QCD Basics for accurate LHC Physics

Lecture III

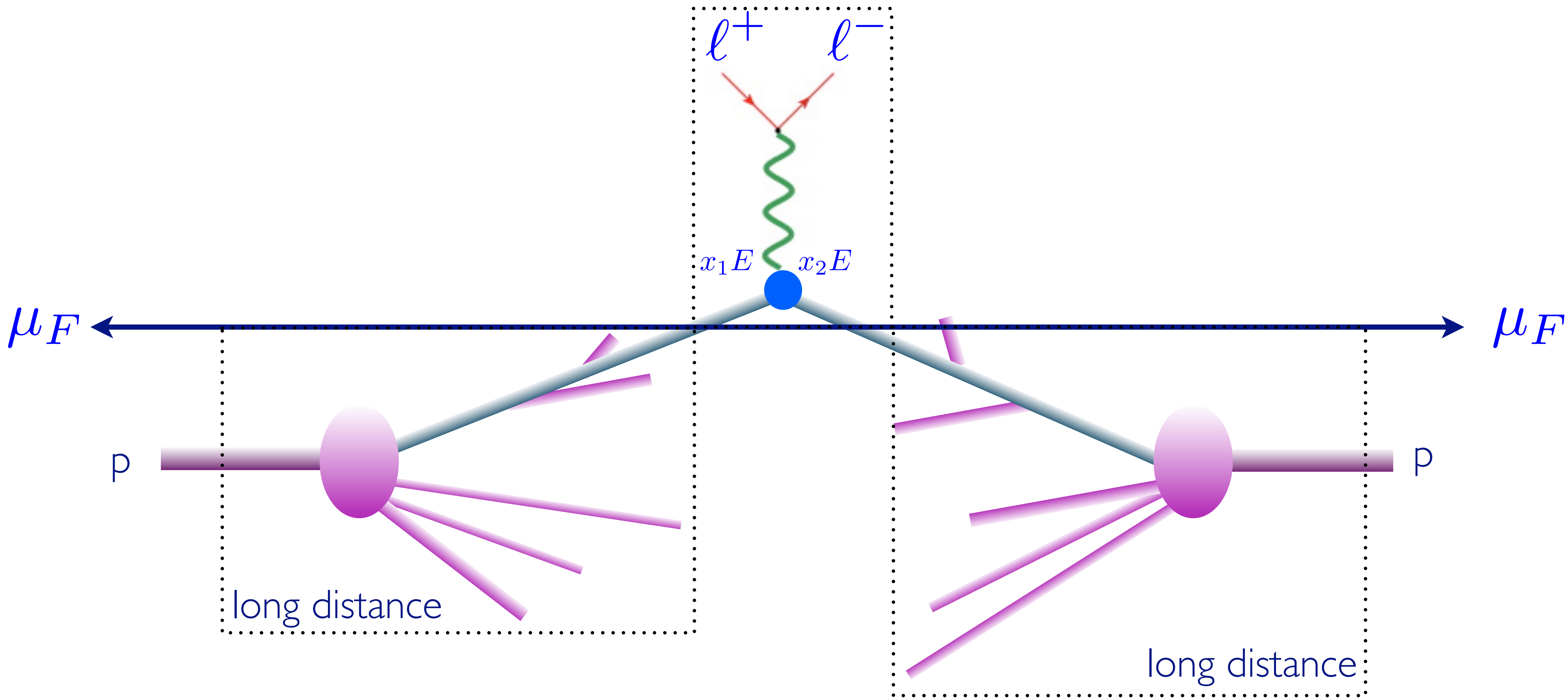
Fabio Maltoni
Center for Particle Physics and Phenomenology (CP3)
Université Catholique de Louvain

Claims and Aims

Four lectures:

1. Intro and QCD fundamentals
2. QCD in the final state
3. From accurate QCD to useful QCD
4. Advanced QCD with applications at the LHC

LHC master formula



$$\sigma_X = \sum_{a,b} \int_0^1 dx_1 dx_2\, f_a(x_1, \mu_F^2) f_b(x_2, \mu_F^2) \times \hat{\sigma}_{ab\to X}(x_1, x_2, \alpha_S(\mu_R^2), \frac{Q^2}{\mu_F^2}, \frac{Q^2}{\mu_R^2})$$

LHC master formula

$$\sigma_X = \sum_{a,b} \int_0^1 dx_1 dx_2\, f_a(x_1, \mu_F^2) f_b(x_2, \mu_F^2) \times \hat{\sigma}_{ab\to X}(x_1, x_2, \alpha_S(\mu_R^2), \frac{Q^2}{\mu_F^2}, \frac{Q^2}{\mu_R^2})$$

Two ingredients necessary:

1. Parton Distribution functions (from exp, but evolution from th).

2. Short distance coefficients as an expansion in α_S (from th).

$$\hat{\sigma}_{ab\to X} = \sigma_0 + \alpha_S \sigma_1 + \alpha_S^2 \sigma_2 + \ldots$$

Leading order

Next-to-leading order

Next-to-next-to-leading order

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

1. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

I. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

II. For each one, calculate the amplitude:

$$\mathcal{A}(\{p\},\{h\},\{c\}) = \sum_i D_i$$

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

I. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

II. For each one, calculate the amplitude:

$$\mathcal{A}(\{p\},\{h\},\{c\}) = \sum_i D_i$$

III. Square the amplitude, sum over spins & color, integrate over the phase space (D ~ 3n)

$$\hat{\sigma} = \frac{1}{2\hat{s}} \int d\Phi_p \sum_{h,c} |\mathcal{A}|^2$$

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

I. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

easy

II. For each one, calculate the amplitude:

$$\mathcal{A}(\{p\},\{h\},\{c\}) = \sum_i D_i$$

III. Square the amplitude, sum over spins & color, integrate over the phase space (D ~ 3n)

$$\hat{\sigma} = \frac{1}{2\hat{s}} \int d\Phi_p \sum_{h,c} |\mathcal{A}|^2$$

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

I. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

easy

II. For each one, calculate the amplitude:

$$\mathcal{A}(\{p\},\{h\},\{c\}) = \sum_i D_i$$

difficult

III. Square the amplitude, sum over spins & color, integrate over the phase space (D ~ 3n)

$$\hat{\sigma} = \frac{1}{2\hat{s}} \int d\Phi_p \sum_{h,c} |\mathcal{A}|^2$$

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

I. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

easy

II. For each one, calculate the amplitude:

$$\mathcal{A}(\{p\},\{h\},\{c\}) = \sum_i D_i$$

difficult

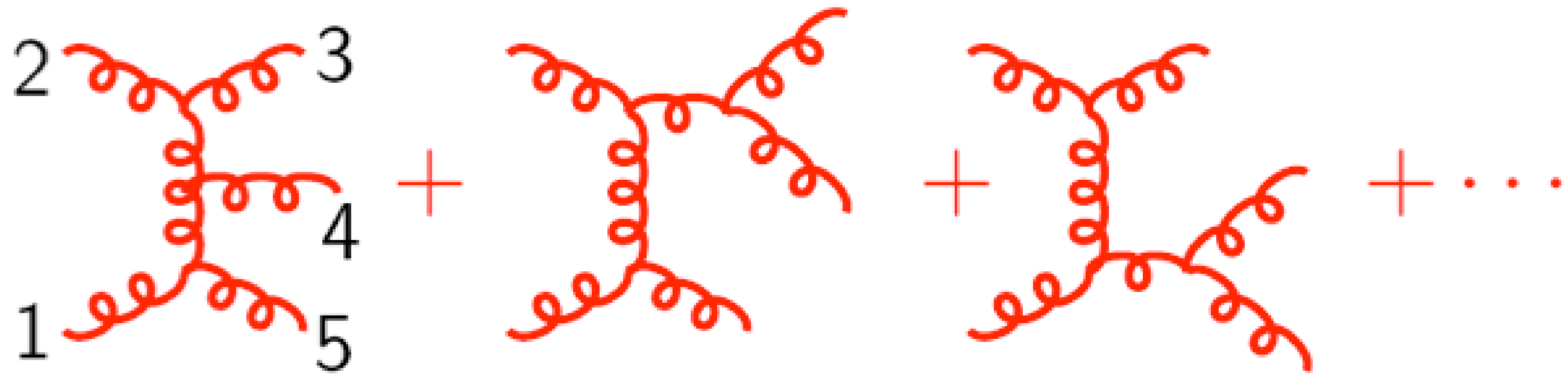
III. Square the amplitude, sum over spins & color, integrate over the phase space (D ~ 3n)

$$\hat{\sigma} = \frac{1}{2\hat{s}} \int d\Phi_p \sum_{h,c} |\mathcal{A}|^2$$

quite hard

How difficult is it to calculate |A|² for arbitrary processes?

Consider a simple 5 gluon amplitude:



There are 25 diagrams with a complicated tensor structure,
so you get....

