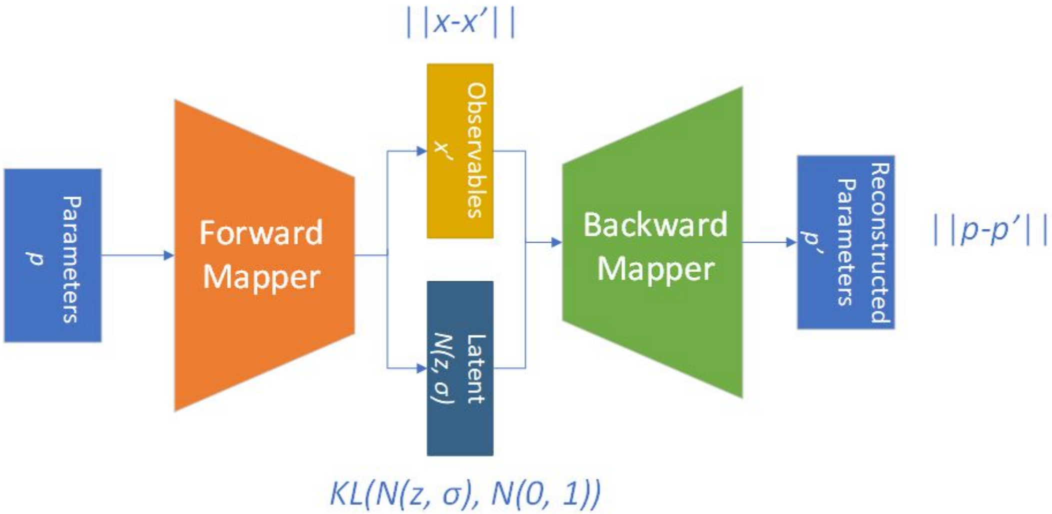


Backward Mapper

Ambiguous



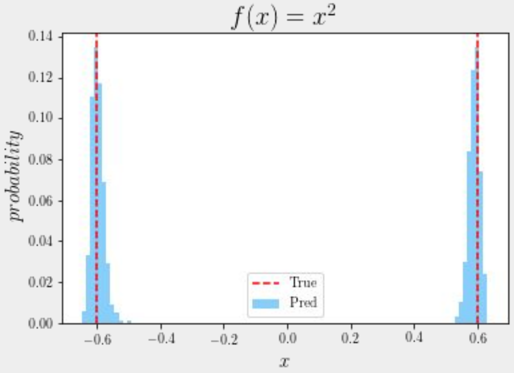
Kinematic-independent inverse mapper:
Variational Autoencoder (VAE)



- Better than previous models
- Remove the grid dependence
- Highly accurate
- No Gaussian mixture assumption

