Updates for KaTie

Andreas van Hameren
Institute of Nuclear Physics
Polish Academy of Sciences
Kraków

presented at the

MCEGs for future ep and eA facilities

21-02-2019, DESY, Hamburg

Contents

- What does KaTie do?
- Off-shell amplitudes
- QED gauge invariance
- How to use KaTie

What does KaTie do?

Let $Y = \{y = y_1y_2 \to y_3y_4\cdots y_n\}$ be a list of partonic processes contributing to a hadron-scattering process with a multi-jet final state, with differential cross section

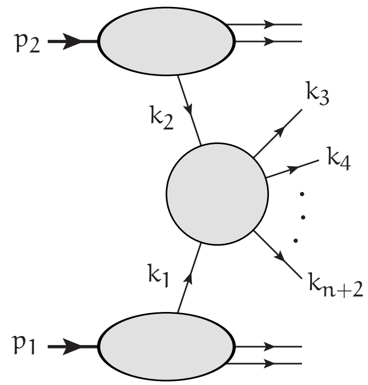
$$d\sigma_Y(p_1,p_2;k_3,\ldots,k_{2+n}) = \sum_{y\in Y}\int d^4k_1\,\mathcal{P}_{y_1}(k_1)\int d^4k_2\,\mathcal{P}_{y_2}(k_2)\,d\hat{\sigma}_y(k_1,k_2;k_3,\ldots,k_{2+n})$$

Collinear factorization:

$$\mathcal{P}_{y_i}(k_i) = \int\frac{dx_i}{x_i}\,f_{y_i}(x_i,\mu)\,\delta^4(k_i - x_ip_i)$$

k_T-dependent factorization factorization:

$$\mathcal{P}_{y_i}(k_i) = \int\frac{d^2\mathbf{k}_{iT}}{\pi}\int\frac{dx_i}{x_i}\,\mathcal{F}_{y_i}(x_i,|\mathbf{k}_{iT}|,\mu)\,\delta^4(k_i - x_ip_i - k_{iT})$$



Differential partonic cross section:

$$d\hat{\sigma}_y(k_1,k_2;k_3,\ldots,k_{2+n}) = d\Phi_Y(k_1,k_2;k_3,\ldots,k_{2+n})\,\Theta_Y(k_3,\ldots,k_{2+n})$$
$$\times\,\mathrm{flux}(k_1,k_2)\times\mathcal{S}_y\,|\mathcal{M}_y(k_1,\ldots,k_{2+n})|^2$$

KaTie creates tree-level event files corresponding to $d\sigma_Y$, if supplied with f_y and/or $\mathcal{F}_y$.

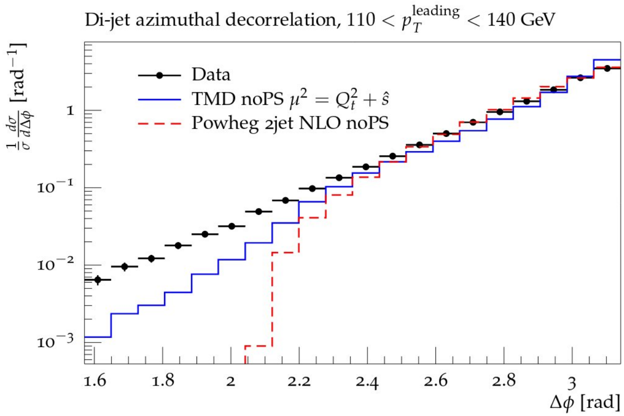
Dijet azimuthal de-correlation

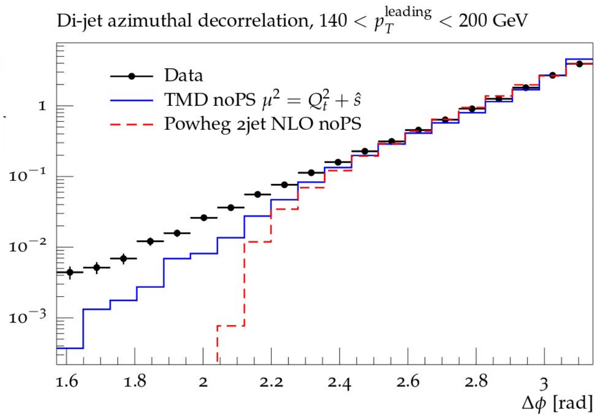
Bury, AvH, Jung, Kutak, Sapeta, Serino 2017

The azimuthal de-correlations, that is the distribution of the angle in the transverse plane between the two hardest jets, for $pp \to jj$ at 7 TeV (data: CMS 2011).

This observable has no distribution at LO (tree-level) in collinear factorization.