A(k1,e1,k2,e2,k3,e3,k4,e4,k5,e5) = + Tr(Ta1,Ta2,Ta3,Ta4,Ta5) * (1/2*den(2*k1.k2)*k1.e2*e1.e3*e4.e5 - den(2*k1.k2)*k1.e2*e1.e4*e3.e5
+ 1/2*den(2*k1.k2)*k1.e2*e1.e5*e3.e4 - 1/4 *den(2*k1.k2)*k1.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*k1.e4*e1.e2*e3.e5 - 1/4*den(2*k1.k2)*k1.e5*e1.e2*e3.e4
- 1/2*den(2*k1.k2)*k2.e1*e2.e3* e4.e5 + den(2*k1.k2)*k2.e1*e2.e4*e3.e5 - 1/2*den(2*k1.k2)*k2.e1*e2.e5 *e3.e4 + 1/4*den(2*k1.k2)*k2.e3*e1.e2*e4.e5
- 1/2*den(2*k1.k2)*k2.e4* e1.e2*e3.e5 + 1/4*den(2*k1.k2)*k2.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)* den(2*k3.k4)*k1.k3*k1.e2*e1.e5*e3.e4
- 1/2*den(2*k1.k2)*den(2*k3.k4)* k1.k3*k2.e1*e2.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k2.e5* e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k3.e4*e1.e2*e3.e5
+ 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k4.e3*e1.e2*e4.e5 - 1/2*den(2* k1.k2)*den(2*k3.k4)*k1.k3*k4.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*
k3.k4)*k1.k4*k1.e2*e1.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4* k2.e1*e2.e5*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k2.e5*e1.e2*
e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k3.e4*e1.e2*e3.e5 + 1/2* den(2*k1.k2)*den(2*k3.k4)*k1.k4*k3.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*
den(2*k3.k4)*k1.k4*k4.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*den(2*k3.k4)* k1.k5*k3.e4*e1.e2*e3.e5 - 1/4*den(2*k1.k2)*den(2*k3.k4)*k1.k5*k3.e5*
e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k5*k4.e3*e1.e2*e4.e5 + 1/4*den(2*k1.k2)*den(2*k3.k4)*k1.k5*k4.e5*e1.e2*e3.e4 -
den(2* k1.k2)*den(2*k3.k4)*k1.e2*k1.e3*k3.e4*e1.e5 + den(2*k1.k2)*den(2* k3.k4)*k1.e2*k1.e4*k4.e3*e1.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k1.e5*k3.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k3.e4*e1.e3 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k4.e1*e3.e4
- den(2* k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k4.e3*e1.e4 + 1/2*den(2*k1.k2)*den(2* k3.k4)*k1.e2*k2.k3*e1.e5*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k2.k4*e1.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e3*k3.e4*e1.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e4*k4.e3*e1.e5 - 1/2*den(2*
k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k3.e1*e3.e4 + den(2*k1.k2)*den(2* k3.k4)*k1.e2*k2.e5*k3.e4*e1.e3 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k2.e5*k4.e1*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k4.e3*e1.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.k5*e1.e5*e3.e4 +
den(2* k1.k2)*den(2*k3.k4)*k1.e2*k3.e1*k3.e4*e3.e5 - den(2*k1.k2)*den(2* k3.k4)*k1.e2*k3.e1*k4.e3*e4.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k3.e1*k4.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e4*k3.e5*e1.e3 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e4*k4.e1*e3.e5 - den(2*k1.k2)*
den(2*k3.k4)*k1.e2*k3.e4*k4.e5*e1.e3 - den(2*k1.k2)*den(2*k3.k4)* k1.e2*k3.e4*k5.e1*e3.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e4*k5.e3
*e1.e5 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e5*k4.e1*e3.e4 + den(2* k1.k2)*den(2*k3.k4)*k1.e2*k3.e5*k4.e3*e1.e4 + 1/2*den(2*k1.k2)*den(2*
k3.k4)*k1.e2*k3.e5*k5.e1*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2* k4.k5*e1.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e1*k4.e3*e4.e5
+ den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e3*k4.e5*e1.e4 + den(2*k1.k2)* den(2*k3.k4)*k1.e2*k4.e3*k5.e1*e4.e5 - den(2*k1.k2)*den(2*k3.k4)*
k1.e2*k4.e3*k5.e4*e1.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e5* k5.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e3*k2.e1*k3.e4*e2.e5
- den(2*k1.k2)*den(2*k3.k4)*k1.e3*k2.e5*k3.e4*e1.e2 + 1/2*den(2*k1.k2)* den(2*k3.k4)*k1.e3*k3.e4*k3.e5*e1.e2 + 1/2*den(2*k1.k2)*den(2*k3.k4)*
k1.e3*k3.e4*k4.e5*e1.e2 - den(2*k1.k2)*den(2*k3.k4)*k1.e4*k2.e1*k4.e3 *e2.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e4*k2.e5*k4.e3*e1.e2
- 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e4*k3.e5*k4.e3*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e4*k4.e3*k4.e5*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*
k1.e5*k2.k3*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.k4* e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k3.e2*e3.e4
- den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k3.e4*e2.e3 - 1/2*den(2* k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k4.e2*e3.e4 + den(2*k1.k2)*den(2*
k3.k4)*k1.e5*k2.e1*k4.e3*e2.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e5* k2.e3*k3.e4*e1.e2 - den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e4*k4.e3*e1.e2
+ 1/4*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k3.k5*e1.e2*e3.e4 - 1/2*den(2* k1.k2)*den(2*k3.k4)*k1.e5*k3.e4*k5.e3*e1.e2 - 1/4*den(2*k1.k2)*den(2*
k3.k4)*k1.e5*k4.k5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5* k4.e3*k5.e4*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k2.e1*e2.e5*
e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k3.e4*e1.e2*e3.e5 - 1/2* den(2*k1.k2)*den(2*k3.k4)*k2.k3*k4.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*
den(2*k3.k4)*k2.k3*k4.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)* k2.k4*k2.e1*e2.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k3.e4*
e1.e2*e3.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k3.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k4.e3*e1.e2*e4.e5 - 1/2*den(2*
k1.k2)*den(2*k3.k4)*k2.k5*k3.e4*e1.e2*e3.e5 + 1/4*den(2*k1.k2)*den(2* k3.k4)*k2.k5*k3.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k5*
k4.e3*e1.e2*e4.e5 - 1/4*den(2*k1.k2)*den(2*k3.k4)*k2.k5*k4.e5*e1.e2* e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e3*k3.e4*e2.e5 -
den(2* k1.k2)*den(2*k3.k4)*k2.e1*k2.e4*k4.e3*e2.e5 + 1/2*den(2*k1.k2)*den(2* k3.k4)*k2.e1*k2.e5*k3.e2*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k2.e1*
k2.e5*k3.e4*e2.e3 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k4.e2* e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k4.e3*e2.e4