Toy problems with multiple solutions

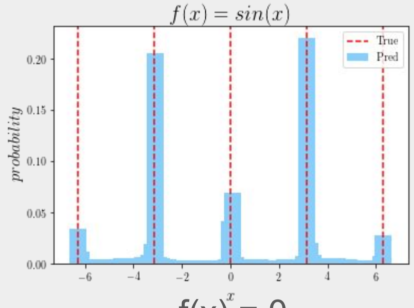
$f(x) = x^2$



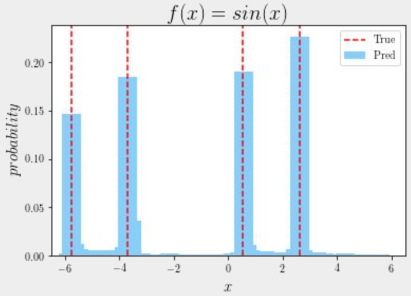
$f(x) = 0.36$

Two Solutions

$f(x) = \sin(x)$



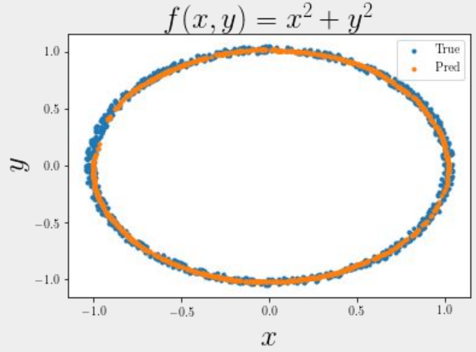
$f(x) = 0$



$f(x) = 0.5$

Multiple Finite Solutions

$f(x, y) = x^2+y^2$

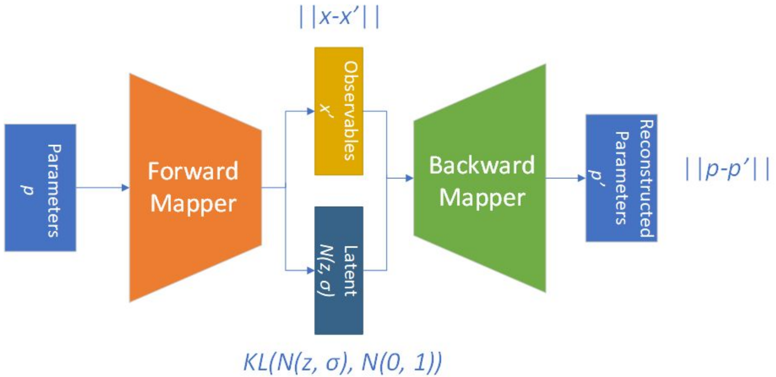


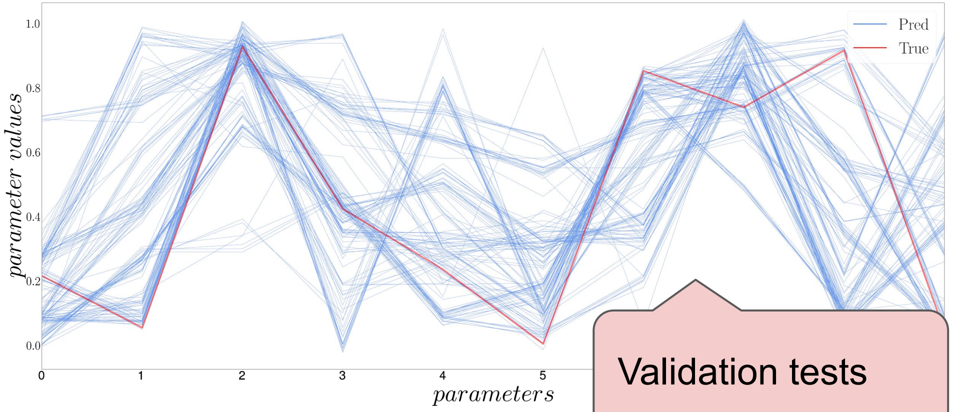
$f(x, y) = 1.0$

Infinite Solutions

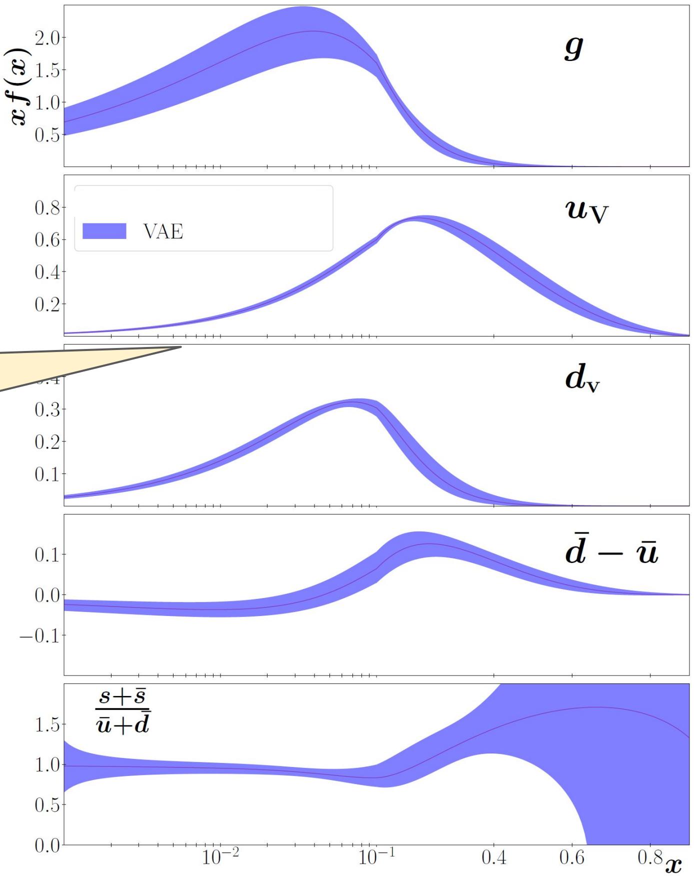
Does it work for DIS?

