QCD carpentry: 1D structure of the nucleon

Nobuo Sato
ODU/JLab

CFNS summer school
Stony Brook, 2019

Outline

- **Part I:** Basics of DIS
- **Part II:** Elementary treatment of DIS Factorization
- **Part III:** Solving RGE - coding
- **Part IV:** DIS phenomenology - coding

Outline

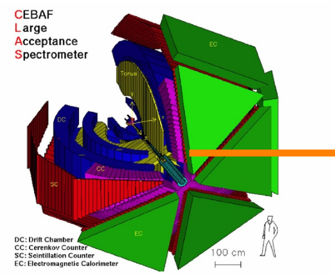
- **Part I: Basics of DIS**
- **Part II: Elementary treatment of DIS Factorization**
- **Part III: Solving RGE - coding**
- **Part IV: DIS phenomenology - coding**

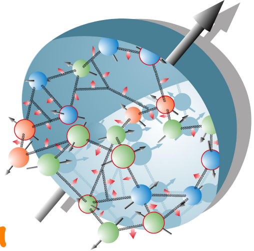
References

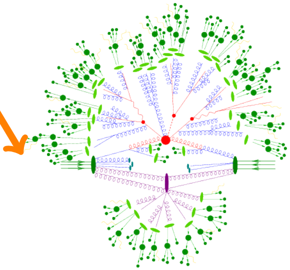
- Collins "Foundations of perturbative QCD"
- Moffat, Melnitchouk, Rogers, NS (PRD95, 2017)
- Vogt (hep-ph/0408244)

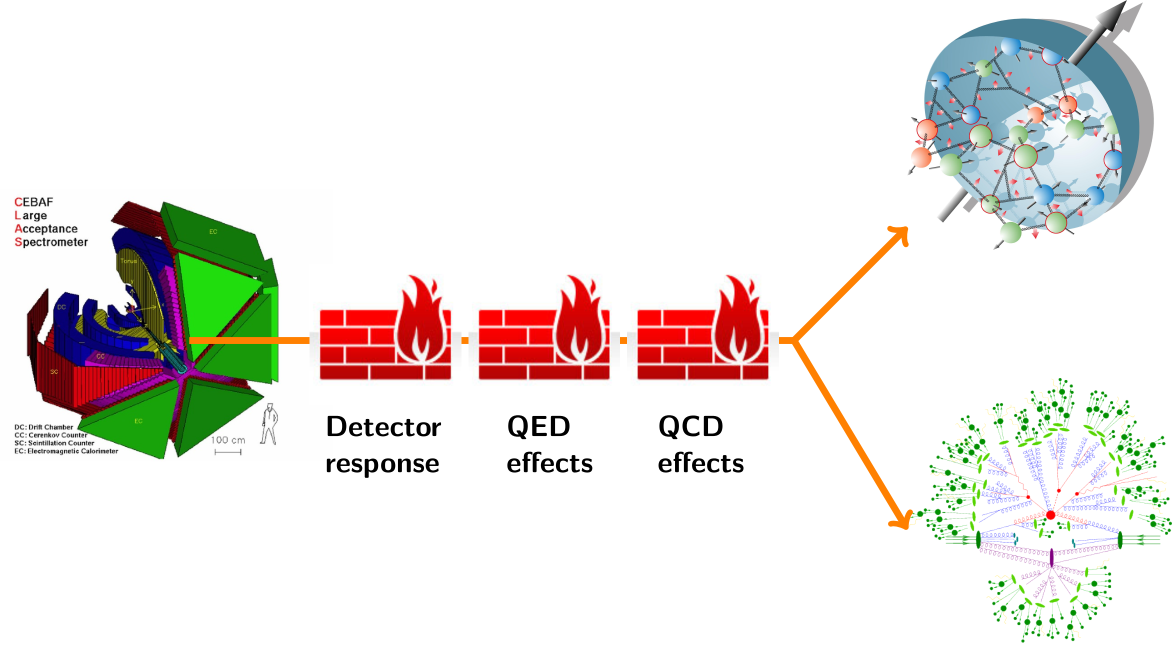
Part I: Basics of DIS

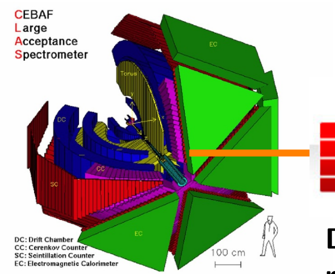
- The experimental observables
- Kinematic variables
- Cross sections and structure functions
- Overview of experimental data


CEBAF
Large
Acceptance
Spectrometer
EC
Torus
DC
CC
SC
EC
DC: Drift Chamber
CC: Cerenkov Counter
SC: Scintillation Counter
EC: Electromagnetic Calorimeter
100 cm





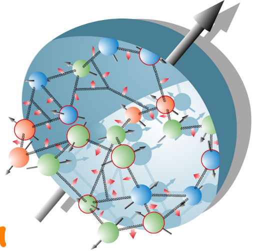

CEBAF
Large
Acceptance
Spectrometer
DC: Drift Chamber
CC: Cerenkov Counter
SC: Scintillation Counter
EC: Electromagnetic Calorimeter
100 cm
Detector response
QED effects
QCD effects

