artemide

ver.3.03

Alexey Vladimirov
January 26, 2026

Manual is ever updating.

If you use the artemidev.1.??, please, quote [1].
If you use the artemidev.2.??, please, quote [2].
If you use the artemidev.2.06, please, quote [3].

If you find mistakes, have suggestions or questions, please, write to:

Alexey Vladimirov: *alexeyvl@ucm.es* or *vladimirov.aleksey@gmail.com*

Stable Repository: https://github.com/VladimirovAlexey/artemide-public
Dev Repository: https://github.com/VladimirovAlexey/artemide-development

Contents

Glossary

TMD = Transverse Momentum Dependent
PDF = Parton Distribution Function
FF = Fragmentation Function
NP = Non-perturbative
DY = Drell-Yan process
SIDIS = Semi-Inclusive Deep-Inelastic Scattering
LO = Leading order (in QCD perturbation theory)
NLO = Next-to-Leading order, (in QCD perturbation theory)
N^{2}LO = Next-to-Next-to-Leading order (etc), (in QCD perturbation theory)
LP = Leading power (in power expansion of TMD factorization)
NLP = Next-to-Leading power (etc), (in power expansion of TMD factorization)
KPC = Kinematic power corrections

I. GENERAL STRUCTURE AND USER INPUT OF artemide

A. General concept

The artemide is a package of Fortran modules for calculation in TMD factorization framework. It is has a modular structure, where each module is responsible for evaluation of some theory construct. For instance: a TMD distribution, a TMD evolution factor, cross-section. Each module produces a single function, which is a composite of integrals/products of lower-level functions. Thus, each level of operation can be used as is, in other programs. The highest-level task is the evaluation of cross-section within the TMD factorization theorem, including all needed integrations, and factors, i.e., such that it can be directly compared to the data. It also includes several tools for analysis of the obtained values, such as variation of scales, search for limiting parameters, etc. The theory structure of artemide is discussed in the next section. The dependency structure of modules is presented in fig.1.

Initially the artemide project was created for pure theoretical games. Since the beginning artemide appears to be successful also in the phenomenology, (and nowadays it is mostly used for it). Nonetheless, conceptually artemide is build as the theory playground, and will continue to be developed in this direction. That is why it architecture is not very optimal from the pure numerical point of view. In fact, there are several ways to optimize the code, melting together some modules and structures, but in this case artemide will lose it theoretical cleanness. Also artemide contains a lot of "unpractical" and rare options, and possibility to control each parameter. The positive side is the possibility to easily implement the new theory founding and check them, which is regularly done.

Wide spectrum of application of artemide code makes it difficult to create a convenient interface. Moreover, at the current stage of development, I prioritize the quality of computation, to the user interface. So, the interface is changing from version to version and often is not compatible with earlier versions. It slowly converges to the (almost) perfect shape – convenient for a wider community. If you have a particular task and not sure how to operate with artemide in this case, better write an e-mail.

The main rule (implemented in ver.2.00) each part encapsulates its theory parameters. It does not affect/change/interact with other modules, except requests for functions. A change of a parameter in a module can require a change in another module for consistency (however, I attempt to avoid such cases). Then each parameter must be changed individually in each module. However, the module aTMDe_control does it automatically. So, I suggest to use aTMDe_control to avoid possible inconsistencies.

Historical note: In versions before ver.2.00 this rule was not implemented. I tried to make connections between modules such that they automatically control consistency. However, at some moment (after inclusion of many hadrons and different types of cross-sections, and different orders) it became practically tough to keep such a system. So, I rearrange some modules (e.g. variation of c_4 scale was in TMDs, while it related to definition of TMD it-self), removed connections between modules (no link for change of NP parameters, etc.), and introduce aTMDe_control.

B. Organization of TMD factorized cross-section and its implementation in artemide

The ultimate goal of the artemide is to evaluate the observables in the TMD factorization framework, such as cross-section, asymmetries, etc. The general structure of the TMD factorized *fully differential* cross-section is

$$\frac{d\sigma}{dX} = d\sigma(q_T) = prefactor \ \times F, \tag{1.1}$$

where $prefactor$ a process-dependent and experiment-dependent prefactor, and F is the reduced structure function. Example, for the photon induced Drell-Yan process one has for $d\sigma/dq_T$

$$\begin{aligned} prefactor &= \frac{4\pi}{9sQ^2}|C_V(Q,\mu_H)|^2\mathcal{P}(\text{cuts}), \\ F &= \int \frac{bdb}{2} J_0(bq_T)\sum_f |e_f|^2 F_f(x_A,b;\mu_H,\zeta_A)F_{\bar{f}}(x_B,b;\mu_H,\zeta_B), \end{aligned}$$

where $\mathcal{P}(\text{cuts})$ is the weighting factor for fiducial cuts. For specific expression we refer in corresponding sections of this text. The structure function F is generally defined as

$$F(q_T,x_1,x_2;\mu,\zeta_1,\zeta_2) = \int \frac{bdb}{2} b^n J_n(bq_T)\sum_{ff'} z_{ff'}F_1^f(x_1,b;\mu,\zeta_1)F_2^{f'}(x_2,b;\mu,\zeta_2), \tag{1.2}$$