ZviBern®

```
A(k1,e1,k2,e2,k3,e3,k4,e4,k5,e5) = + Tr(Ta1,Ta2,Ta3,Ta4,Ta5) * ( 1/2*den(2*k1.k2)*k1.e2*e1.e3*e4.e5 - den(2*k1.k2)*k1.e2*e1.e4*e3.e5
+ 1/2*den(2*k1.k2)*k1.e2*e1.e5*e3.e4 - 1/4*den(2*k1.k2)*k1.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*k1.e4*e1.e2*e3.e5 - 1/4*den(2*k1.k2)*k1.e5*e1.e2*e3.e4
- 1/2*den(2*k1.k2)*k2.e1*e2.e3* e4.e5 + den(2*k1.k2)*k2.e1*e2.e4*e3.e5 - 1/2*den(2*k1.k2)*k2.e1*e2.e5 *e3.e4 + 1/4*den(2*k1.k2)*k2.e3*e1.e2*e4.e5
- 1/2*den(2*k1.k2)*k2.e4* e1.e2*e3.e5 + 1/4*den(2*k1.k2)*k2.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)* den(2*k3.k4)*k1.k3*k1.e2*e1.e5*e3.e4
- 1/2*den(2*k1.k2)*den(2*k3.k4)* k1.k3*k2.e1*e2.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k2.e5* e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k3.e4*e1.e2*e3.e5
+ 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k4.e3*e1.e2*e4.e5 - 1/2*den(2* k1.k2)*den(2*k3.k4)*k1.k3*k4.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*
k3.k4)*k1.k4*k1.e2*e1.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4* k2.e1*e2.e5*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k2.e5*e1.e2*
e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k3.e4*e1.e2*e3.e5 + 1/2* den(2*k1.k2)*den(2*k3.k4)*k1.k4*k3.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*
```

```
+ Tr(Ta1,Ta2,Ta3,Ta4,Ta5) * ( 1/2*den(2*k1.k2)*k1.e2*e1.e3*e4.e5
```

```
k1.e5*k3.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k3.e4*e1.e3 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k4.e1*e3.e4
- den(2* k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k4.e3*e1.e4 + 1/2*den(2*k1.k2)*den(2* k3.k4)*k1.e2*k2.k3*e1.e5*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k2.k4*e1.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e3*k3.e4*e1.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e4*k4.e3*e1.e5 - 1/2*den(2*
k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k3.e1*e3.e4 + den(2*k1.k2)*den(2* k3.k4)*k1.e2*k2.e5*k3.e4*e1.e3 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k2.e5*k4.e1*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k4.e3*e1.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.k5*e1.e5*e3.e4 +
den(2* k1.k2)*den(2*k3.k4)*k1.e2*k3.e1*k3.e4*e3.e5 - den(2*k1.k2)*den(2* k3.k4)*k1.e2*k3.e1*k4.e3*e4.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k3.e1*k4.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e4*k3.e5*e1.e3 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e4*k4.e1*e3.e5 - den(2*k1.k2)*
den(2*k3.k4)*k1.e2*k3.e4*k4.e5*e1.e3 - den(2*k1.k2)*den(2*k3.k4)* k1.e2*k3.e4*k5.e1*e3.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e4*k5.e3
*e1.e5 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e5*k4.e1*e3.e4 + den(2* k1.k2)*den(2*k3.k4)*k1.e2*k3.e5*k4.e3*e1.e4 + 1/2*den(2*k1.k2)*den(2*
k3.k4)*k1.e2*k3.e5*k5.e1*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2* k4.k5*e1.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e1*k4.e3*e4.e5
+ den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e3*k4.e5*e1.e4 + den(2*k1.k2)* den(2*k3.k4)*k1.e2*k4.e3*k5.e1*e4.e5 - den(2*k1.k2)*den(2*k3.k4)*
k1.e2*k4.e3*k5.e4*e1.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e5* k5.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e3*k2.e1*k3.e4*e2.e5
- den(2*k1.k2)*den(2*k3.k4)*k1.e3*k2.e5*k3.e4*e1.e2 + 1/2*den(2*k1.k2)* den(2*k3.k4)*k1.e3*k3.e4*k3.e5*e1.e2 + 1/2*den(2*k1.k2)*den(2*k3.k4)*
k1.e3*k3.e4*k4.e5*e1.e2 - den(2*k1.k2)*den(2*k3.k4)*k1.e4*k2.e1*k4.e3 *e2.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e4*k2.e5*k4.e3*e1.e2
- 1/2*den( 2*k1.k2)*den(2*k3.k4)*k1.e4*k3.e5*k4.e3*e1.e2 - 1/2*den(2*k1.k2)*den( 2*k3.k4)*k1.e4*k4.e3*k4.e5*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*
k1.e5*k2.k3*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.k4* e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k3.e2*e3.e4
- den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k3.e4*e2.e3 - 1/2*den(2* k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k4.e2*e3.e4 + den(2*k1.k2)*den(2*
k3.k4)*k1.e5*k2.e1*k4.e3*e2.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e5* k2.e3*k3.e4*e1.e2 - den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e4*k4.e3*e1.e2
+ 1/4*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k3.k5*e1.e2*e3.e4 - 1/2*den(2* k1.k2)*den(2*k3.k4)*k1.e5*k3.e4*k5.e3*e1.e2 - 1/4*den(2*k1.k2)*den(2*
k3.k4)*k1.e5*k4.k5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5* k4.e3*k5.e4*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k2.e1*e2.e5*
e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k3.e4*e1.e2*e3.e5 - 1/2* den(2*k1.k2)*den(2*k3.k4)*k2.k3*k4.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*
den(2*k3.k4)*k2.k3*k4.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)* k2.k4*k2.e1*e2.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k3.e4*
e1.e2*e3.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k3.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k4.e3*e1.e2*e4.e5 - 1/2*den(2*
k1.k2)*den(2*k3.k4)*k2.k5*k3.e4*e1.e2*e3.e5 + 1/4*den(2*k1.k2)*den(2* k3.k4)*k2.k5*k3.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k5*
k4.e3*e1.e2*e4.e5 - 1/4*den(2*k1.k2)*den(2*k3.k4)*k2.k5*k4.e5*e1.e2* e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e3*k3.e4*e2.e5 -
den(2* k1.k2)*den(2*k3.k4)*k2.e1*k2.e4*k4.e3*e2.e5 + 1/2*den(2*k1.k2)*den(2* k3.k4)*k2.e1*k2.e5*k3.e2*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k2.e1*
k2.e5*k3.e4*e2.e3 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k4.e2* e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k4.e3*e2.e4
```

•••

ZviBern®

```
A(k1,e1,k2,e2,k3,e3,k4,e4,k5,e5) = + Tr(Ta1,Ta2,Ta3,Ta4,Ta5) * ( 1/2*den(2*k1.k2)*k1.e2*e1.e3*e4.e5 - den(2*k1.k2)*k1.e2*e1.e4*e3.e5
+ 1/2*den(2*k1.k2)*k1.e2*e1.e5*e3.e4 - 1/4*den(2*k1.k2)*k1.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*k1.e4*e1.e2*e3.e5 - 1/4*den(2*k1.k2)*k1.e5*e1.e2*e3.e4
- 1/2*den(2*k1.k2)*k2.e1*e2.e3*e4.e5 + den(2*k1.k2)*k2.e1*e2.e4*e3.e5 - 1/2*den(2*k1.k2)*k2.e1*e2.e5*e3.e4 + 1/4*den(2*k1.k2)*k2.e3*e1.e2*e4.e5
- 1/2*den(2*k1.k2)*k2.e4*e1.e2*e3.e5 + 1/4*den(2*k1.k2)*k2.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k1.e2*e1.e5*e3.e4
- 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k2.e1*e2.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k2.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k3.e4*e1.e2*e3.e5
+ 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k4.e3*e1.e2*e4.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k3*k4.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*
k3.k4)*k1.k4*k1.e2*e1.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k2.e1*e2.e5*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k2.e5*e1.e2*
e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k3.e4*e1.e2*e3.e5 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.k4*k3.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*
```

+ Tr(Ta1,Ta2,Ta3,Ta4,Ta5) * (1/2*den(2*k1.k2)*k1.e2*e1.e3*e4.e5

```
k1.e5*k3.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k3.e4*e1.e3 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k4.e1*e3.e4
- den(2*k1.k2)*den(2*k3.k4)*k1.e2*k1.e5*k4.e3*e1.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.k3*e1.e5*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k2.k4*e1.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e3*k3.e4*e1.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e4*k4.e3*e1.e5 - 1/2*den(2*
k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k3.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k3.e4*e1.e3 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k2.e5*k4.e1*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k2.e5*k4.e3*e1.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.k5*e1.e5*e3.e4 +
den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e1*k3.e4*e3.e5 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k3.e1*k4.e3*e4.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*
k3.e1*k4.e5*e3.e4 - den(2*k … k1.k2)*
den(2*k3.k4)*k1.e2*k3.e4* … *k5.e3
*e1.e5 - den(2*k1.k2)*den(2 … )*den(2*
k3.k4)*k1.e2*k3.e5*k5.e1*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.k5*e1.e5*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e1*k4.e3*e4.e5
+ den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e3*k4.e5*e1.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e3*k5.e1*e4.e5 - den(2*k1.k2)*den(2*k3.k4)*
k1.e2*k4.e3*k5.e4*e1.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e2*k4.e5*k5.e1*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e3*k2.e1*k3.e4*e2.e5
- den(2*k1.k2)*den(2*k3.k4)*k1.e3*k2.e5*k3.e4*e1.e2 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e3*k3.e4*k3.e5*e1.e2 + 1/2*den(2*k1.k2)*den(2*k3.k4)*
k1.e3*k3.e4*k4.e5*e1.e2 - den(2*k1.k2)*den(2*k3.k4)*k1.e4*k2.e1*k4.e3*e2.e5 + den(2*k1.k2)*den(2*k3.k4)*k1.e4*k2.e5*k4.e3*e1.e2
- 1/2*den( 2*k1.k2)*den(2*k3.k4)*k1.e4*k3.e5*k4.e3*e1.e2 - 1/2*den(2*k1.k2)*den( 2*k3.k4)*k1.e4*k4.e3*k4.e5*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*
k1.e5*k2.k3*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.k4*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k3.e2*e3.e4
- den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k3.e4*e2.e3 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e1*k4.e2*e3.e4 + den(2*k1.k2)*den(2*
k3.k4)*k1.e5*k2.e1*k4.e3*e2.e4 + den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e3*k3.e4*e1.e2 - den(2*k1.k2)*den(2*k3.k4)*k1.e5*k2.e4*k4.e3*e1.e2
+ 1/4*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k3.k5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k3.e4*k5.e3*e1.e2 - 1/4*den(2*k1.k2)*den(2*
k3.k4)*k1.e5*k4.k5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k1.e5*k4.e3*k5.e4*e1.e2 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k2.e1*e2.e5*
e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k3.e4*e1.e2*e3.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k3*k4.e3*e1.e2*e4.e5 + 1/2*den(2*k1.k2)*
den(2*k3.k4)*k2.k3*k4.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k2.e1*e2.e5*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k3.e4*
e1.e2*e3.e5 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k3.e5*e1.e2*e3.e4 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k4*k4.e3*e1.e2*e4.e5 - 1/2*den(2*
k1.k2)*den(2*k3.k4)*k2.k5*k3.e4*e1.e2*e3.e5 + 1/4*den(2*k1.k2)*den(2*k3.k4)*k2.k5*k3.e5*e1.e2*e3.e4 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.k5*
k4.e3*e1.e2*e4.e5 - 1/4*den(2*k1.k2)*den(2*k3.k4)*k2.k5*k4.e5*e1.e2*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e3*k3.e4*e2.e5 -
den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e4*k4.e3*e2.e5 + 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k3.e2*e3.e4 - den(2*k1.k2)*den(2*k3.k4)*k2.e1*
k2.e5*k3.e4*e2.e3 - 1/2*den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k4.e2*e3.e4 + den(2*k1.k2)*den(2*k3.k4)*k2.e1*k2.e5*k4.e3*e2.e4
```

•••

Brute force is not an option!