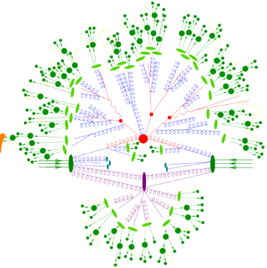


Detector response

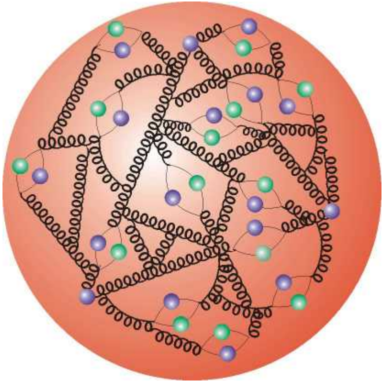
QED effects

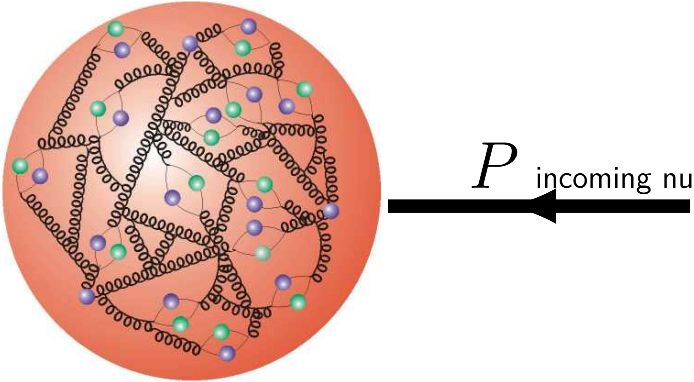
QCD effects

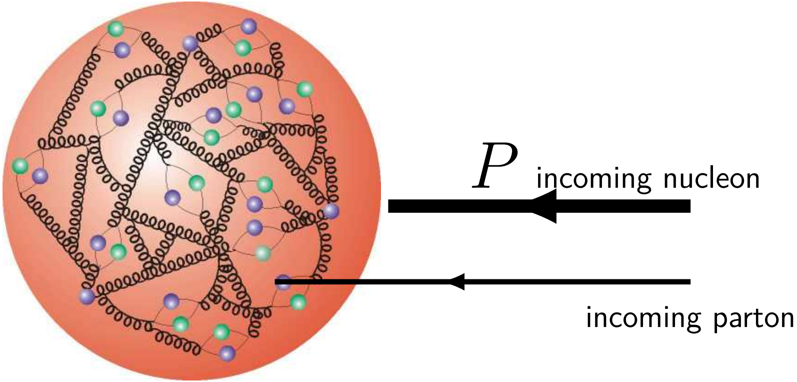




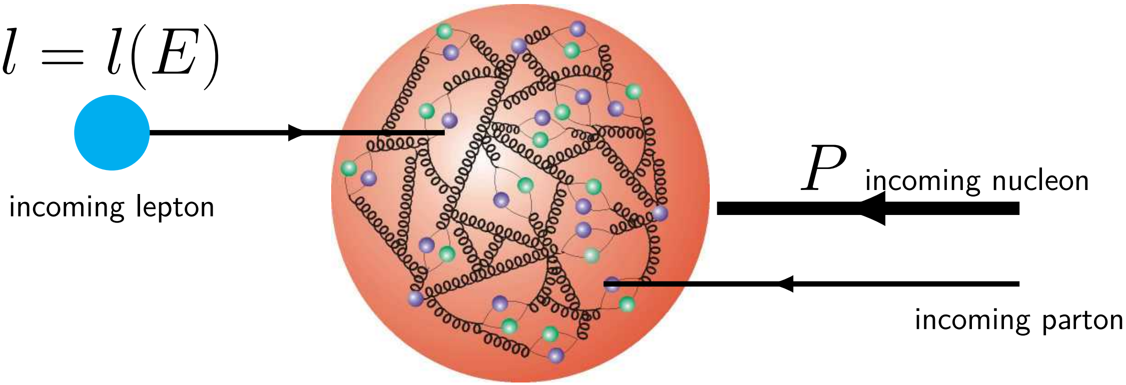
$$\sigma^{\mathrm{EXP}} = w^{\mathrm{DR}} \otimes w^{\mathrm{QED}} \otimes \sigma^{\mathrm{QCD}}$$

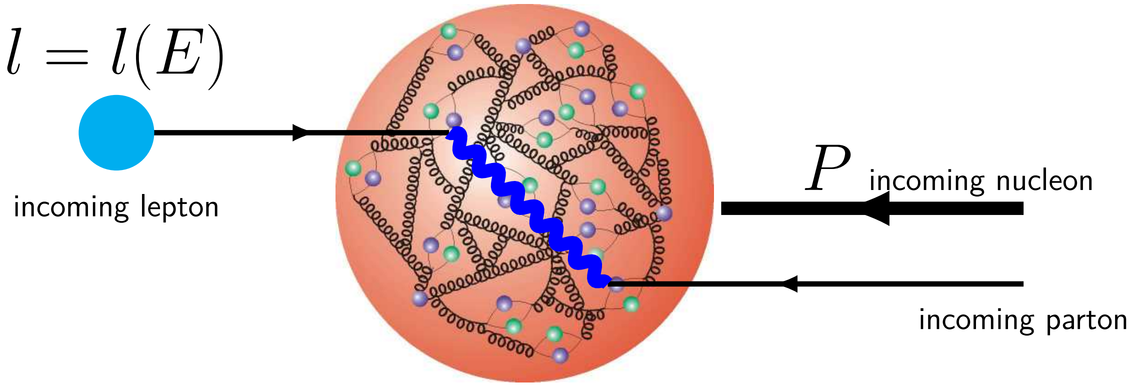


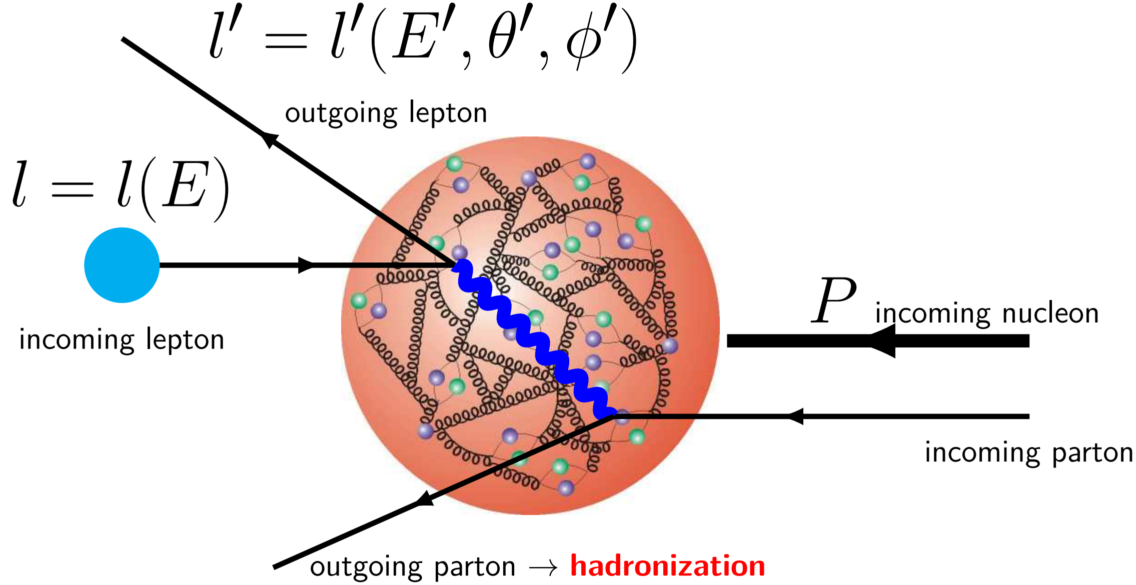

P incoming nucleon

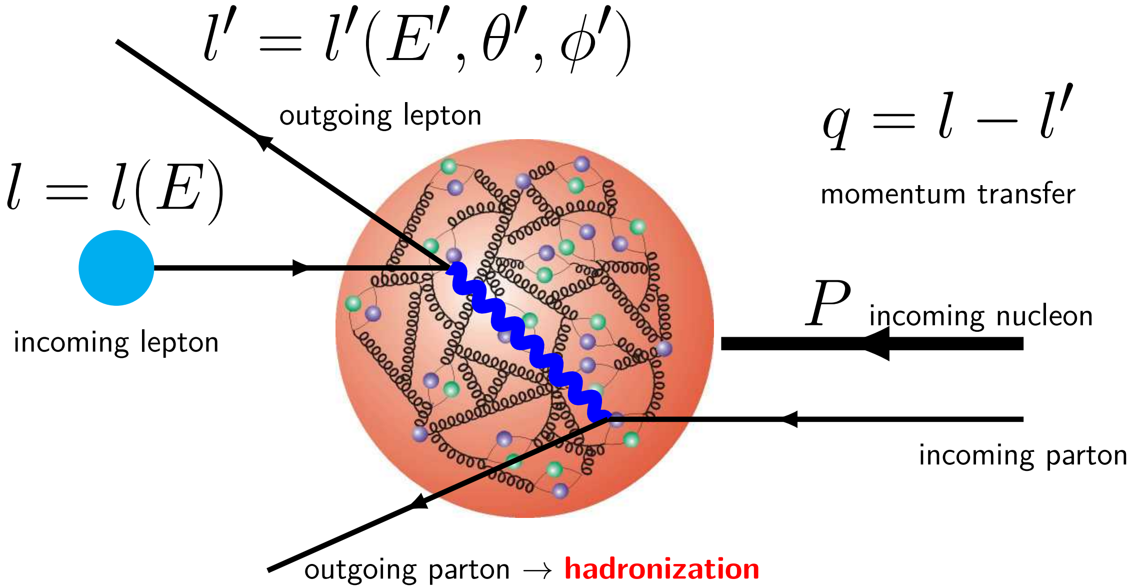


P incoming nucleon
incoming parton


$l = l(E)$
incoming lepton
P incoming nucleon
incoming parton


$l = l(E)$
incoming lepton
P
incoming nucleon
incoming parton


$l' = l'(E', \theta', \phi')$
outgoing lepton
$l = l(E)$
incoming lepton
P incoming nucleon
incoming parton
outgoing parton $\rightarrow$ **hadronization**