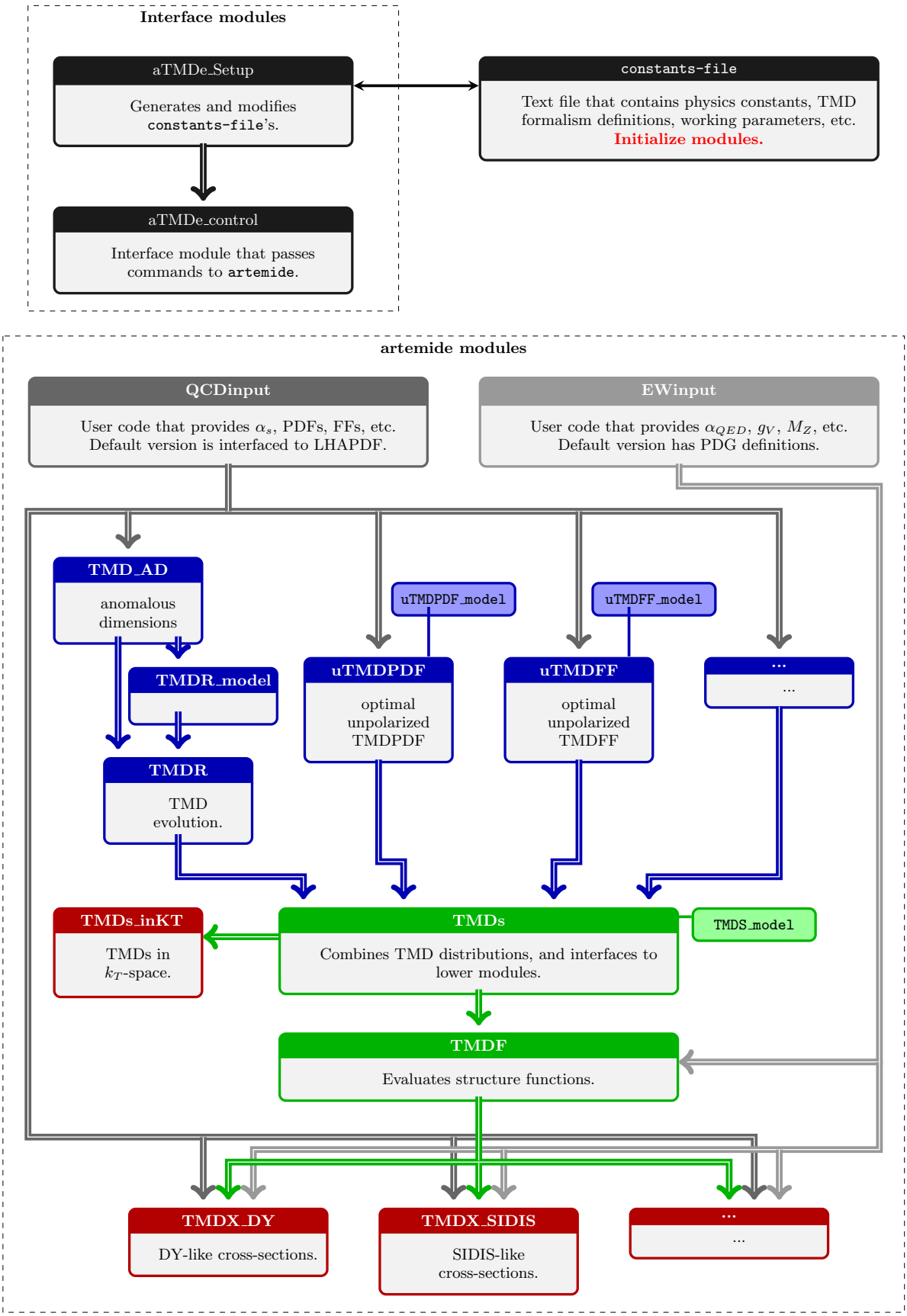


FIG. 1: Modules of `artemide`: purpose and dependencies

where $F_{1,2}$ are TMD distributions (of any origin and polarization), $z_{ff'}$ is the process dependent flavor mixing factor. The number n is also process dependent, e.g. for unpolarized observables it is $n = 0$, while for SSA's it is $n = 1$. Evaluation structure functions F is performed in the module `TMDF`. The evaluation of cross-sections is performed in the modules `TMDX`. The TMD distributions are evaluated by the module `TMDs` and related submodules.

In this way, the computation of TMD-factorized cross-section can be naturally split into ordered parts. Each part is evaluated in corresponding module of `artemide`. See example, of evaluation scheme in fig.2.