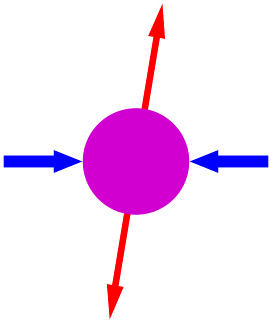
Red prediction: collinear factorization at NLO
Blue prediction: k_T-dependent factorization at tree-level



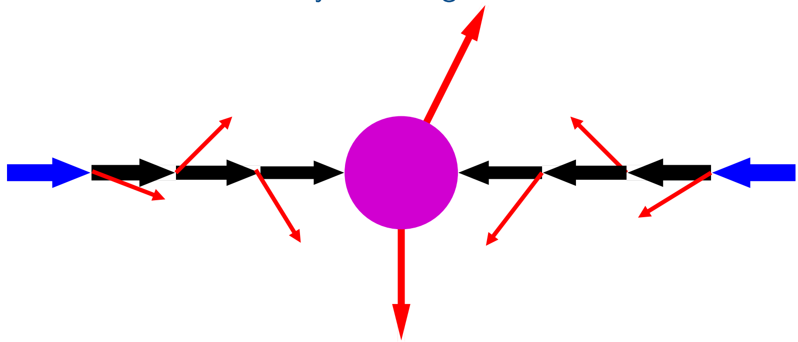


k_T and parton shower

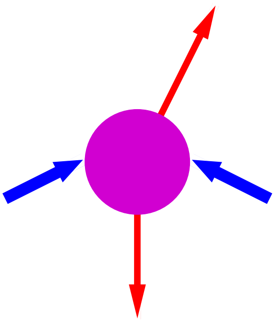
collinear hard scattering

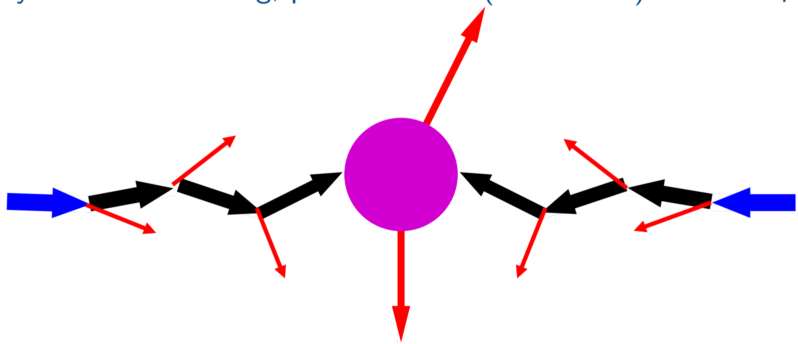


momentum inbalance by enforcing momentum conservation



momentum inbalance already at hard scattering, parton shower (CASCADE) unfolds k_T

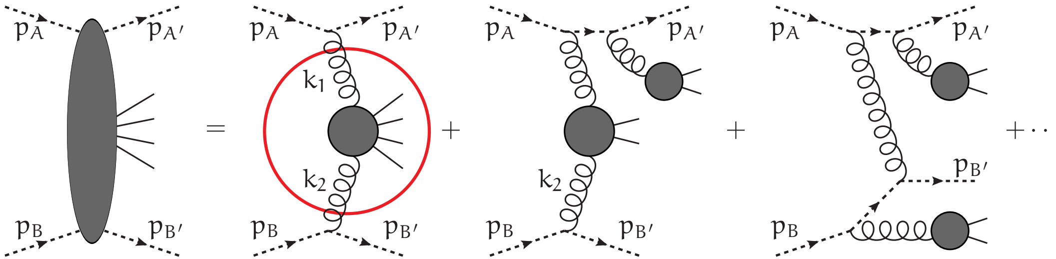




Amplitude as embedding

AvH, Kutak, Kotko 2013

Embed the process in an on-shell process with auxiliary partons and eikonal Feynman rules.



$$p_A^\mu = \Lambda p_1^\mu - \frac{\kappa_1^*}{2}\,\varepsilon_1^{*\mu}$$

$$p_{A'}^\mu = -(\Lambda - x_1)p_1^\mu - \frac{\kappa_1}{2}\,\varepsilon_1^\mu$$

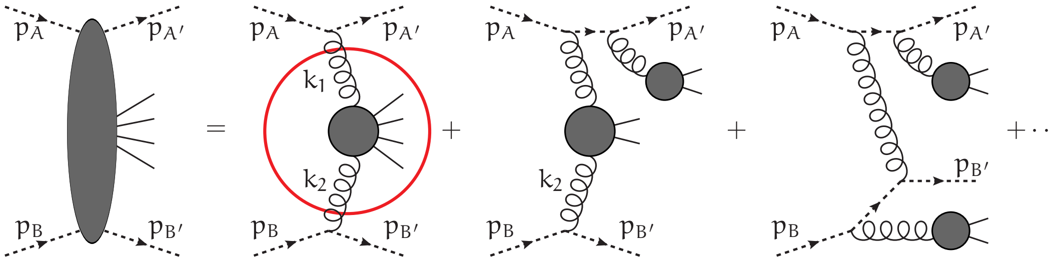
$$p_A^2 = p_{A'}^2 = 0 \qquad k_{1T}^\mu = -\frac{\kappa_1}{2}\,\varepsilon_1^\mu - \frac{\kappa_1^*}{2}\,\varepsilon_1^{*\mu}$$

$$p_A^\mu + p_{A'}^\mu = x_1 p_1^\mu - \frac{\kappa_1}{2}\,\varepsilon_1^\mu - \frac{\kappa_1^*}{2}\,\varepsilon_1^{*\mu} = k_1^\mu$$