$l' = l'(E', \theta', \phi')$
outgoing lepton
$q = l - l'$
momentum transfer
$l = l(E)$
incoming lepton
P incoming nucleon
incoming parton
outgoing parton $\rightarrow$ **hadronization**

From counts to cross section

- The measurement

$$\mathcal{L}^{-1} \left.\frac{N}{\Delta E' \Delta \theta'}\right|_{\mathrm{bin}} = \left.\left\langle \frac{d\sigma}{dE' d\theta'} \right\rangle\right|_{\mathrm{bin}}$$

From counts to cross section

- The measurement

$$\mathcal{L}^{-1} \left.\frac{N}{\Delta E' \Delta\theta'}\right|_{\text{bin}} = \left.\left\langle \frac{d\sigma}{dE' d\theta'} \right\rangle\right|_{\text{bin}}$$

- Born cross section

$$\left\langle \frac{d\sigma}{dE' d\theta'} \right\rangle = \left\langle \text{Acc.} \otimes \underbrace{\text{Rad.Cor}}_{\textbf{QED}} \otimes \underbrace{\frac{d\sigma^{\text{Born}}}{dE' d\theta'}}_{\textbf{QCD}} \right\rangle$$

Kinematics variables

- DIS cross sections depend on 3 parameters:

$$E', \theta', s$$

Kinematics variables

- DIS cross sections depend on 3 parameters:

$$E', \theta', s$$

- A convenient set of Lorentz invariants

$$Q^2 = -q^2 \qquad x = \frac{Q^2}{2P \cdot q} \qquad y = \frac{P \cdot q}{P \cdot l}$$

Kinematics variables

- DIS cross sections depend on 3 parameters:

$$E', \theta', s$$

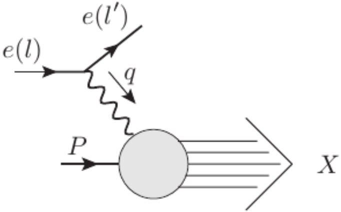
- A convenient set of Lorentz invariants

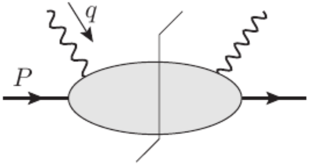
$$Q^2 = -q^2 \qquad x = \frac{Q^2}{2P \cdot q} \qquad y = \frac{P \cdot q}{P \cdot l}$$

- Differential cross sections are typically given as

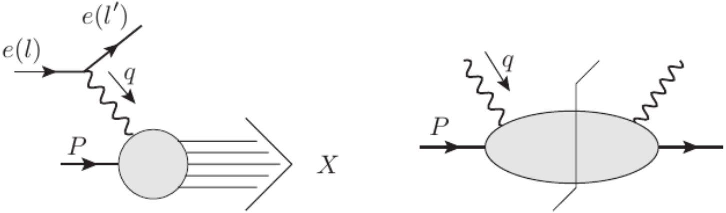
$$\frac{d\sigma}{dx dQ^2} \qquad \frac{d\sigma}{dx dy}$$

Cross section and structure functions - Collins 2.3



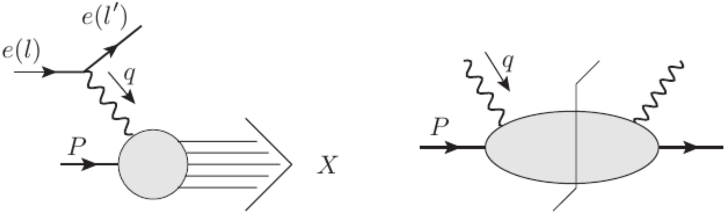


Cross section and structure functions - Collins 2.3



$$E' \frac{d\sigma}{d^3 \boldsymbol{l}'} \simeq \frac{\pi e^4}{2s} \sum_X \delta^{(4)}(p_X - P - q) \left| \langle l' | \, j_\mu^{\text{lept.}} \, | l \rangle \, \frac{1}{q^2} \, \langle X, \text{out} | \, j^\mu \, | P \rangle \right|^2$$

Cross section and structure functions - Collins 2.3



$$E' \frac{d\sigma}{d^3 \boldsymbol{l}'} \simeq \frac{\pi e^4}{2s} \sum_X \delta^{(4)}(p_X - P - q) \left| \langle l' | \, j_\mu^{\text{lept.}} \, | l \rangle \, \frac{1}{q^2} \, \langle X, \text{out} | \, j^\mu \, | P \rangle \right|^2$$

$$= \frac{2\alpha_{\text{EM}}^2}{sQ^4} L_{\mu\nu} W^{\mu\nu}$$

Cross section and structure functions

- Leptonic tensor

$$L_{\mu\nu} = 2(l_\mu l'_\nu + l'_\mu l_\nu - g_{\mu\nu} l \cdot l' - i\lambda\epsilon_{\mu\nu\alpha\beta} l^\alpha l'^\beta)$$

Cross section and structure functions

- Leptonic tensor

$$L_{\mu\nu} = 2(l_\mu l'_\nu + l'_\mu l_\nu - g_{\mu\nu} l \cdot l' - i\lambda \epsilon_{\mu\nu\alpha\beta} l^\alpha l'^\beta)$$

- Hadronic tensor

$$W^{\mu\nu} = 4\pi^3 \sum_X \delta^{(4)}(p_X - P - q) \langle P, S| j^\mu(0) |X\rangle \langle X| j^\nu(0) |P, S\rangle$$

Cross section and structure functions

- Leptonic tensor

$$L_{\mu\nu} = 2(l_\mu l'_\nu + l'_\mu l_\nu - g_{\mu\nu} l \cdot l' - i\lambda \epsilon_{\mu\nu\alpha\beta} l^\alpha l'^\beta)$$

- Hadronic tensor

$$\begin{aligned} W^{\mu\nu} &= 4\pi^3 \sum_X \delta^{(4)}(p_X - P - q) \langle P, S| j^\mu(0) |X\rangle \langle X| j^\nu(0) |P, S\rangle \\ &= \frac{1}{4\pi} \int d^4 z e^{iq\cdot z} \langle P, S| j^\mu(z) j^\nu(0) |P, S\rangle \end{aligned}$$

Cross section and structure functions

- Leptonic tensor

$$L_{\mu\nu} = 2(l_\mu l'_\nu + l'_\mu l_\nu - g_{\mu\nu} l \cdot l' - i\lambda \epsilon_{\mu\nu\alpha\beta} l^\alpha l'^\beta)$$