How difficult is it to calculate $|A|^2$ for arbitrary processes?

Solution

- Work always at the amplitude level (not squared)
- Keep track of all the quantum numbers, (momenta, spin and color)
- Organize them in efficient way, by choosing appropriate basis

How difficult is it to calculate |A|^2 for arbitrary processes?

Calculate **helicity amplitudes**, ie amplitudes for gluons and quarks in a definite helicity states. For massless quarks this amounts to condering chirality states:

$$u_\pm(k) = \frac{1}{2}(1 \pm \gamma_5)u(k)$$

External gluons you always think them as attached to a quark-anti-quark pair with a definite (yet arbitrary) polarization vectors:

$$\varepsilon^+_\mu(k;q) = \frac{\langle q^- | \gamma_\mu | k^- \rangle}{\sqrt{2}\,\langle q\,k \rangle}, \qquad \varepsilon^-_\mu(k,q) = \frac{\langle q^+ | \gamma_\mu | k^+ \rangle}{\sqrt{2}\,[k\,q]}$$

It's just a more sophisticated version of the circular polarization. Choosing appropriately the gauge vector, expressions simplify dramatically.

How difficult is it to calculate |A|² for arbitrary processes?

Inspired by the way gauge theories appear as the zero-slope limits of (open) string theories, it has been suggested to decompose the full amplitude as a sum of gauge invariant **subamplitudes** times color coefficients:

$$\mathcal{A}_n(g_1,\ldots,g_n) = g^{n-2} \sum_{\sigma\in S_{n-1}} \mathrm{Tr}(\mathbf{t}^{a_1}\mathbf{t}^{a_{\sigma_2}}\cdots\mathbf{t}^{a_{\sigma_n}}) A_n(1,\sigma_2,\ldots,\sigma_n)$$

where the formula $if^{abc} = \mathrm{Tr}(t^a,[t^b, t^c])$ has been repeatedly used to reduce the f's into traces of lambdas and the Fierz identities to cancel traces of length l<n. Analogously for quarks:

$$\mathcal{A}_n(q_1,g_2,\ldots,g_{n-1},\bar{q}_n) = g^{n-2} \sum_{\sigma\in S_{n-2}} (\mathbf{t}^{a_{\sigma_2}}\cdots\mathbf{t}^{a_{\sigma_{n-1}}})^i_j A_n(1_q,\sigma_2,\ldots,\sigma_{n-2},n_{\bar{q}})$$

The A_n are MUCH simpler objects to calculate, with many less diagrams...

How difficult is it to calculate $|A|^2$ for arbitrary processes?

n	full Amp	partial Amp
4	4	3
5	25	10
6	220	36
7	2485	133
8	34300	501
9	559405	1991
10	10525900	7335
11	224449225	28199
12	5348843500	108280

How difficult is it to calculate |A|^2 for arbitrary processes?

n	full Amp	partial Amp
4	4	3
5	25	10
6	220	36
7	2485	133
8	34300	501
9	559405	1991
10	10525900	7335
11	224449225	28199
12	5348843500	108280

$$(2n)!$$

How difficult is it to calculate $|A|^2$ for arbitrary processes?

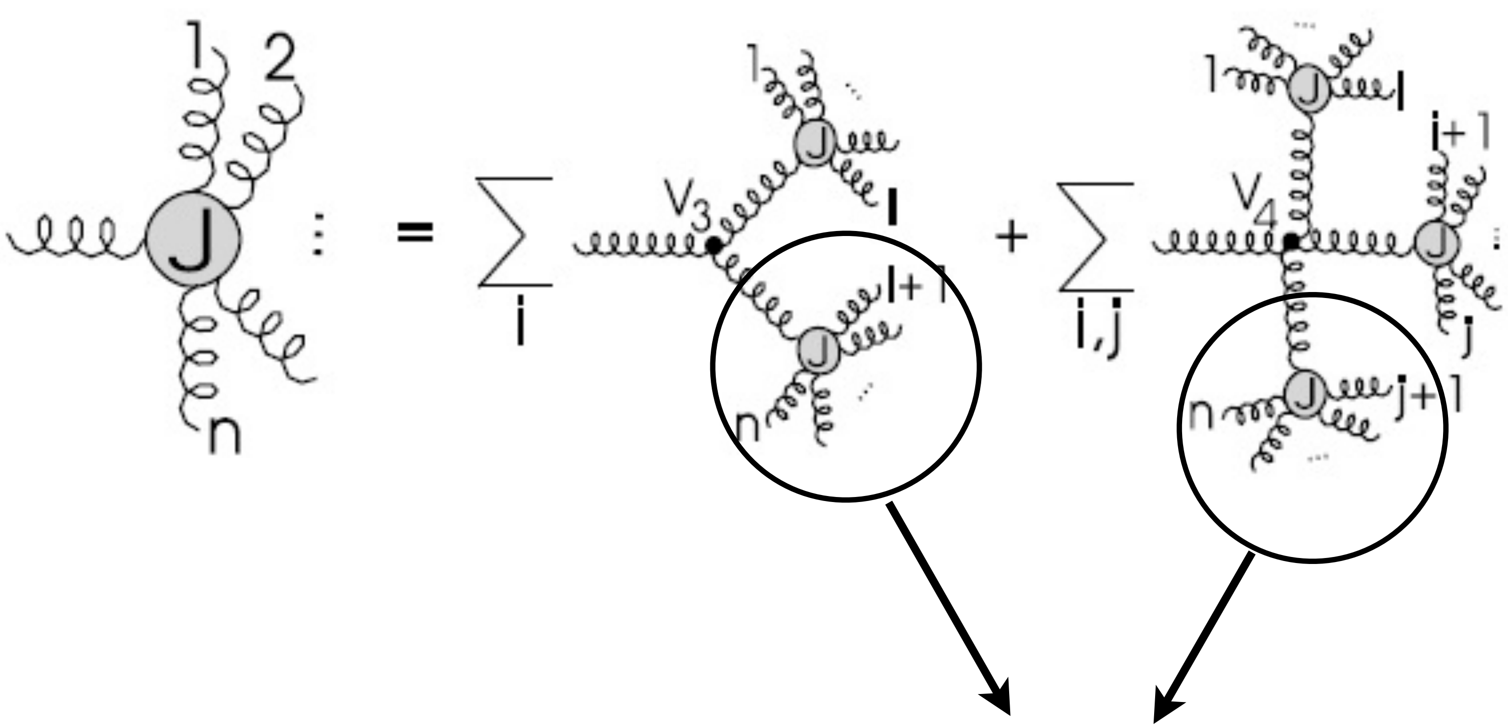
n	full Amp	partial Amp
4	4	3
5	25	10
6	220	36
7	2485	133
8	34300	501
9	559405	1991
10	10525900	7335
11	224449225	28199
12	5348843500	108280

$(2n)!$ 3.8^n

How difficult is it to calculate $|A|^2$ for arbitrary processes?

Feynman diagrams are not efficient because the same subdiagrams are recomputed over and over.
Solution: cash them! In other words use recursive relations.

For the color-ordered subamplitudes for n gluons, such relations (called Berends-Giele) are very easy:



Off-shell amplitudes with max n-1 number of legs !

How difficult is it to calculate $|A|^2$ for arbitrary processes?

n	full Amp	partial Amp	BG
4	4	3	3
5	25	10	10
6	220	36	35
7	2485	133	70
8	34300	501	126
9	559405	1991	210
10	10525900	7335	330
11	224449225	28199	495
12	5348843500	108280	715
	$(2n)!$	3.8^n	n^4

How difficult is it to calculate $|A|^2$ for arbitrary processes?

n	full Amp	partial Amp	BG
4	4	3	3
5	25	10	10
6	220	36	35
7	2485	133	70
8	34300	501	126
9	559405	1991	210
10	10525900	7335	330
11	224449225	28199	495
12	5348843500	108280	715
	$(2n)!$	3.8^n	n^4

The factorial growth is tamed to a polynomial one!

How difficult is it to calculate $|A|^2$ for arbitrary processes?

n	full Amp	partial Amp	BG
4	4	3	3
5	25	10	10
6	220	36	35
7	2485	133	70
8	34300	501	126
9	559405	1991	210
10	10525900	7335	330
11	224449225	28199	495
12	5348843500	108280	715
	$(2n)!$	3.8^n	n^4

The factorial growth is tamed to a polynomial one!

Note, however, one still needs to sum over color!!

LO predictions : final remarks

$$\sigma_X = \sum_{a,b} \int_0^1 dx_1 dx_2\, f_a(x_1,\mu_F^2) f_b(x_2,\mu_F^2) \times \hat{\sigma}_{ab\to X}(x_1,x_2,\alpha_S(\mu_R^2),\frac{Q^2}{\mu_F^2},\frac{Q^2}{\mu_R^2})$$

- By calculating the short distance coefficient at tree-level we obtain the first estimate of rates for inclusive final states.

- Even at LO **extra radiation is included:** it is described by the PDF's in the initial state and by the definition of a final state parton, which at LO represents all possible final state evolutions.

- Due to the above approximations a cross section at LO can strongly depend on the factorization and renormalization scales.

- Predictions can be systematically improved, at NLO and NNLO, by including higher order corrections in the short distance and in the evolution of the PDF's.

How do improve?