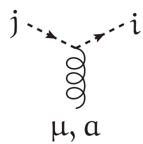
Amplitude as embedding

AvH, Kutak, Kotko 2013

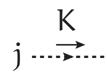
Embed the process in an on-shell process with auxiliary partons and eikonal Feynman rules.



$$\left.\begin{aligned} p_A^\mu &= \Lambda p_1^\mu - \frac{\kappa_1^*}{2}\,\varepsilon_1^{*\mu} \\ p_{A'}^\mu &= -(\Lambda - x_1)p_1^\mu - \frac{\kappa_1}{2}\,\varepsilon_1^\mu \\ \Lambda &\to \infty \end{aligned}\right\} \Rightarrow$$



$$= -\mathrm{i}\, T^a_{i,j}\, p_1^\mu$$

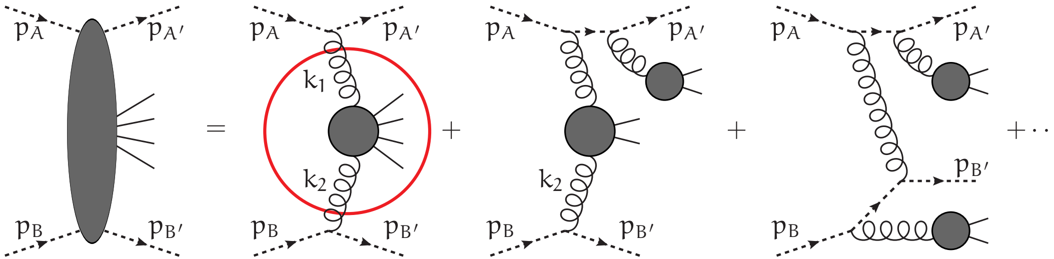


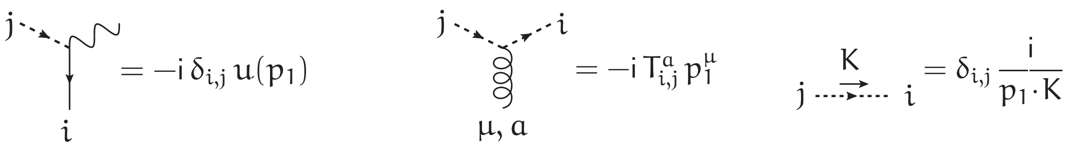
$$= \delta_{i,j}\,\frac{\mathrm{i}}{p_1\cdot K}$$

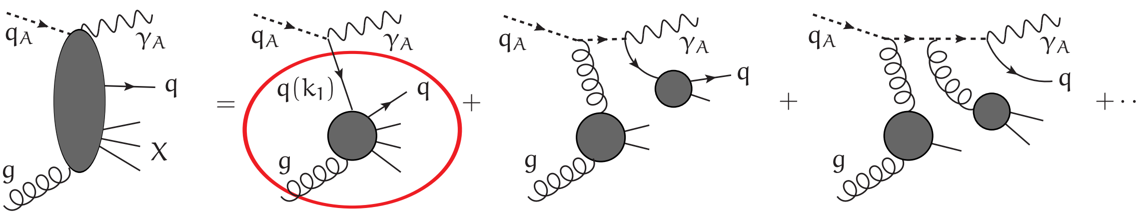
Amplitude as embedding

AvH, Kutak, Kotko 2013
AvH, Kutak, Salwa 2013

Embed the process in an on-shell process with auxiliary partons and eikonal Feynman rules.