- Hadronic tensor

$$\begin{aligned}
W^{\mu\nu} &= 4\pi^3 \sum_X \delta^{(4)}(p_X - P - q) \langle P, S| j^\mu(0) |X\rangle \langle X| j^\nu(0) |P, S\rangle \\
&= \frac{1}{4\pi} \int d^4 z e^{iq\cdot z} \langle P, S| j^\mu(z) j^\nu(0) |P, S\rangle \\
&= \left(-g^{\mu\nu} + \frac{q^\mu q^\nu}{q^2}\right) F_1(x, Q^2) + \frac{\left(P^\mu - q^\mu \frac{P\cdot q}{q^2}\right)\left(P^\nu - q^\nu \frac{P\cdot q}{q^2}\right)}{P \cdot q} F_2(x, Q^2) \\
&+ i\epsilon^{\mu\nu\alpha\beta} \frac{q_\alpha S_\beta}{P \cdot q} g_1(x, Q^2) + i\epsilon^{\mu\nu\alpha\beta} \frac{q_\alpha \left(S_\beta - P_\beta \frac{S\cdot q}{P\cdot q}\right)}{P \cdot q} g_2(x, Q^2)
\end{aligned}$$

Cross section and structure functions

- Unpolarized cross sections

$$\frac{d\sigma}{dxdy} \simeq \frac{4\pi\alpha_{\mathrm{EM}}^2}{xyQ^2}\left[\left(1-y-\frac{x^2y^2M^2}{Q^2}\right)F_2(x,Q^2)+y^2xF_1(x,Q^2)\right]$$

Cross section and structure functions

- Unpolarized cross sections

$$\frac{d\sigma}{dxdy} \simeq \frac{4\pi\alpha_{\mathrm{EM}}^2}{xyQ^2}\left[\left(1 - y - \frac{x^2y^2M^2}{Q^2}\right) F_2(x,Q^2) + y^2 x F_1(x,Q^2)\right]$$

- Polarized cross sections

$$\Delta\sigma = \sigma(\lambda_N = -1, \lambda_l) - \sigma(\lambda_N = 1, \lambda_l)$$

Cross section and structure functions

- Unpolarized cross sections

$$\frac{d\sigma}{dxdy} \simeq \frac{4\pi\alpha_{\mathrm{EM}}^2}{xyQ^2}\left[\left(1-y-\frac{x^2y^2M^2}{Q^2}\right)F_2(x,Q^2)+y^2xF_1(x,Q^2)\right]$$

- Polarized cross sections

$$\Delta\sigma = \sigma(\lambda_N=-1,\lambda_l)-\sigma(\lambda_N=1,\lambda_l)$$

$$\frac{d\Delta\sigma}{dxdy} \simeq \frac{8\pi\alpha_{\mathrm{EM}}^2}{xyQ^2}\left[-\lambda_l y\left(2-y-2\frac{x^2y^2M^2}{Q^2}\right)xg_1(x,Q^2)+\lambda_l 4x^3y^2\frac{M^2}{Q^2}g_2(x,Q^2)\right]$$

Asymmetries for polarized DIS

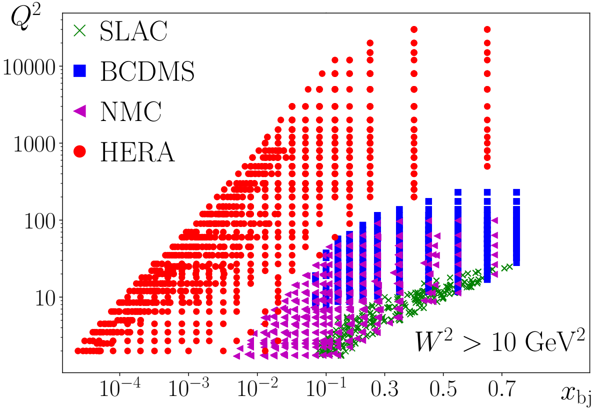
- Cross section rations are taken to minimize syst. uncertainties

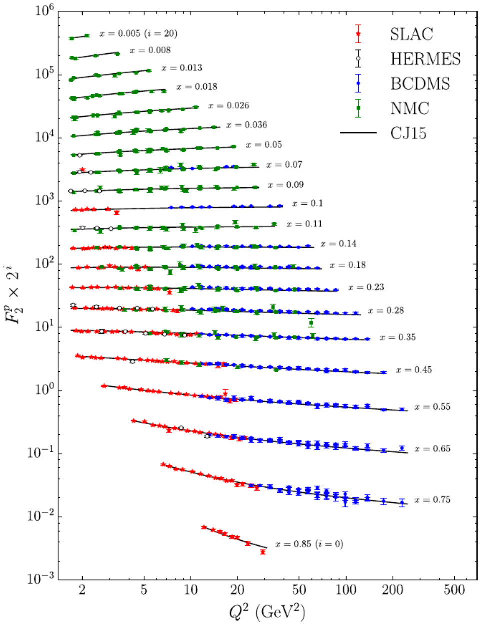
$$A_{||} = \frac{\sigma^{\uparrow\Downarrow} - \sigma^{\uparrow\Uparrow}}{\sigma^{\uparrow\Downarrow} + \sigma^{\uparrow\Downarrow}} = D(A_1 + \eta A_2)$$

$$A_{\perp} = \frac{\sigma^{\uparrow\Rightarrow} - \sigma^{\uparrow\Leftarrow}}{\sigma^{\uparrow\Rightarrow} + \sigma^{\uparrow\Leftarrow}} = d(A_2 - \xi A_1)$$

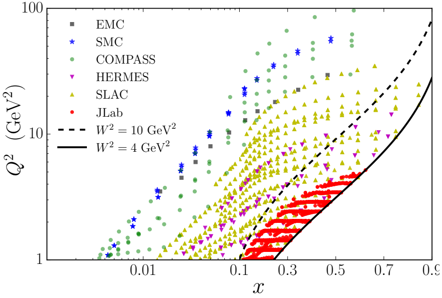
$$A_1 = \frac{(g_1 - \gamma^2 g_2)}{F_1} \qquad A_2 = \gamma \frac{(g_1 + g_2)}{F_1} \qquad \gamma^2 = \frac{4M^2x^2}{Q^2}$$

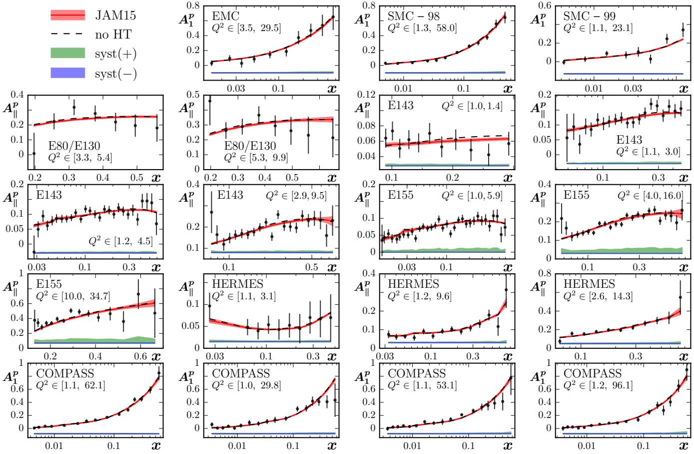
Experimental data: Unpolarized DIS





Experimental data: Polarized DIS





Part II: Elementary treatment of factorization

- Handbag diagrams, leading region and power counting
- Factorization, hard and soft parts
- Parton densities and DGLAP
- Treatments for the hard parts, subtractions

Handbag diagrams

- Differential cross sections are proportional to squared amplitudes

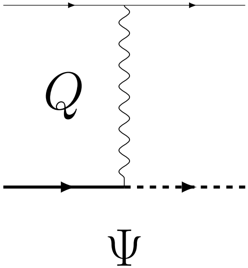
$$d\sigma = \frac{1}{\text{flux}} \mid \Psi \mid^2 d\Omega$$