1. We reach NLO and NNLO accuracy — TH-Accurate
2. We include parton showers — EXP-Useful

How do improve?

1. We reach NLO and NNLO accuracy — TH-Accurate
2. We include parton showers — EXP-Useful

Predictions at NLO

$$\sigma_X = \sum_{a,b} \int_0^1 dx_1 dx_2\, f_a(x_1, \mu_F^2) f_b(x_2, \mu_F^2) \times \hat{\sigma}_{ab\to X}(x_1, x_2, \alpha_S(\mu_R^2), \frac{Q^2}{\mu_F^2}, \frac{Q^2}{\mu_R^2})$$

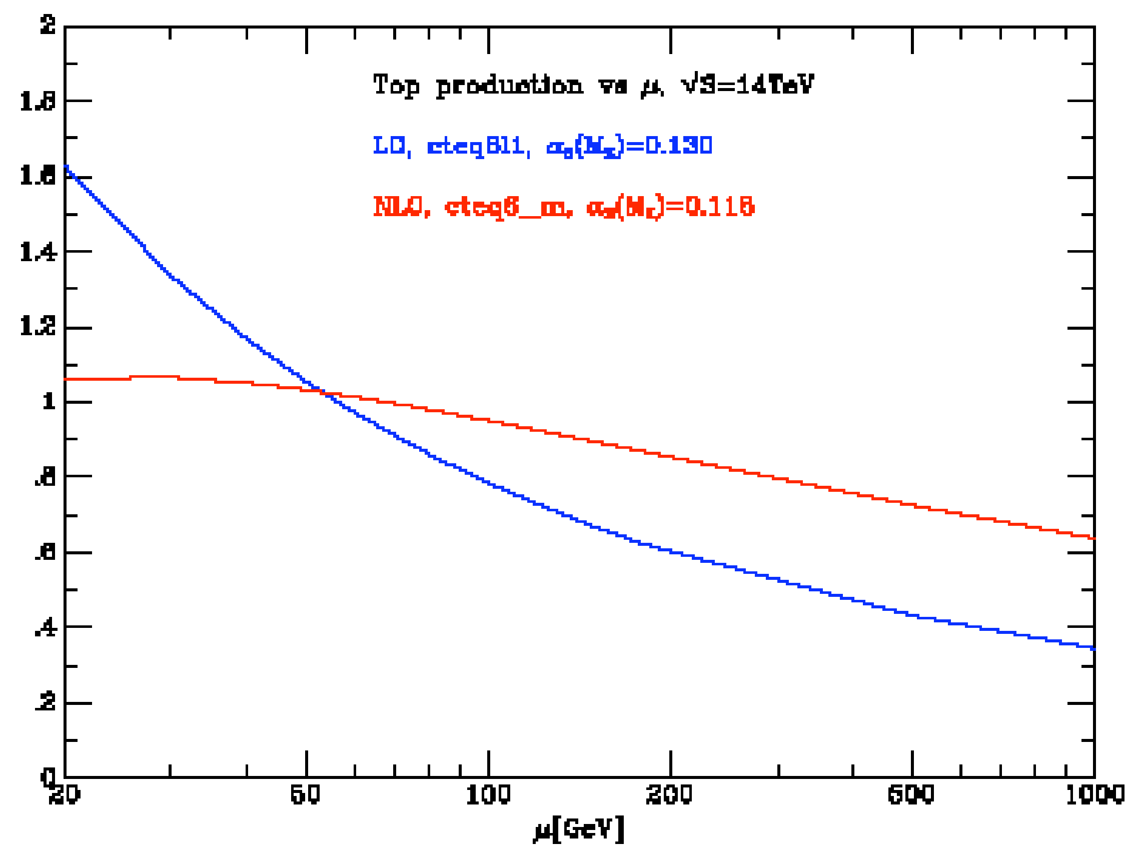
$$\hat{\sigma}_{ab\to X} = \sigma_0 + \alpha_S \sigma_1 + \alpha_S^2 \sigma_2 + \dots$$

Why?

1. First order where scale dependences are compensated by the running of α_S and the evolution of the PDF's: FIRST RELIABLE ESTIMATE OF THE TOTAL CROSS SECTION.

2. The impact of extra radiation is included. For example, jets now have a structure.

3. New effects coming up from higher order terms (e.g., opening up of new production channels or phase space dimensions) can be evaluated.



Predictions at NNLO

$$\sigma_X = \sum_{a,b} \int_0^1 dx_1 dx_2 \, f_a(x_1, \mu_F^2) f_b(x_2, \mu_F^2) \times \hat{\sigma}_{ab\to X}(x_1, x_2, \alpha_S(\mu_R^2), \frac{Q^2}{\mu_F^2}, \frac{Q^2}{\mu_R^2})$$

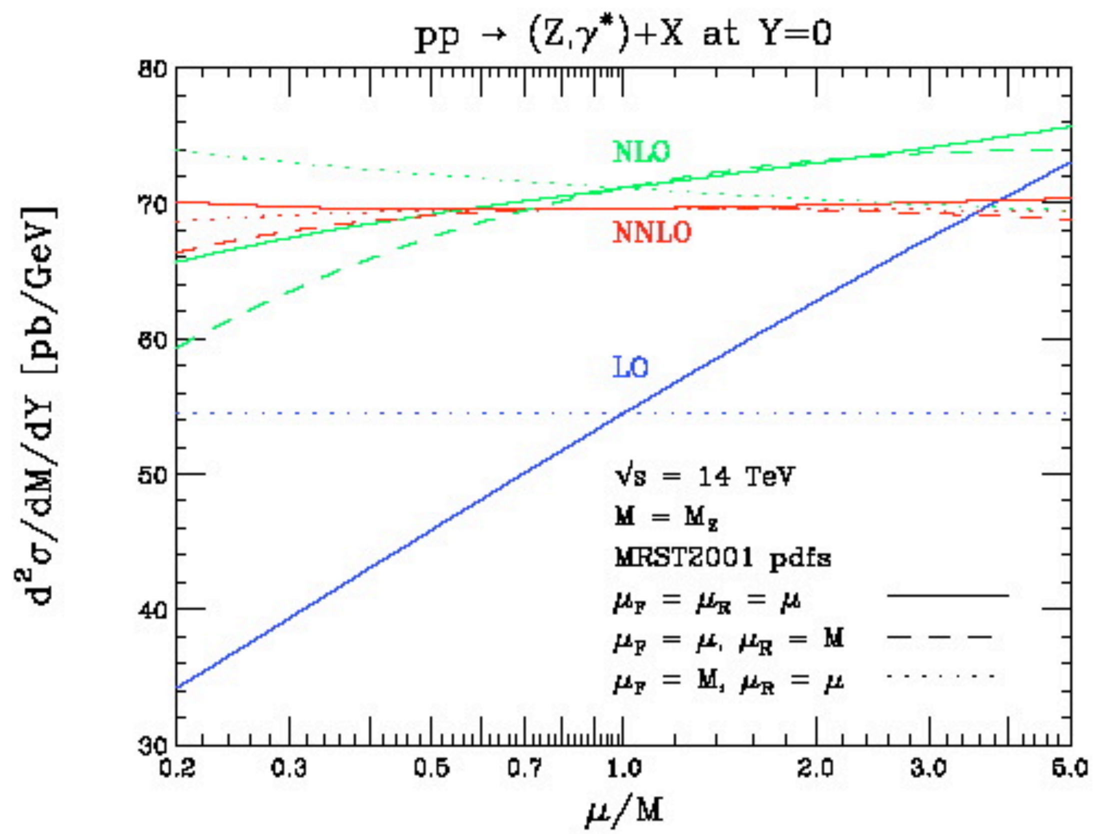
$$\hat{\sigma}_{ab\to X} = \sigma_0 + \alpha_S \sigma_1 + \alpha_S^2 \sigma_2 + \ldots$$

Why?

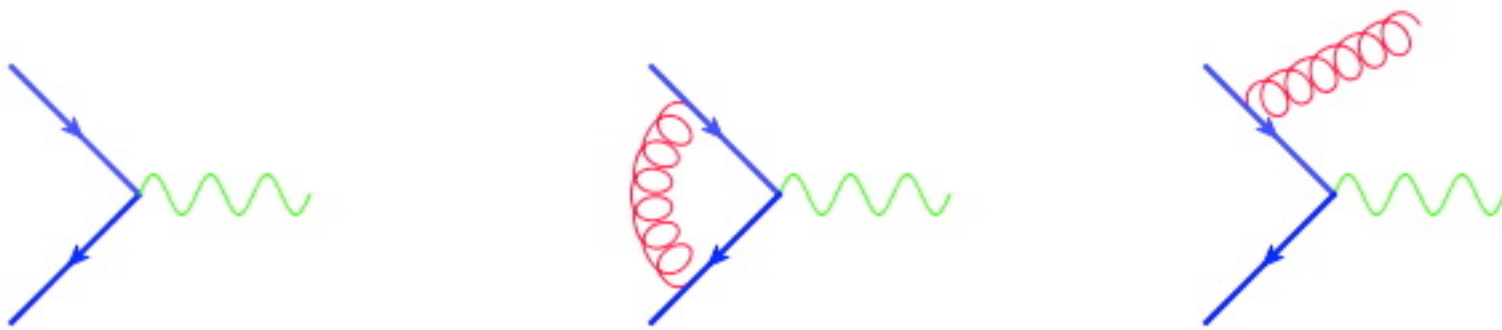
- A NNLO computation gives control on the uncertainties of a perturbative calculation.