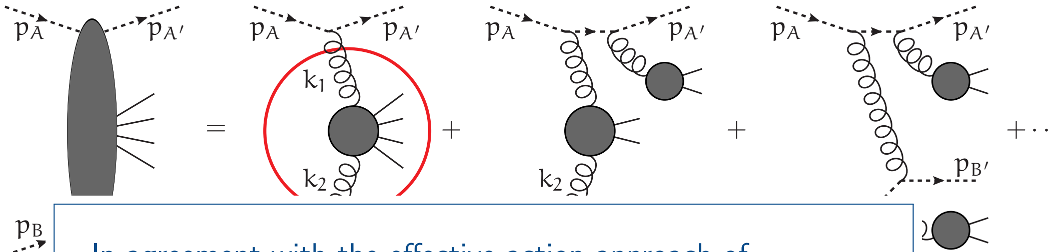




Amplitude as embedding

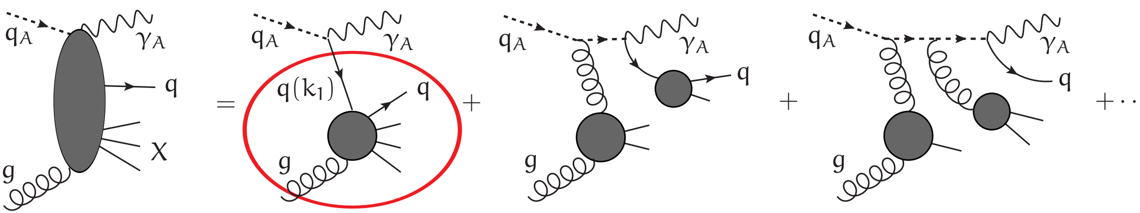
AvH, Kutak, Kotko 2013
AvH, Kutak, Salwa 2013

Embed the process in an on-shell process with auxiliary partons and eikonal Feynman rules.



In agreement with the effective action approach of
Lipatov 1995, Antonov, Lipatov, Kuraev, Cherednikov 2005
Lipatov, Vyazovsky 2000, Nefedov, Saleev, Shipilova 2013
and the Wilson-line approach of
Kotko 2014

j i $= \delta_{i,j} \frac{i}{p_1 \cdot K}$



QED gauge invariance

Once you have *both* off-shell intial-state quarks *and* electro-weak interactions involved in the process, QED gauge invariance becomes an issue. Nefedov, Saleev 2018, Fadin, Sherman 1976.

QED gauge invariance

Once you have *both* off-shell intial-state quarks *and* electro-weak interactions involved in the process, QED gauge invariance becomes an issue. Nefedov, Saleev 2018, Fadin, Sherman 1976.

- to ensure QCD gauge invariance with an off-shell gluon, an auxiliary quark-antiquark pair $q_A, \bar{q}_A$ must be included, plus the interaction vertex

$$q_A \, \bar{q}_A \, g$$

QED gauge invariance

Once you have *both* off-shell intial-state quarks *and* electro-weak interactions involved in the process, QED gauge invariance becomes an issue. Nefedov, Saleev 2018, Fadin, Sherman 1976.

- to ensure QCD gauge invariance with an off-shell gluon, an auxiliary quark-antiquark pair $q_A, \bar{q}_A$ must be included, plus the interaction vertex

$$q_A\, \bar{q}_A\, g$$

- to ensure QCD gauge invariance with an off-shell (anti)quark, an auxiliary quark-antiquark pair $u_A, \bar{u}_A$ must be included (for each flavor), plus an auxiliary photon γ_A, plus interaction vertices

$$u_A\, \gamma_A\, \bar{u} \quad , \quad \bar{u}_A\, \gamma_A\, u \quad , \quad u_A\, \bar{u}_A\, g$$

QED gauge invariance

Once you have *both* off-shell intial-state quarks *and* electro-weak interactions involved in the process, QED gauge invariance becomes an issue. Nefedov, Saleev 2018, Fadin, Sherman 1976.

- to ensure QCD gauge invariance with an off-shell gluon, an auxiliary quark-antiquark pair $q_A, \bar{q}_A$ must be included, plus the interaction vertex

$$q_A\,\bar{q}_A\,g$$

- to ensure QCD gauge invariance with an off-shell (anti)quark, an auxiliary quark-antiquark pair $u_A, \bar{u}_A$ must be included (for each flavor), plus an auxiliary photon γ_A, plus interaction vertices

$$u_A\,\gamma_A\,\bar{u} \quad , \quad \bar{u}_A\,\gamma_A\,u \quad , \quad u_A\,\bar{u}_A\,g$$

- to ensure QED gauge invariance wuth an off-shell (anti)quark, the following vertices must also be included (for each family)

$$u_A\,\bar{u}_A\,\gamma \;\;,\;\; u_A\,\bar{u}_A\,Z \;\;,\;\; d_A\,\bar{d}_A\,\gamma \;\;,\;\; d_A\,\bar{d}_A\,Z \;\;,\;\; u_A\,\bar{d}_A\,W^- \;\;,\;\; \bar{u}_A\,d_A\,W^+$$

QED gauge invariance

Once you have *both* off-shell intial-state quarks *and* electro-weak interactions involved in the process, QED gauge invariance becomes an issue. Nefedov, Saleev 2018, Fadin, Sherman 1976.