Handbag diagrams

- Differential cross sections are proportional to squared amplitudes

$$d\sigma = \frac{1}{\mathrm{flux}} \mid \Psi \mid^2 d\Omega$$

- Schematically

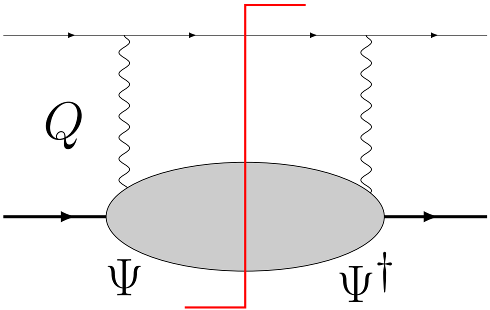


Handbag diagrams

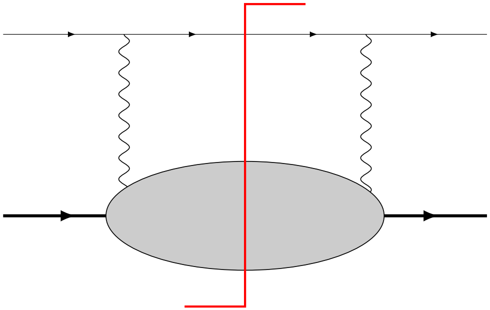
- Differential cross sections are proportional to squared amplitudes

$$d\sigma = \frac{1}{\mathrm{flux}} \mid \Psi \mid^2 d\Omega$$

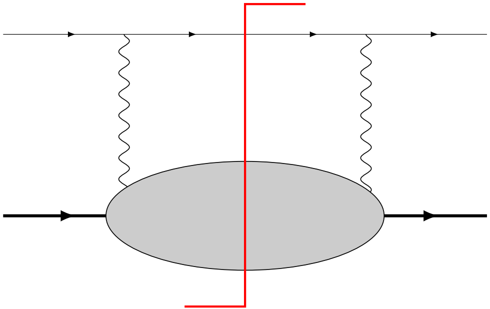
- Schematically

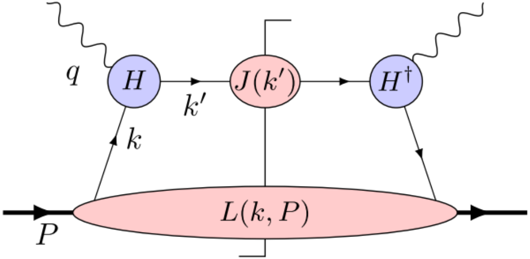


Leading regions - Collins 6, Moffat et al

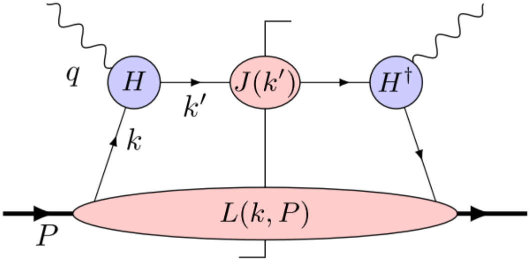


Leading regions - Collins 6, Moffat et al

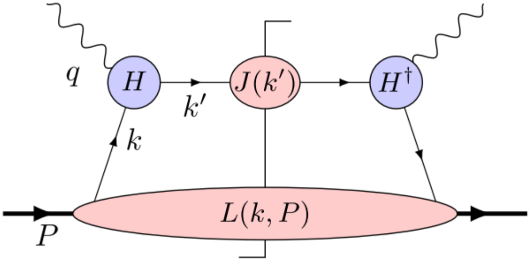




Leading regions



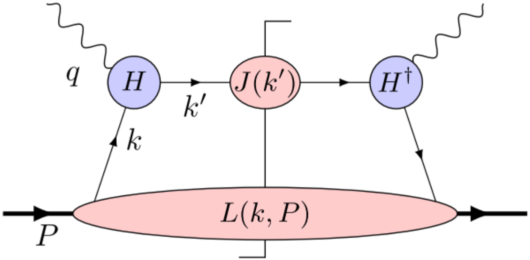
Leading regions



$$k \sim \left(O(Q), O\left(\frac{m^2}{Q}\right), O(\boldsymbol{m}_{\mathrm{T}}) \right)$$

$$k' \sim \left(O(Q), O\left(Q\right), O(\boldsymbol{m}_{\mathrm{T}}) \right)$$

Leading regions

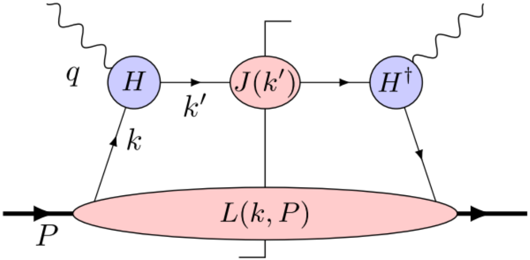


$$k \sim \left(O(Q), O\left(\frac{m^2}{Q}\right), O(\boldsymbol{m}_{\mathrm{T}})\right)$$

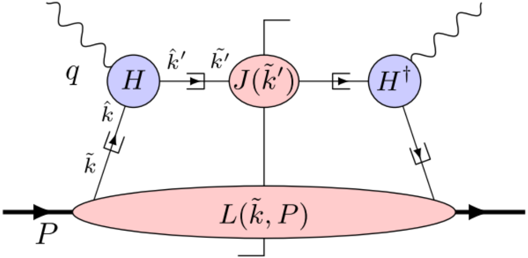
$$k' \sim \left(O(Q), O\left(Q\right), O(\boldsymbol{m}_{\mathrm{T}})\right)$$

$$W^{\mu\nu}(P,q) = \int \frac{d^4k}{(2\pi)^4} \mathrm{Tr}\left[\boxed{H^{\mu}(k,k')}\,\boxed{J(k')}\,\boxed{H^{\nu\dagger}(k,k')}\,\boxed{L(k,P)}\right]$$

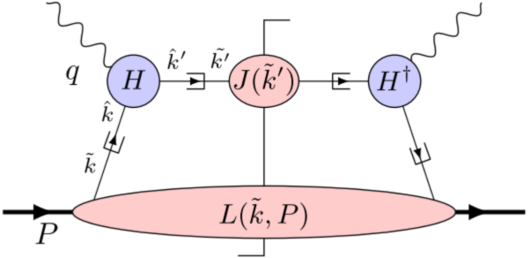
Approximations



Approximations



Approximations



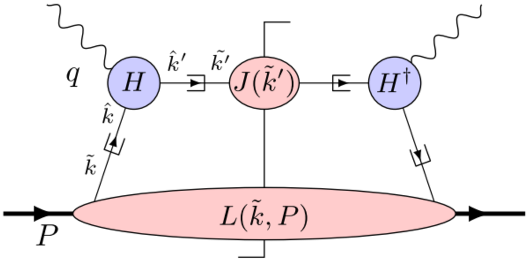
$k \rightarrow$ $\boxed{\hat{k} \sim \left(\xi P^+, 0, \mathbf{0}_\mathrm{T}\right)}$

$\boxed{\tilde{k} \sim \left(\xi P^+, k^-, \boldsymbol{k}_\mathrm{T}\right)}$

$k' \rightarrow$ $\boxed{\hat{k}' = \hat{k} + q}$

$$\boxed{\tilde{k}' \sim \left(-\xi P^+ + \frac{k_\mathrm{T}^2}{2q^-}, q^-, \mathbf{0}_\mathrm{T}\right)}$$