M. Almaeen *et al.* (in preparation, 2020)





Validation tests

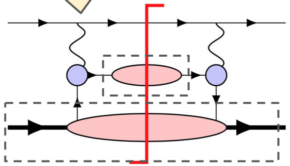


Inverse mapper in action!

The workforce of FemtoAnalyzer

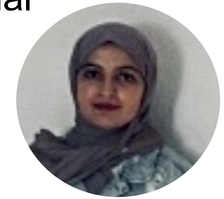
Jefferson Lab

QCD Theory

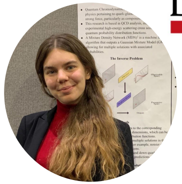


ODU

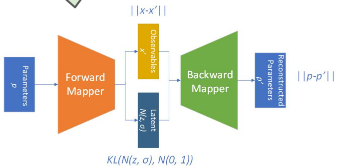
ODU
Manal



Eleni
DAVIDSON



Inverse mappers

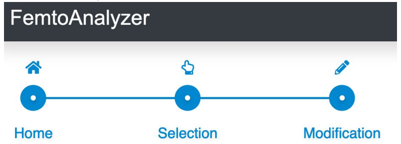


DAVIDSON

Rida
DAVIDSON

ODU
Heramb

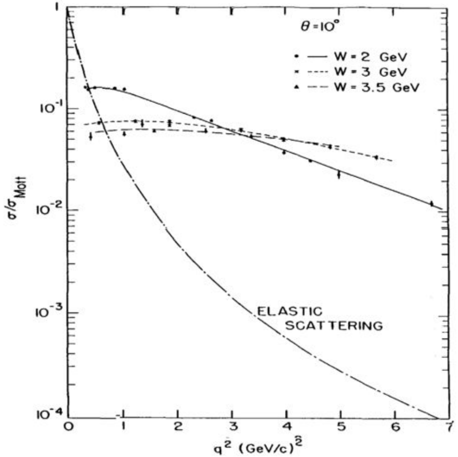
Web-interface

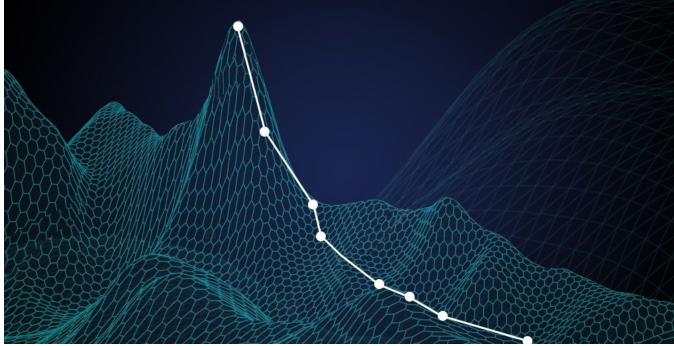


Raghu
DAVIDSON

Annabel
DAVIDSON

Summary





Loss function

$$\chi^2(\boldsymbol{a}) = \sum_{i,e}\left(\frac{d_{i,e} - \sum_k r_e^k \beta_{i,e}^k - T_{i,e}(\boldsymbol{a})/N_e}{\alpha_{i,e}}\right)^2 + \sum_k \left(r_e^k\right)^2 + \left(\frac{1-N_e}{\delta N_e}\right)^2$$

⇧

Observables

$$F_i^p(x_{\mathrm{bj}}, Q^2) = \sum_q e_q^2 \int_{x_{\mathrm{bj}}}^1 \frac{d\xi}{\xi}\, f_{q/p}(\xi,\mu^2) C_{q,i}\left(\frac{x_{\mathrm{bj}}}{\xi}, \frac{Q^2}{\mu^2}, \alpha_S(\mu^2)\right) + (q \to g)$$

⇧

Reset boundary conditions

$$\frac{\partial}{\partial \ln \mu^2} f_{j/H}(\xi,\mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} P_{jj'}(z,g) f_{j'/H}(\xi/z,\mu)$$

⇧

parametrization

$$T(\xi;\boldsymbol{a}) = \mathcal{M}\frac{\xi^\alpha (1-\xi)^\beta \left(1+\gamma\sqrt{\xi}+\delta\xi\right)}{\int_0^1 \mathrm{d}\xi\, \xi^{\alpha+1}(1-\xi)^\beta \left(1+\gamma\sqrt{\xi}+\delta\xi\right)}$$