- to ensure QCD gauge invariance with an off-shell gluon, an auxiliary quark-antiquark pair $q_A, \bar{q}_A$ must be included, plus the interaction vertex

$$q_A\, \bar{q}_A\, g$$

- to ensure QCD gauge invariance with an off-shell (anti)quark, an auxiliary quark-antiquark pair $u_A, \bar{u}_A$ must be included (for each flavor), plus an auxiliary photon γ_A, plus interaction vertices

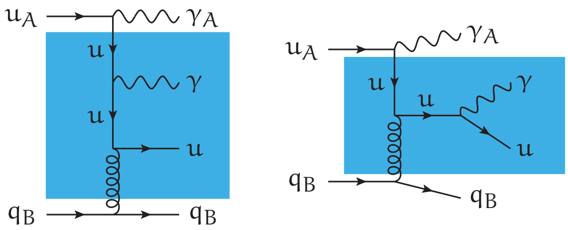
$$u_A\, \gamma_A \bar{u} \quad , \quad \bar{u}_A\, \gamma_A\, u \quad , \quad u_A\, \bar{u}_A\, g$$

- to ensure QED gauge invariance wuth an off-shell (anti)quark, the following vertices must also be included (for each family)

$$u_A\, \bar{u}_A\, \gamma \;\; , \;\; u_A\, \bar{u}_A\, Z \;\; , \;\; d_A\, \bar{d}_A\, \gamma \;\; , \;\; d_A\, \bar{d}_A\, Z \;\; , \;\; u_A\, \bar{d}_A\, W^- \;\; , \;\; \bar{u}_A\, d_A\, W^+$$

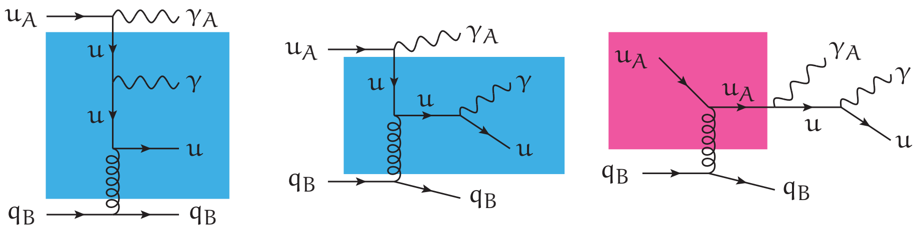
Duplicate auxiliary partons (q_B, u_B, γ_B, etc.) in case of two off-shell.

QED gauge invariance for $u^* g^* \to u\gamma$ for example



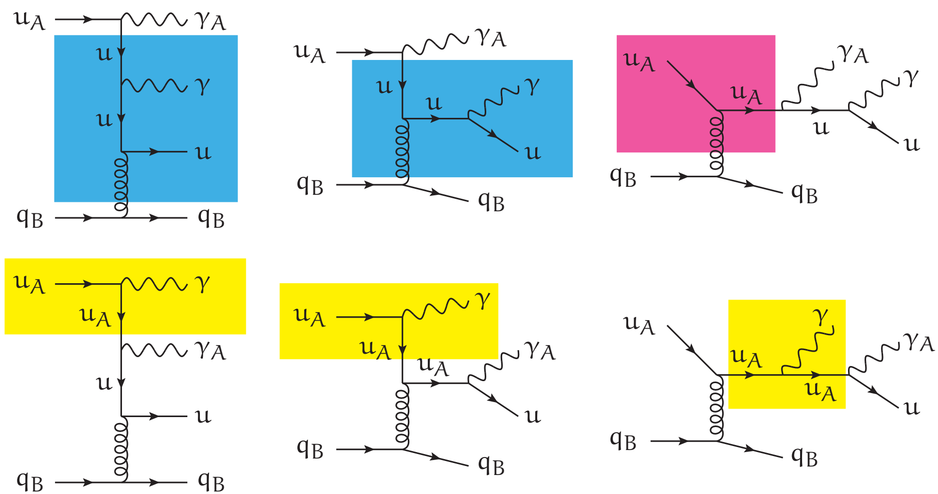
Graphs with the on-shell topology.

QED gauge invariance for $u^* g^* \to u\gamma$ for example



- Graphs with the on-shell topology.
- Extra graph if you allow for the $(u_A \bar{u}_A\, g)$-vertex to ensure QCD gauge invariance.

QED gauge invariance for $u^*g^* \to u\gamma$ for example

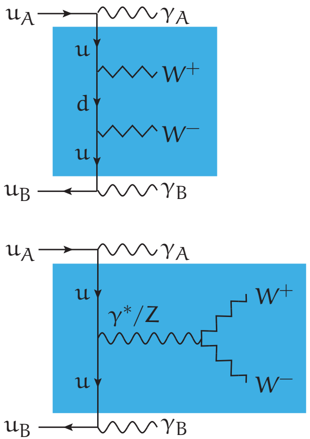


Graphs with the on-shell topology.

Extra graph if you allow for the $(u_A\bar{u}_A\, g)$-vertex to ensure QCD gauge invariance.