Approximations



$$k \to \boxed{\hat{k} \sim \left(\xi P^+, 0, \mathbf{0}_\mathrm{T}\right)}$$

$$\boxed{\tilde{k} \sim \left(\xi P^+, k^-, \boldsymbol{k}_\mathrm{T}\right)}$$

$$k' \to \boxed{\hat{k}' = \hat{k} + q}$$

$$\boxed{\tilde{k}' \sim \left(-\xi P^+ + \frac{k_\mathrm{T}^2}{2q^-}, q^-, \mathbf{0}_\mathrm{T}\right)}$$

$$W^{\mu\nu}(P,q) = \frac{\pi}{Q^2} \boxed{\mathrm{Tr}\left[H^\mu(Q^2)\hat{\not{k}}' H^{\nu\dagger}\hat{\not{k}}\right]} \boxed{\left(\int \frac{dk^- d^2\boldsymbol{k}_\mathrm{T}}{(2\pi)^4}\mathrm{Tr}\left[\frac{\gamma^+}{2}L(\tilde{k},P)\right]\right)} + O\left(\frac{m^2}{Q^2}\right)$$

Connection with nucleon structures

- The lower blob can be decomposed as follow

$$L(\tilde{k},P) = \gamma_\mu \Phi^\mu(\tilde{k},P) + \Phi_S(\tilde{k},P) + \gamma_5 \Phi_P(\tilde{k},P) + \gamma_5\gamma_\mu \Phi_A^\mu(\tilde{k},P) + \sigma_{\mu\nu}\Phi_T^{\mu\nu}(\tilde{k},P)$$

- In particular

$$\Phi^\mu(\tilde{k},P) = \int \frac{d^4z}{(2\pi)^4} e^{i\tilde{k}\cdot z} \langle P| \bar{\psi}(0)\gamma^\mu \psi(z) |P\rangle$$
$$\Phi_A^\mu(\tilde{k},P) = \int \frac{d^4z}{(2\pi)^4} e^{i\tilde{k}\cdot z} \langle P| \bar{\psi}(0)\gamma_5\gamma^\mu \psi(z) |P\rangle$$

- At leading power in m^2/Q^2 the "$+$" components dominate

$$L(\tilde{k},P) = \gamma_- \Phi^+(\tilde{k},P) + \gamma_5\gamma_-\Phi_A^+(\tilde{k},P) + ... + O(m^2/Q^2)$$

Connection with nucleon structures

- Then we get the so called light cone distributions

$$f(\xi) = \int \frac{dk^- d^2\boldsymbol{k}_\mathrm{T}}{(2\pi)^4} \mathrm{Tr}\left[\frac{\gamma^+}{2}\gamma_- \Phi^+(\tilde{k}, P)\right]$$
$$= \int \frac{dz^-}{2\pi} e^{-i\xi P^+ z^-} \langle P| \bar{\psi}(0, z^-, \mathbf{0}_\mathrm{T}) \frac{\gamma^+}{2} \psi(0) |P\rangle$$

$$\lambda_\mathrm{tar} \Delta f(\xi) = \int \frac{dk^- d^2\boldsymbol{k}_\mathrm{T}}{(2\pi)^4} \mathrm{Tr}\left[\frac{\gamma^+}{2}\gamma_5 \Phi_A^+(\tilde{k}, P)\right]$$
$$= \int \frac{dz^-}{2\pi} e^{-i\xi P^+ z^-} \langle P| \bar{\psi}(0, z^-, \mathbf{0}_\mathrm{T}) \frac{\gamma^+\gamma_5}{2} \psi(0) |P\rangle$$

aka unpolarized and polarized **parton distribution functions** (PDFs)

Issues with the approximations

- The approximations break momentum conservation

Issues with the approximations

- The approximations break momentum conservation
- $\boldsymbol{k}_{\mathrm{T}}$ integral is **unbounded**

Issues with the approximations

- The approximations break momentum conservation
- $\boldsymbol{k}_\mathrm{T}$ integral is **unbounded**
- The lower blob

$$f(x) = \int \frac{dk^- d^2\boldsymbol{k}_\mathrm{T}}{(2\pi)^4} \mathrm{Tr}\left[\frac{\gamma^+}{2} L(\tilde{k}, P)\right]$$

has a **UV** divergence $\rightarrow$ needs renormalization

RGE for PDFs aka DGLAP

- Multiplicative renormalization of composite operators

$$f_{j/H}(\xi;\mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} Z_{jj'}(z,g,\epsilon) f^{\text{bare}}_{j'/H}\left(\frac{\xi}{z}\right)$$

RGE for PDFs aka DGLAP

- Multiplicative renormalization of composite operators

$$f_{j/H}(\xi;\mu) = \sum_{j'} \int_\xi^1 \frac{dz}{z} Z_{jj'}(z,g,\epsilon) f^{\text{bare}}_{j'/H}\left(\frac{\xi}{z}\right)$$

- Using RG invariance of bare PDF we get

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

$$P_{jj'}(z,g) = \frac{1}{2}\frac{\partial}{\partial \ln \mu} \ln Z_{jj'}(z,g,\epsilon=0)$$

Factorization of cross sections or structure functions

- Recall the hadronic tensor

$$W^{\mu\nu} = \left(-g^{\mu\nu} + \frac{q^\mu q^\nu}{q^2}\right) F_1(x,Q^2) + \frac{\left(P^\mu - q^\mu \frac{P\cdot q}{q^2}\right)\left(P^\nu - q^\nu \frac{P\cdot q}{q^2}\right)}{P\cdot q} F_2(x,Q^2)$$
$$+ i\epsilon^{\mu\nu\alpha\beta}\frac{q_\alpha S_\beta}{P\cdot q} g_1(x,Q^2) + i\epsilon^{\mu\nu\alpha\beta}\frac{q_\alpha\left(S_\beta - P_\beta \frac{S\cdot q}{P\cdot q}\right)}{P\cdot q} g_2(x,Q^2)$$

- We just showed how to factorize $W^{\mu\nu}$ in terms of *hard* and *soft* parts

Factorization of cross sections or structure functions

- Factorization can be done at the structure function level, e.g

$$F_2(x,Q) = x\sum_j e_j^2 \int_x^1 \frac{d\xi}{\xi} C_2(\xi,\mu) f_j\left(\frac{x}{\xi},\mu\right)$$

- Notice that $f_j(\xi,\mu)$ is already a renormalized PDF

- The task now is to get C_2.