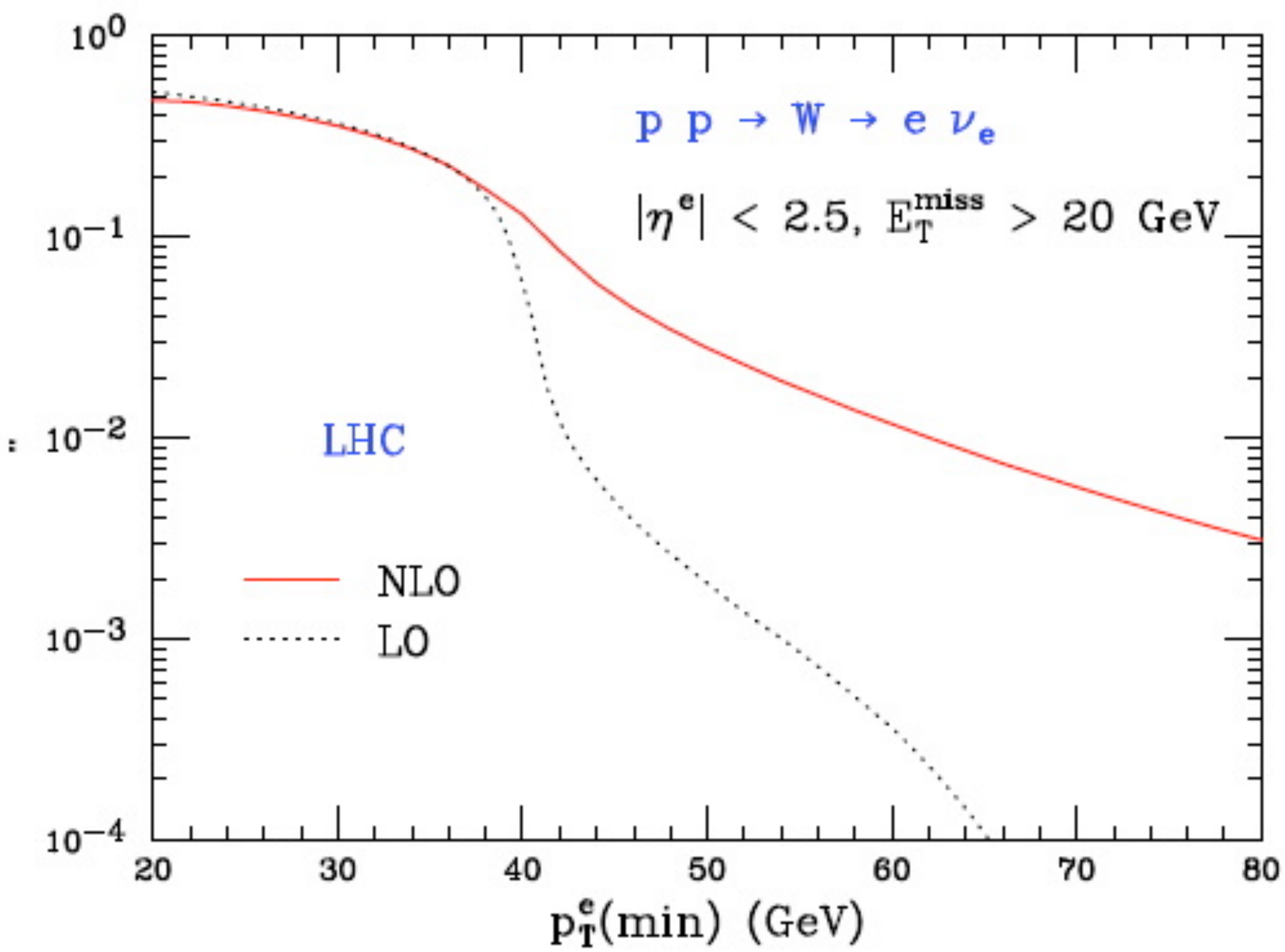
- It's "mandatory" if NLO corrections are very large to check the behaviour of the perturbative series

- It's the best we have! It is needed for Standard Candles and for really exploiting all the available information, for example that of NNLO PDF's.



Drell-Yan predictions at NLO



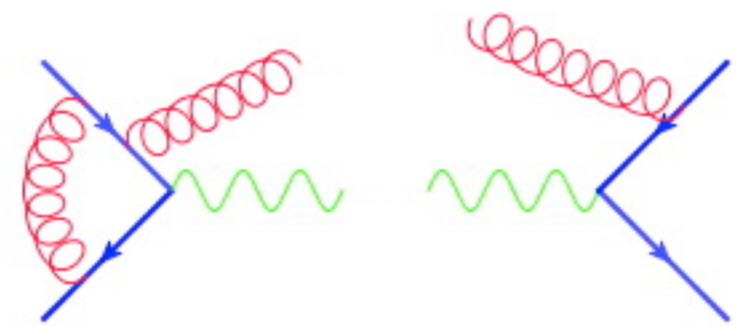


- At LO the W has no p_T, therefore the pt of the lepton has a sharp cutoff.
- The "K-factor" looks like enormous at high p_T. When this happens it means that the observable you are looking at it is actually at LO not at NLO!
- It is important to keep the spin correlations of the lepton in the calculation.