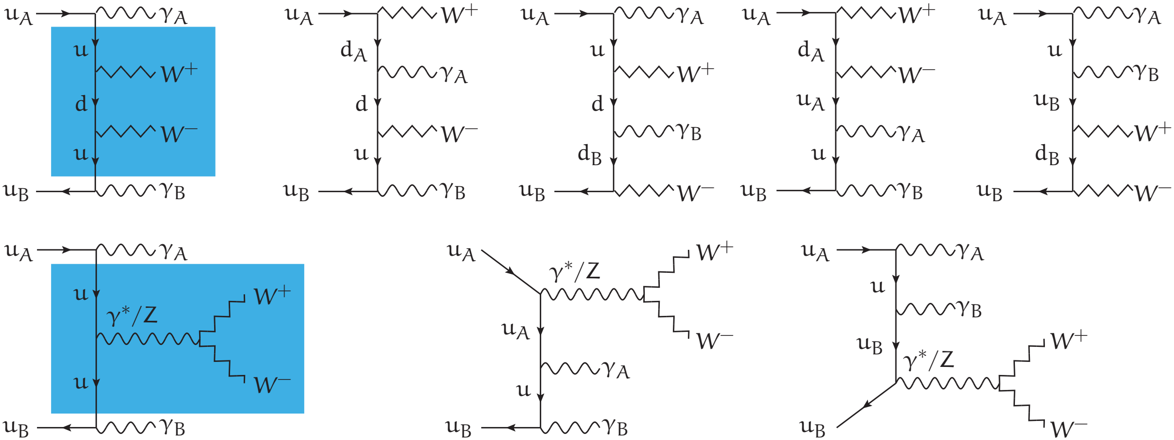
Extra graphs if you allow for the $(u_A\bar{u}_A\gamma)$-vertex to ensure QED gauge invariance.

QED gauge invariance for $u^*u^* \to W^+W^-$ for example



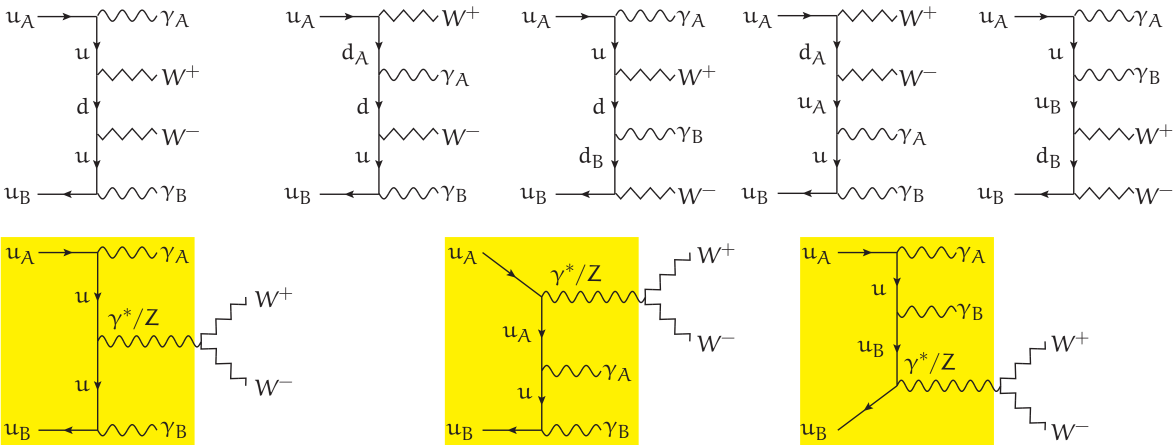
Graphs with the on-shell topology.

QED gauge invariance for $u^* u^* \to W^+ W^-$ for example



Graphs with the on-shell topology.

QED gauge invariance for $u^*u^* \to W^+W^-$ for example



Graphs composing the Fadin-Sherman vertex.

$$(v - a\gamma^5)\left[\gamma^\mu - \frac{p_A^\mu}{p_A \cdot k_B}\not{k}_A - \frac{p_B^\mu}{p_B \cdot k_A}\not{k}_B\right]$$

How to use KaTie

Installation

Download repositories from

`https://bitbucket.org/hameren/katie/downloads`

`https://bitbucket.org/hameren/avhlib/downloads`

and unzip files. Open `settings.py` inside the `KaTie`-directory and set paths:

Path to the `AVHLIB`-directory you just unzipped:

```
AVHLIBpath = '/home/user123/software/hameren-avhlib-xxxx'
```

Path to the directory where `libLHAPDF.so` is:

```
LHAPDFpath = '/usr/lib'
```

Fortran compiler with flags:

```
FC = 'gfortran -fcheck=bounds'
```

If you want to use `TMDlib`, the path to the directory where `libTMDlib.so` is:

```
TMDLIBpath = '/usr/local/lib'
```

If you want to use `TMDlib`, the path to the directory where `libgsl.so` is:

```
GSLpath = '/usr/lib'
```

Now, inside the `KaTie`-directory, execute

```
$ ./config.py lib
```

You can always re-compile with

```
$ ./run.sh clean; ./run.sh lib
```

Input file: processes

The user must explicitly list all desired parton-level processes:

```
process = g g  -> g g g  , factor = 1
process = g g  -> g u u~ , factor = Nf
process = u d~ -> g u d~ , factor = 1
```

The value of `Nf` is set with

```
Nflavors = 4
```

Input file: processes

The user must explicitly list all desired parton-level processes:

```
process = g g  -> g g g  , factor = 1
process = g g  -> g q q~ , factor = Nf
process = q r  -> g q r  , factor = 1
```