Treatment of the hard part

- **Attention**: this is key to define C_2 or any other hard part
  - **see Collins 9.6**

DIS Factorization is a procedure independent of the target

Treatment of the hard part

- **Attention**: this is key to define C_2 or any other hard part - **see Collins 9.6**

DIS Factorization is a procedure independent of the target

- Why is this so crucial? $\rightarrow$ because we can calculate C_2 using a simpler target $\rightarrow$ a **"parton target"**

Treatment of the hard part

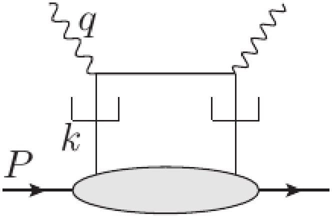
- **Attention**: this is key to define C_2 or any other hard part - **see Collins 9.6**

DIS Factorization is a procedure independent of the target

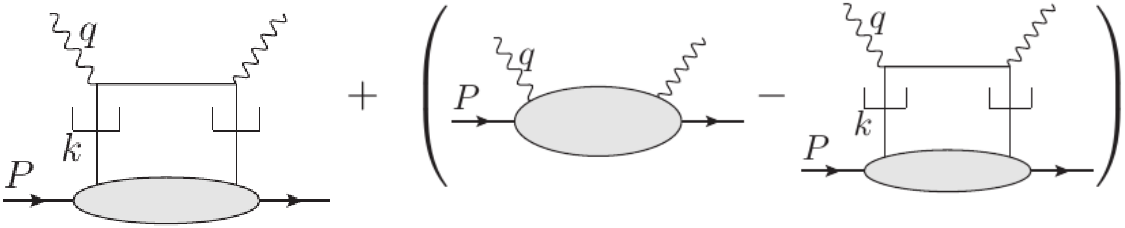
- Why is this so crucial? $\rightarrow$ because we can calculate C_2 using a simpler target $\rightarrow$ a **"parton target"**

- The procedure to compute the hard part is called **"successive approximation method"** - **see Collins 8.8**

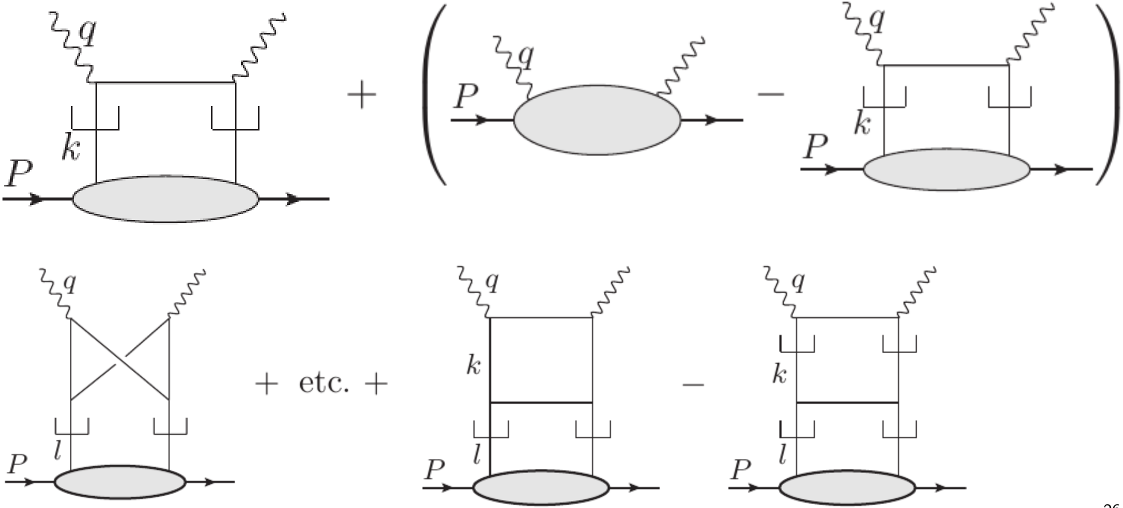
$$F_2 = \mathrm{T_{LO}} F_2$$



$$F_2 = \mathrm{T}_{\mathrm{LO}} F_2 + [F_2 - \mathrm{T}_{\mathrm{LO}} F_2]$$



$$F_2 = \textcolor{red}{\mathrm{T}_{\mathrm{LO}}} F_2 + \textcolor{blue}{\mathrm{T}_{\mathrm{NLO}}} \left[F_2 - \textcolor{red}{\mathrm{T}_{\mathrm{LO}}} F_2 \right]$$



Treatment for the hard part

- Work with parton target

$$
\begin{aligned}
F_2 &= \left(C^{(0)} + C^{(1)}_{\mathrm{unsub}} + \ldots\right) \otimes \left(f^{(0)} + f^{(1)} + \ldots\right) \\
&= \left(C^{(0)} + C^{(1)}_{\mathrm{unsub}} + \ldots\right) \otimes \left(\delta(1-\xi) - \frac{\alpha_S}{4\pi}\frac{S_\epsilon}{\epsilon}P(\xi) + \ldots\right)
\end{aligned}
$$

Treatment for the hard part

- Work with parton target

$$F_2 = (C^{(0)} + C^{(1)}_{\text{unsub}} + \ldots) \otimes (f^{(0)} + f^{(1)} + \ldots)$$
$$= (C^{(0)} + C^{(1)}_{\text{unsub}} + \ldots) \otimes \left(\delta(1-\xi) - \frac{\alpha_S}{4\pi}\frac{S_\epsilon}{\epsilon}P(\xi) + \ldots\right)$$

- Apply the subtraction procedure

$$F_2 = \mathrm{T}_{\mathrm{LO}}F_2 + \mathrm{T}_{\mathrm{NLO}}\left[F_2 - \mathrm{T}_{\mathrm{LO}}F_2\right] + O(\alpha_S^2)$$
$$= C^{(0)} \otimes f^{(0)} + C^{(1)}_{\text{unsub}} \otimes f^{(0)} - C^{(0)} \otimes f^{(1)} + O(\alpha_S^2)$$
$$= C^{(0)} + \left[C^{(1)}_{\text{unsub}} + \frac{\alpha_S}{4\pi}\frac{S_\epsilon}{\epsilon}P(\xi)\right] + O(\alpha_S^2)$$

Critique of conventional treatments - Collins 9.11

- One finds in the literature the following assesment

$$
\begin{aligned}
F_2 &= C \otimes f^{\mathrm{bare}} \\
&= C_{\mathrm{finite}} \otimes D_{\mathrm{IR\ div.}} \otimes f^{\mathrm{bare}} \\
&= C_{\mathrm{finite}} \otimes f^{\mathrm{ren.}}
\end{aligned}
$$

Critique of conventional treatments - Collins 9.11

- One finds in the literature the following assesment

$$
\begin{aligned}
F_2 &= C \otimes f^{\mathrm{bare}} \\
&= C_{\mathrm{finite}} \otimes D_{\mathrm{IR\ div.}} \otimes f^{\mathrm{bare}} \\
&= C_{\mathrm{finite}} \otimes f^{\mathrm{ren.}}
\end{aligned}
$$

- For certain observables like DIS the results are the same

Critique of conventional treatments - Collins 9.11

- One finds in the literature the following assesment

$$
\begin{aligned}
F_2 &= C \otimes f^{\text{bare}} \\
&= C_{\text{finite}} \otimes D_{\text{IR div.}} \otimes f^{\text{bare}} \\
&= C_{\text{finite}} \otimes f^{\text{ren.}}
\end{aligned}
$$

- For certain observables like DIS the results are the same

- Obscures the logic of factorization as it does not involve the subtractions method $\rightarrow$ **needed in TMD physics i.e W+Y**

Part III: Solving RGE

- The β function
- Mellin transforms
- DGLAP - non-singlet and singlet evolution

The β function

- RGE for strong coupling

$$a_S(\mu^2) = \frac{\alpha_S(\mu^2)}{4\pi}$$
$$\frac{da_S}{d\ln\mu^2} = \beta(a_s) = -\left(\beta_0 a_S^2 + \beta_1 a_S^3 + \ldots\right)$$

The β function

- RGE for strong coupling

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

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

- Coefficients up to two loops

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

The β function

- The RGE depends on $N_f \rightarrow$ **"mass thresholds"**

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

The β function

- The RGE depends on $N_f \to$ **"mass thresholds"**

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

- The RGE is discontinuous at the mass thresholds

The β function

- The RGE depends on $N_f \to$ **"mass thresholds"**

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

- The RGE is discontinuous at the mass thresholds

- a_S is continuous at the mass thresholds

The β function

- To solve the RGE we need boundary conditions (BC) for each N_f
- But we need continuous $a_s \to$ the recipe
  - Start with $a_S(m_Z) = 0.118/4\pi$
  - Compute $a_S(m_b)$ evolved with BC $a_S(m_Z)$ and $N_f = 5$
  - Compute $a_S(m_c)$ evolved with BC $a_S(m_b)$ and $N_f = 4$