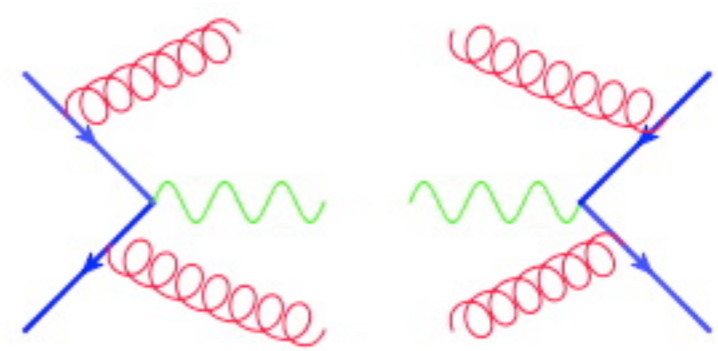
Drell-Yan predictions at NNLO

- Virtual-Virtual : O(100) terms

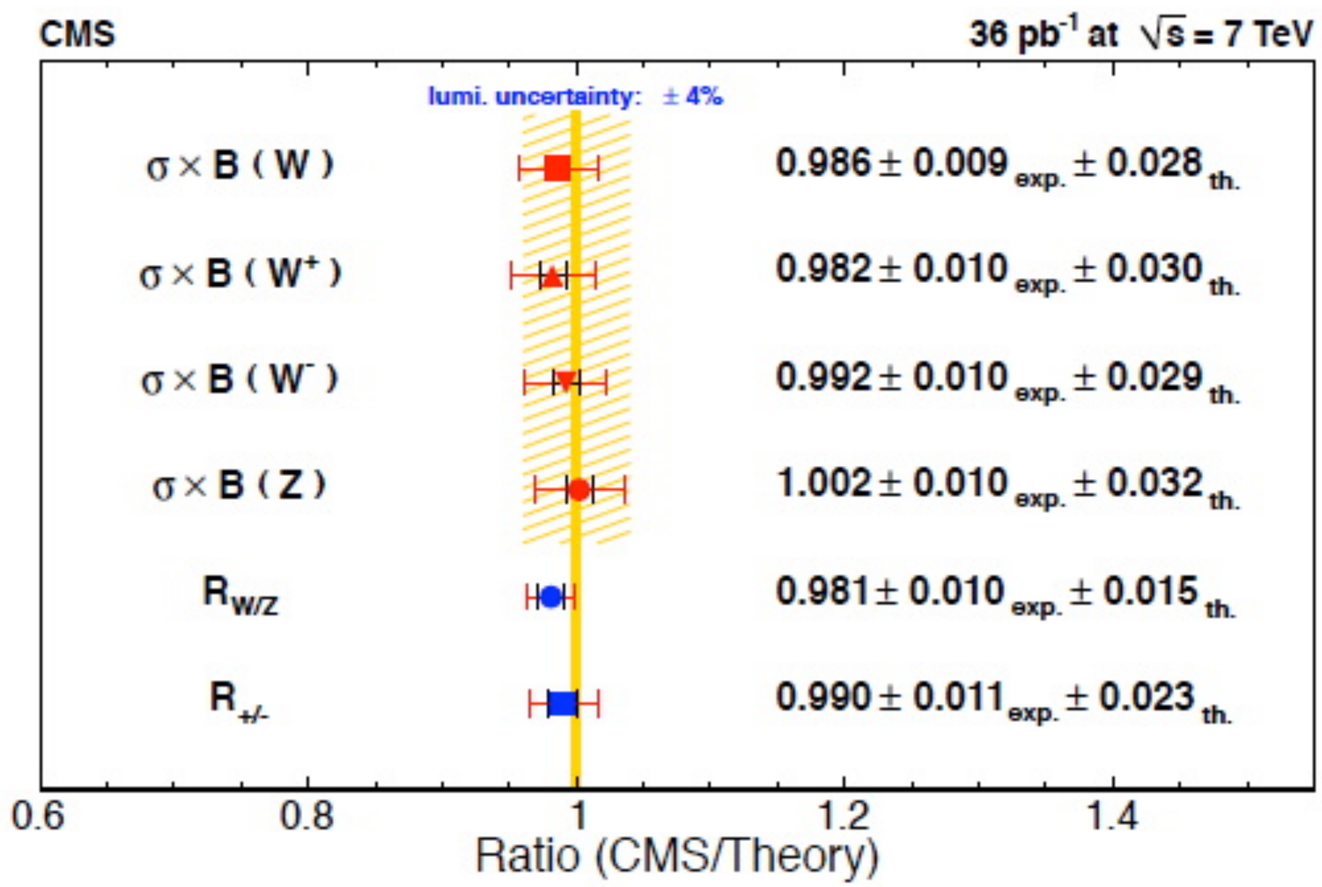
- Real-Virtual : O(300) terms



- Real-Real : O(500) terms



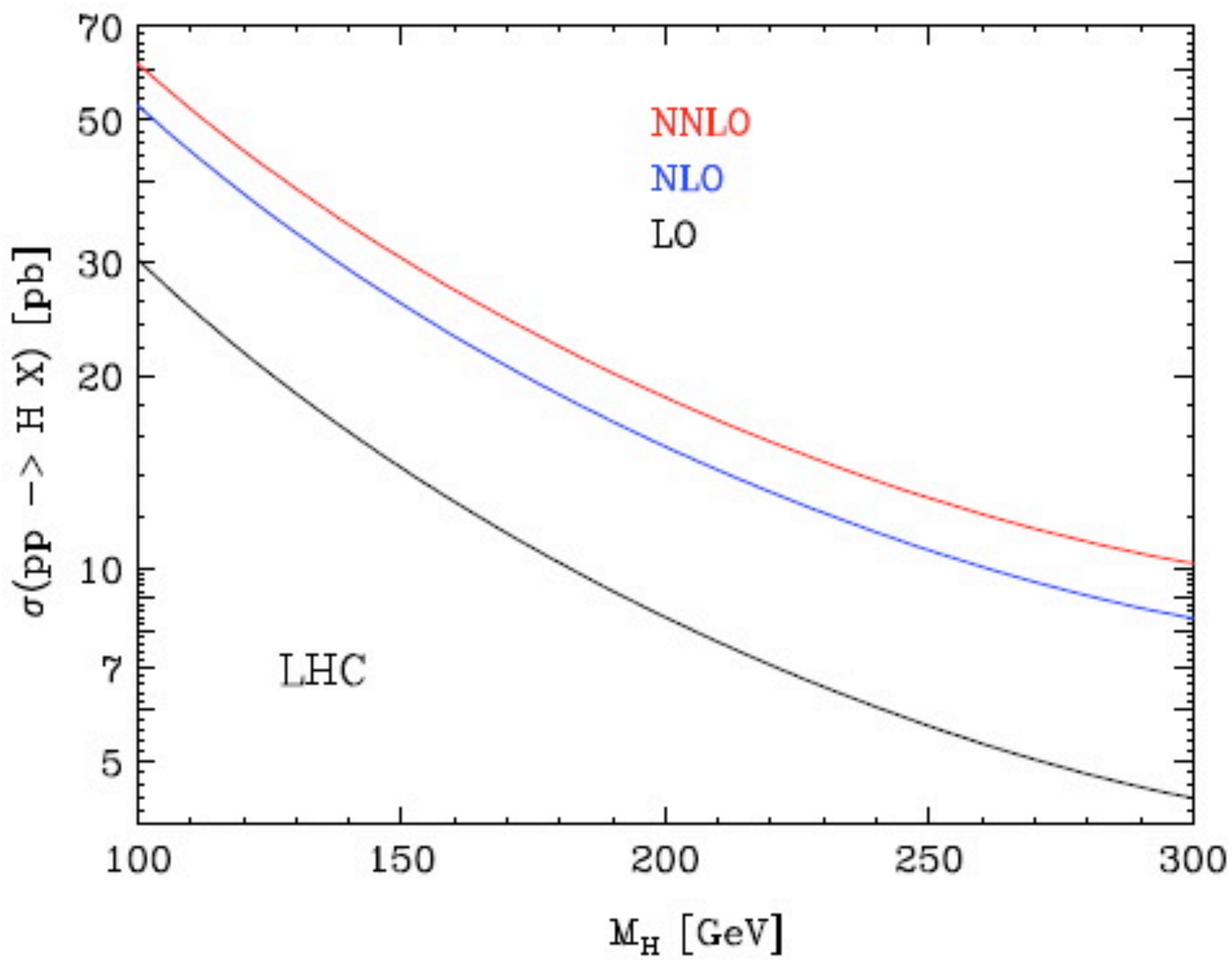
Drell-Yan predictions at NNLO

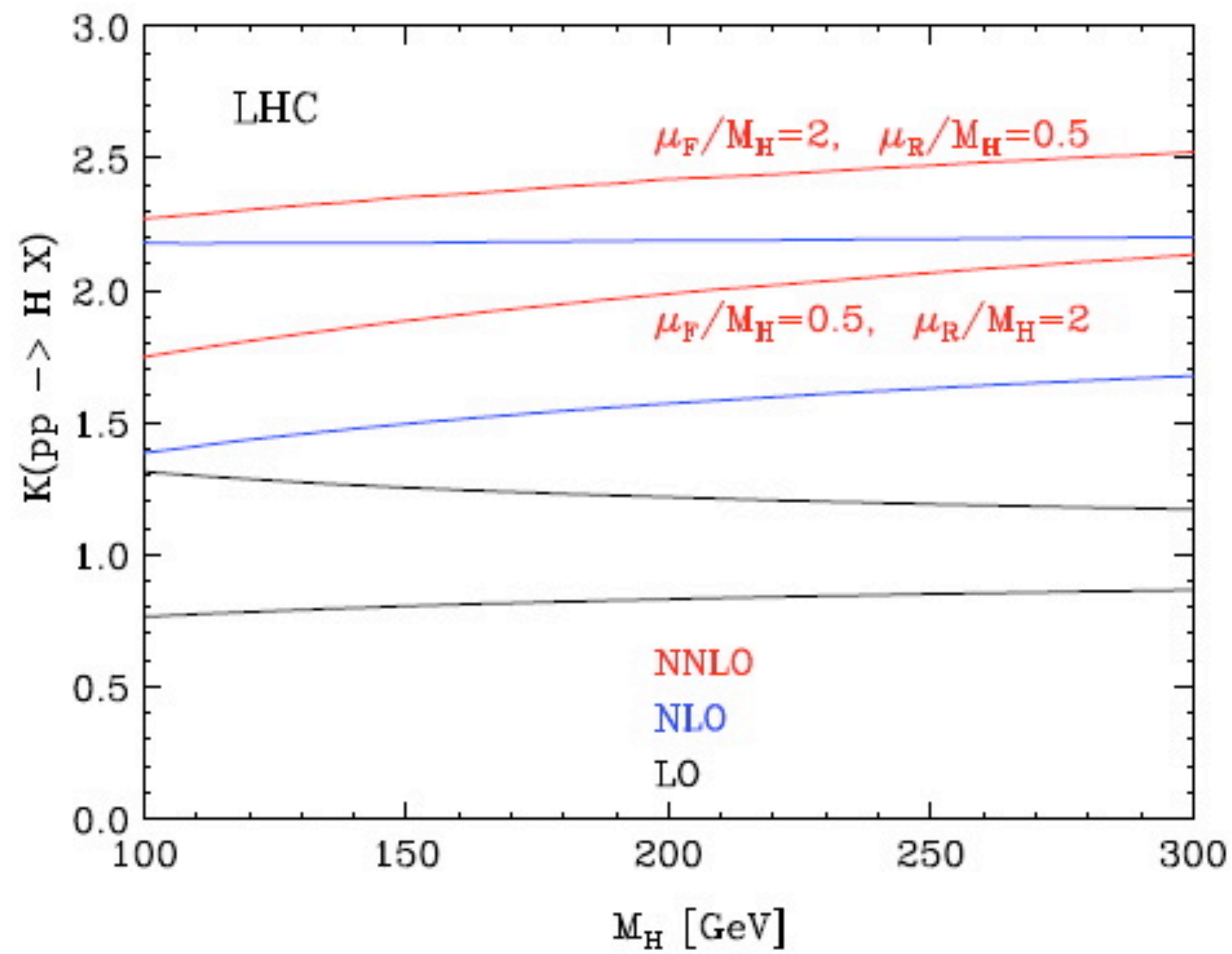


[TH = Anastasiou, Dixon, Melnikov, Petriello. 2004]

- Impressive improvement of the scale dependence.
- High-p_T end of the electron and extra jet known at NLO accuracy