The value of `Nf` is set with

```
Nflavors = 4
```

To sum over initial-state partons with identical matrix elements:

```
partlumi = combined
```

Input file: processes

The user must explicitly list all desired parton-level processes:

```
process = g g  -> g g g  , factor = 1
process = g g  -> g u u~ , factor = Nf
process = u d~ -> g u d~ , factor = 1
```

The value of `Nf` is set with

```
Nflavors = 4
```

Interactions can be switched on/off with

```
switch = withQCD yes
switch = withQED yes
switch = withWeak yes
switch = withHiggs no
switch = withHG no
switch = withHA no
```

and the user must set the number of non-QCD vertices in the amplitudes, eg.

```
pNonQCD = 2 0 0  # Electro-Weak  Higgs-gluon  Higgs-photon
```

to have stricktly $\mathcal{O}(\alpha_{\mathsf{EW}}^2\alpha_{\mathsf{S}}^2)$ contributions in

```
process = u u~ -> mu+ mu- d d~
```

Input file: PDFs

`lhapdf`-set (always necessary for the evaluation of α_S):

```
lhaSet = CT10
```

Which initial-state partons should be off-shell?

```
offshell = 1 0  # eg. g* g -> ...
```

`TMDlib`-set:

```
TMDlibSet = PB-TMDNLO-HERAI+II-2018-aspt
```

or alternatively, refer to grid-files explicitly:

```
TMDtableDir = /home/user123/software/tmdlib-1.0.29/data/PB-NLO-2018/
tmdpdf = g  PB-TMDNLO-HERAI+II-2018-aspt_g.dat
tmdpdf = u  PB-TMDNLO-HERAI+II-2018-aspt_u.dat
tmdpdf = d~ PB-TMDNLO-HERAI+II-2018-aspt_dbar.dat
```

If different sets for two off-shell partons are desired (labelling `B A -> 1 2 3 ...`):

```
TMDtableDir = /home/user123/software/tmdlib-1.0.29/data/PB-NLO-2018/
tmdpdf A = g PB-TMDNLO-HERAI+II-2018-aspt_g.dat
TMDtableDir = /home/user123/software/tmdlib-1.0.29/data/nCTEQ15FullNuc_208_82/
tmdpdf B = g PB-aspt_g.dat
```

Input file: kinematics

Beam energies:

```
EbeamA = 7000 # also EbeamPosRap or EbeamHadron
EbeamB = 7000 # also EbeamNegRap or EbeamElectron
```

Inclusive cuts (final-state momenta enumerated as in `B A -> 1 2 3...`):

```
cut = {deltaR|2,3|}
cut = {pT|2|} > 20
cut = {rapidity|2|} > -2.0
cut = {rapidity|2|} <  2.0
```

More complicated cuts are possible, even by providing explicit “almost”-source code:

```
cut source = if ({rapidity|1|}.gt.{rapidity|2|}) then
cut source = if ({pT|2|}.lt.30d0) REJECT
cut source = endif
```

To set all hard scales to the same variable:

```
scale = ({pT|1|}+{pT|2|}+{pT|3|})/2
```

This can be overruled with

```
renormalization scale = ({pT|1|}+{pT|2|}+{pT|3|})/4
scaleA = 91.2d0
scaleB = {pT|1|}+{pT|2|}
```

Deep inelastic scattering

Put processes in input file as (for example for $e^- p \to jj\, e^-$):

```
process = DIS g  -> u u~
process = DIS u  -> g u
process = DIS d~ -> g d~
```

The user must make sure that the number of EW couplings is correct, so at least

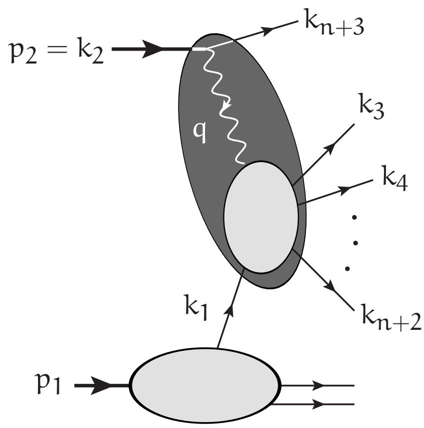
```
pNonQCD = 2 0 0
```

Cuts involving the final-state electron:

```
cut = {energy|electron|} > 11
cut = {theta|electron|} > 30
cut = {plus|1+2+electron|} > 35 # plus=E-pz
```

Other variables typically necessary for DIS:

```
cut = {xBjorken} < 0.5
cut = {Qsquare} > 150
cut = {inelast} > 0.2
cut = {inelast} < 0.7
cut = {deltaRbreit|1,2|} > 1.0
cut = {pTbreit|1|} > 5
cut = {pTbreit|2|} > 5
```