QCD carpentry: coding up α_S RGE

- params.py
- alphaS.py

params.py

```
#--color factors
CA=3.0
CF=4.0/3.0
TR=0.5
TF=0.5

#--set_masses
mc    = 1.28
mb    = 4.18
mZ    = 91.1876
mW    = 80.398
M     = 0.93891897

mc2    = mc**2
mb2    = mb**2
mZ2    = mZ**2
M2     = M**2
```

params.py

```
#--QED and QCD couplings
alfa  = 1/137.036
alphaSMZ = 0.118
alfa2  = alfa**2

#--quark charges

eU2 = 4.0/9.0
eD2 = 1.0/9.0

couplings={1:eU2,2:eD2,3:eD2,4:eU2,5:eD2,6:eU2}
```

alphaS.py

```
#!/usr/bin/env python
import sys,os
import numpy as np
import params as par

beta=np.zeros((7,3))
for Nf in range(3,7):
    beta[Nf,0]=11.0-2.0/3.0*Nf
    beta[Nf,1]=102.-38.0/3.0*Nf

def beta_func(a,Nf):
    return -(beta[Nf,0]+a*beta[Nf,1])*a**2

def get_Nf(Q2):
    Nf=3
    if Q2>=(4.*par.mc2): Nf+=1
    if Q2>=(4.*par.mb2): Nf+=1
    return Nf
```

alphaS.py

```
def evolve_a(Q20,a,Q2,Nf):
    #--Runge-Kutta implemented in pegasus
    LR = np.log(Q2/Q20)/20.0
    for k in range(20):
        XK0 = LR * beta_func(a,Nf)
        XK1 = LR * beta_func(a + 0.5 * XK0,Nf)
        XK2 = LR * beta_func(a + 0.5 * XK1,Nf)
        XK3 = LR * beta_func(a + XK2,Nf)
        a+= (XK0 + 2.* XK1 + 2.* XK2 + XK3) * 0.166666666666666
    return a

#--build boundary conditions
ab=evolve_a(par.mZ2,par.aZ,par.mb2,5)
ac=evolve_a(par.mb2,ab,par.mc2,4)
a0=evolve_a(par.mc2,ac,par.Q20,3)
```

alphaS.py

```
storage={}
def get_a(Q2):
    if Q2 not in storage:
        if par.mb2<=Q2:
            storage[Q2]=evolve_a(par.mb2,ab,Q2,5)
        elif par.mc2<=Q2 and Q2<par.mb2:
            storage[Q2]=evolve_a(par.mc2,ac,Q2,4)
        elif Q2<par.mc2:
            storage[Q2]=evolve_a(par.Q20,a0,Q2,3)
    return storage[Q2]

def get_alphaS(Q2):
    return get_a(Q2)*4*np.pi
```

alphaS.py

```
if __name__=='__main__':

    print '========================='
    print 'test alphaS evolution'
    print '========================='
    print 'Q2=1             alphaS=%0.5f'%get_alphaS(1.0)
    print 'Q2=(1+mc2)/2     alphaS=%0.5f'%get_alphaS(0.5*(1.0+par.mc2))
    print 'Q2=mc2           alphaS=%0.5f'%get_alphaS(par.mc2)
    print 'Q2=(mc2+mb2)/2 alphaS=%0.5f'%get_alphaS(0.5*(par.mc2+par.mb2))
    print 'Q2=mb2           alphaS=%0.5f'%get_alphaS(par.mb2)
    print 'Q2=(mb2+mZ2)/2 alphaS=%0.5f'%get_alphaS(0.5*(par.mb2+par.mZ2))
    print 'Q2=mZ2           alphaS=%0.5f'%get_alphaS(par.mZ2)
```

Next steps at command line

```
chmod +x alphaS.py
./alpha.py
```

Mellin transform

- Definition:

$$\boxed{F(N)} = \int dx x^{N-1} \boxed{f(x)} \quad \to \quad \boxed{f(x)} = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} dN x^{-N} \boxed{F(N)}$$

Mellin transform

- Definition:

$$\boxed{F(N)} = \int dx x^{N-1} \boxed{f(x)} \quad \rightarrow \quad \boxed{f(x)} = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} dN x^{-N} \boxed{F(N)}$$

- The c is chosen to be at the right of the rightmost pole of $F(N)$

Mellin transform

- Definition:

$$\boxed{F(N)} = \int dx x^{N-1} \boxed{f(x)} \quad \rightarrow \quad \boxed{f(x)} = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} dN x^{-N} \boxed{F(N)}$$

- The c is chosen to be at the right of the rightmost pole of $F(N)$

- Example:
  - $f(x) = x \rightarrow F(N) = \frac{1}{N+1}$
  - The rightmost pole is at $N = -1$ so $c > -1$

Mellin transform: numerical recipe

- Parametrize the contour

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

Mellin transform: numerical recipe

- Parametrize the contour

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

- A useful identity

$$f(x) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} dN x^{-N} F(N)$$

$$= \frac{1}{\pi}\int_{0}^{\infty} dz \mathrm{Im}\left[e^{i\phi} x^{-N(z)} F(N(z))\right]$$

Mellin transform: numerical recipe

- For efficiency purposes we use Gaussian quadrature, i.e

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

Mellin transform: numerical recipe

- For efficiency purposes we use Gaussian quadrature, i.e

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

- For an arbitrary integration region we can use

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

Mellin transform: numerical recipe

- Back to inverse Mellin transform

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

Mellin transform: numerical recipe

- Back to inverse Mellin transform

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

- We partition the z integration in k subintervals and apply Gaussian-quadrature on each subinterval

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

$$z_i^j = \frac{1}{2}\left[\left(z_{\max}^j - z_{\min}^j\right) x_i + \left(z_{\max}^j + z_{\min}^j\right)\right]$$

QCD carpentry: coding up Mellin transforms

- mellin.py

mellin.py

```
#!/usr/bin/env python
import sys,os
import numpy as np

c=1.9   #--contour crossing
npts=8  #--number of gaussian quadrature

#--define intervals for z integration
znodes=[0,0.1,0.3,0.6,1.0,1.6,2.4,3.5,5,7,10,14,19,25,32,40,50,63]

#--compute gaussian nodes and weights
x,w=np.polynomial.legendre.leggauss(npts)
```

mellin.py

```
#--generate z and w values along coutour
Z,W,JAC=[],[],[]
for i in range(len(znodes)-1):
    a,b=znodes[i],znodes[i+1]
    Z.extend(0.5*(b-a)*x+0.5*(a+b))
    W.extend(w)
    JAC.extend([0.5*(b-a) for j in range(x.size)])
Z=np.array(Z)
W=np.array(W)
JAC=np.array(JAC)
```

mellin.py

```
def invert(x,F):
    return np.sum(np.imag(phase * x**(-N) * F)/np.pi * W * JAC)

if __name__=='__main__':

    F=1/(N+1)
    f=lambda x: x
    X=10**np.linspace(-5,-1,10)
    for x in X:
        print 'x=%10.4e  f=%10.4e  inv=%10.4e'%(x,f(x),invert(x,F))
```

Next steps at command line

```
chmod +x mellin.py
./mellin.py
```

...Questions?