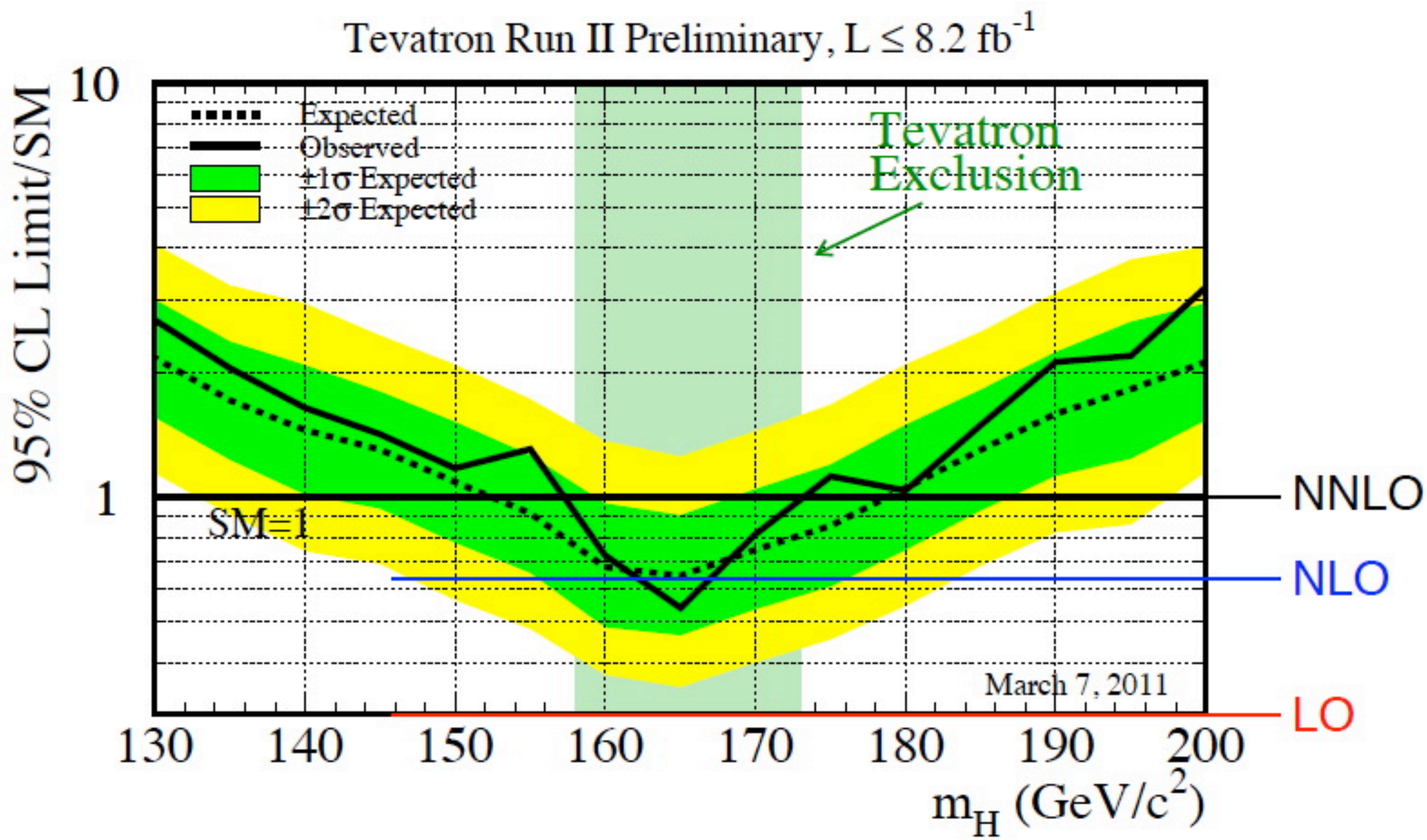
Higgs predictions at NNLO





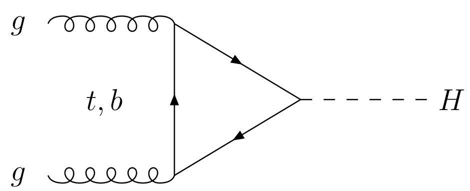
- The perturbative series stabilizes.
- NLO estimation of higher orders effects by scale uncertainty works reasonably well

Higgs predictions at NNLO

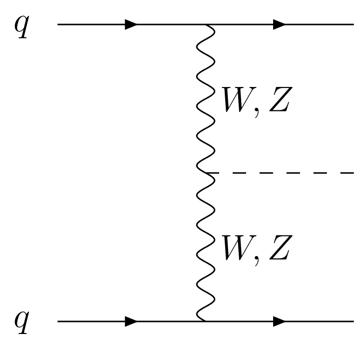


be careful : just illustrative example, not very precise

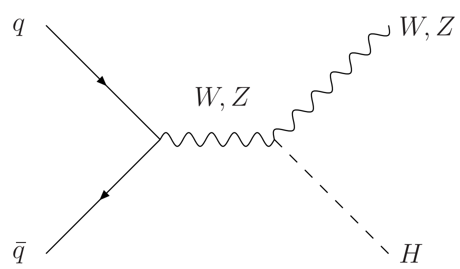
Higgs predictions at 7 TeV



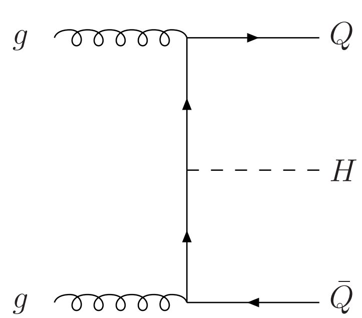
GluonFusion



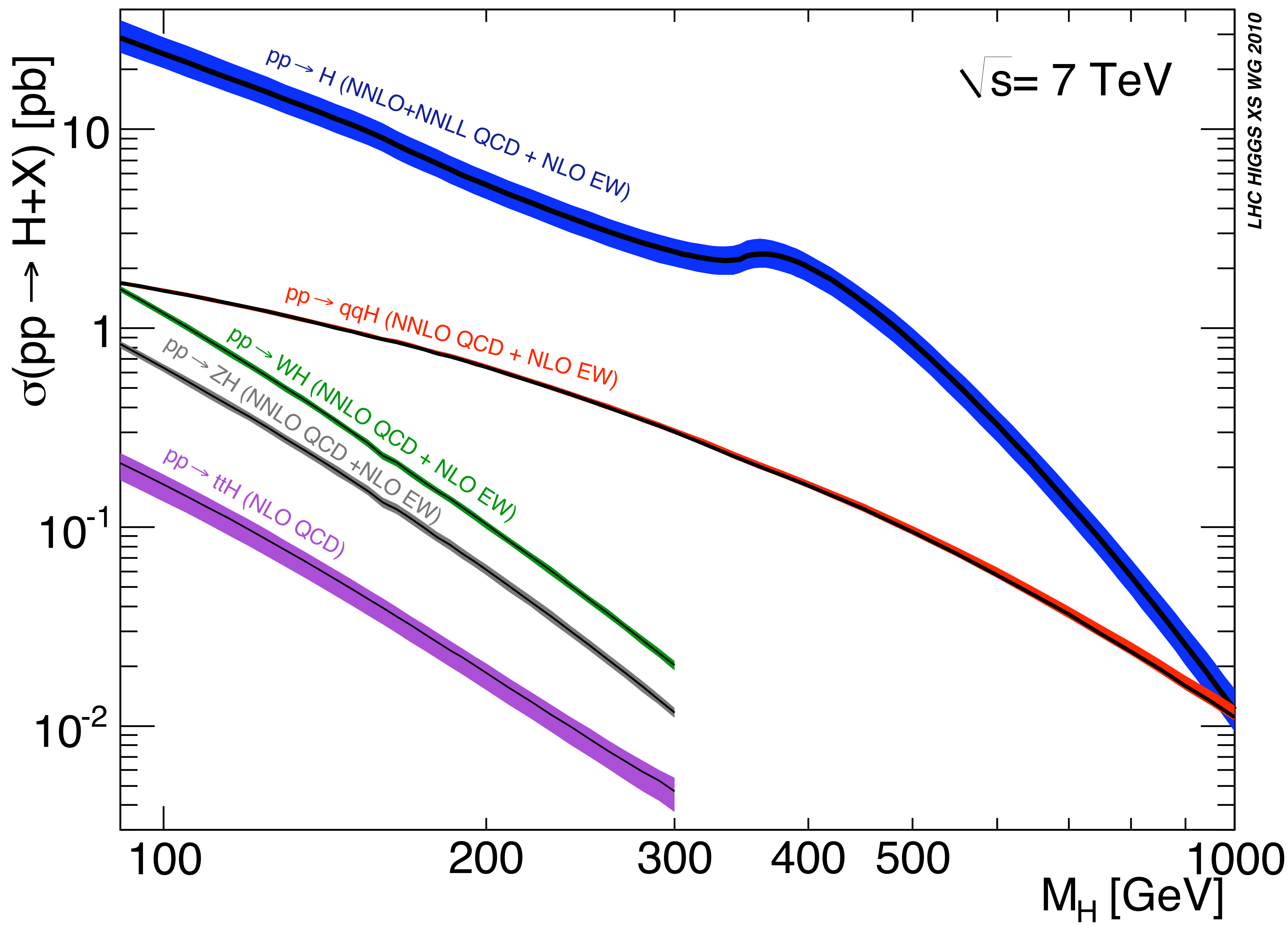
vector boson fusion (VBF)



associated production with vector bosons



associated production with heavy quarks



Predictions at NNLO : final remarks

- Handful of precious predictions at NNLO now available for Higgs and Drell-Yan processes at the parton level for distributions.

- Others (VV, ttbar) in progress and in sight.

NNLO stays to the LHC era
as
NLO stayed to the Tevatron era