Preparation and optimization

Make the run-script available somehow, eg. via

```
$ ln -s -T ~/bin/KaTie ~/software/hameren-katie-xxxx/run.sh
```

Given an input file `input_example`, a directory `playground` with the necessary compiled programs is created with

```
$ KaTie prepare input_example playground
```

The directory `playground` contains a directory for each process in `input_example` containing a phase space generator that must be optimized:

```
$ cd playground
$ ./optimize.sh Nparallel=4  # run only 4 processes at the same time
```

The optimization process can be monitored with for example

```
$ tail -f proc*/output
```

and should reach a few percent of statistical precision

```
MESSAGE from Kaleu stats: Ntot = 11,563,078
MESSAGE from Kaleu stats: +     100,000 (.25667397+/-.00605831)E+02 2.360%
MESSAGE from Kaleu stats: stopping optimization, re-starting collection
```

If `100,000` accepted phase space points appears to be too few, set, for example,

```
Noptim = 400,000
```

in the input file, and re-prepare. You can also edit `optimize.sh`.

Event generation

After optimization, you can run several instances of the the `main.out` in `playground`:

```
$ nohup ./main.out seed=123401 > output1234501 &
$ nohup ./main.out seed=123402 > output1234502 &
```

will create "proto" event files `raw123401.dat` and `raw123402.dat`. These are turned into a single event file in the LHEF format with

```
$ ./create_eventfile.out lhef raw*
```

The source file `create_eventfile.f90` can be edited, for example for event re-weighting,

Template create_eventfile.f90

```
program create_eventfile !============================ DO NOT TOUCH THIS LINE ==
use katie_eventfile !================================= DO NOT TOUCH THIS LINE ==
use katie_histogramtools !============================ DO NOT TOUCH THIS LINE ==
!== USE OTHER MODULES BELOW THIS LINE ==========================================
use my_pdfs_mod !=================================================== EXAMPLE ==

implicit none !================= REMOVE THIS LINE AND SUFFER THE CONSEQUENCES ==
!== DECLARE NEW VARIABLES BELOW THIS LINE ======================================

!== CHANGE UNIT NUMBERS IN THE LINE BELOW, IF YOU WISH =========================
call initialize( rawFile_unit=21 ,eventFile_unit=22 ) !========================
!== INITIALIZE YOUR ROUTINES BELOW THIS LINE ===================================

do ;if (exitLoop>0) exit ;call read_event !=========== DO NOT TOUCH THIS LINE ==
!== COLLECT DATA AND ALTER EVENTWEIGHT BELOW THIS LINE =========================
eventWeight = eventWeight/pdfA*my_pdf(flavorA,xA,kTA,scaleA) !======= EXAMPLE ==

call write_event ;enddo !============================= DO NOT TOUCH THIS LINE ==
!== DO STUFF AFTER CREATION OF THE EVENT FILE BELOW THIS LINE ==================

end program !========================================= DO NOT TOUCH THIS LINE ==
```

Event generation

After optimization, you can run several instances of the the `main.out` in `playground`:

```
$ nohup ./main.out seed=123401 > output1234501 &
$ nohup ./main.out seed=123402 > output1234502 &
```

will create "proto" event files `raw123401.dat` and `raw123402.dat`. These are turned into a single event file in the LHEF format with

```
$ ./create_eventfile.out lhef raw*
```

The source file `create_eventfile.f90` can be edited, for example for event re-weighting, and compiled with

```
$ bash create_eventfile.sh
```

This script can also be edited, for example to include other external libraries.

Template create_eventfile.sh

```
here=/home/user123/sandbox/playground
katieBuild=/home/user123/software/hameren-katie-xxxx/build
gfortran -fcheck=bounds \
-I $katieBuild \
$here/create_eventfile.f90 \
-L$katieBuild -Wl,-rpath,$katieBuild -lhead \
-L/usr/lib -Wl,-rpath,/usr/lib -lLHAPDF \
-L/usr/local/lib -Wl,-rpath,/usr/local/lib -lTMDlib \
-L/usr/lib -Wl,-rpath,/usr/lib -lgsl -lgslcblas -lm \
-J $katieBuild \
-o $here/create_eventfile.out \
```

Event generation

After optimization, you can run several instances of the the `main.out` in `playground`:

```
$ nohup ./main.out seed=123401 > output1234501 &
$ nohup ./main.out seed=123402 > output1234502 &
```

will create "proto" event files `raw123401.dat` and `raw123402.dat`. These are turned into a single event file in the LHEF format with

```
$ ./create_eventfile.out lhef raw*
```

The source file `create_eventfile.f90` can be edited, for example for event re-weighting, and compiled with

```
$ bash create_eventfile.sh
```

This script can also be edited, for example to include other external libraries.

The executable can take more comma-separated options:

```
$ ./create_eventfile.out lhef,nb,dir=/tmp,name=events_v1.dat,label=_v1 raw*
```

Summary

- KaTie can generate parton-level event files for DIS with an off-shell initial-state parton.
- This required some special attention to QED gauge invariance.