C. User defined functions and options

The `artemide` package has been created such that it allows to control **each aspect** of TMD factorization theorem. The TMD factorization has a large number of free, and "almost-free" parameters. It is a generally difficult task to provide a convenient interface for all these inputs. I do my best to make the interface convenient; however, some parts (e.g., setup of f_{NP}) could not be simpler (at least within FORTRAN). Also, take care that `artemide` is evolving, and I try to keep back compatibility, but it is not the main option.

Starting from ver.2.0, `artemide` uses the text initialization file, which contains all required information on static parameters for a given setup. Throughout the text I call this file `constants-file`.

The user has to provide (or **use the default values**) the set of parameters, that control various aspects of evaluation. It includes PDF sets, f_{NP}, perturbative scales, parameters of numerics, non-QCD inputs, etc. There are three input sources for statical parameters.

General parameters: These are working parameters of `artemide`, such as amount of output, tolerance of integration routines, number of NP parameters, type of used evolution, griding parameters, triggering of particular contributions, etc. There are many of them, and typically they are unchanged. These are set in `constants-file`. **Changes do not require recompilation.**

External physics input: It includes the definition of α_s, collinear PDFs, and other distributions. Twist-2 distributions are taken from LHAPDF [4], with routines defined in QCDinput module. For non-QCD parameters, e.g. α_{QED}, SM parameters, there is a module `EWinput`. These are set in `constants-file`. **Changes do not require recompilation.**

NP model: The NP model consists in NP profiles of TMD distributions, NP model for large-b evolution, selection of scales μ, etc. These parameter and functions enter nearly each low level module. The code for corresponding functions is provided by user, in appropriate files, which are collected in the subdirectory `src/Model`. The name of files are shown on diagram in colored blobs adjusted to the related module. **Changes require recompilation.**

Comments:

- **IMPORTANT:** Each module is initialized individually via `constants-file`. So, each module can be used independently on the full package, given proper section of `constants-file` and submodules (see diagram). However, unless you understand what is going on, it is recommended to use `aTMDe_setup` module for creation of `constant-file`, and `aTMDe_control` module for proper control, initialization and operations of sub-modules.
- `constants-file` can be saved and used in future to reproduce setup. I try to keep compatibility between these files.
- `constants-file` is created and modified within `aTMDe_setup` module. It could be also modified manually.
- NP functions are typically defined with a number of numeric parameters. The value of these parameters could be changed without restart (or recompilation) of the `artemide` by appropriate command. E.g. (`call TMDs_SetNPParameters(lambda)`) on the level of `TMDs` module. See sections of corresponding modules.
- The number of parameters in the model for each module is set in `constants-file`.

- The directory `/Model` together with `constants-file` are convenient to keep as they are. They contain full information about particular evaluation, and thus results can be always reproduced (at least within the same version of `artemide`). I provide results of our extraction as such directories.

- Before ver.2.0, the interface was different and chaotic.

D. Installation

Download and unpack `artemide`. The actual code is in the /src. Check the `makefile`. **You must provide options `FC` and `FOPT`, which are defined in the top of it.** `FC` is the FORTRAN compiler, `FOPT` is additional options for compiler (e.g. linking to LHAPDF library).

There is no actual installation procedure, there is just compilation. If model, inputs, etc, are set correctly (typical problem is linking to LHAPDF, be sure that it is installed correctly), then `make` compiles the library. The result are object files (`*.o`) (which are collected in `/obj`) and module files (`*.mod`) (which are collected in `/mod`).

The test of current compilation could be performed by `make test`. It compiles program `test.f90` from /Prog and run it (default test uses `NPDF31_nnlo_as_0118` set from LHAPDF, check that it is present in your LHAPDF installation). Program `test` runs some elementary code with minimum input. Output is shown later

Next, do your code, `include` appropriate modules of `artemide`, and compile it together with object-files (do not forget to add proper references to module files `-I/mod`). It should work! Linking could be done automatically if you call for `make program TARGET=...`, where ... is the name of the file with the code.

```
 artemide.control: initialization done.
uTMDPDF: Grid is built  (  250 x  750)  calc.time=  0.53s.
Calculating some values for cross-section one-by-one (DY around Z-boson peak, ATLAS 8TeV kinematics)
ptMin   --      ptMax           xSec
  1.0000000000000000      --    3.0000000000000000        48.499041273610260
  3.0000000000000000      --    5.0000000000000000        57.189916866533792
  5.0000000000000000      --    7.0000000000000000        49.321931214709110

Now the same by list
It must be faster since you use OPENMP
result:    48.499041273610260        57.189916866533792        49.321931214709110
-----------------------------------------------------------------------------------------------
The programm evaluation took    8.8279999999999994      sec. (CPU time)
The programm evaluation took    6.1287177319172770      sec. (wallclock time)

If you do not like so many terminal messages check the parameter outputlevel in constants file.
Not forget to cite artemide [1706.01473]
-----------------------------------------------------------------------------------------------
```

E. Python interface: `harpy`

For simplicity of data analysis the `artemide` has a python interface, called `harpy` (linking is made by f2py library). It is not possible to interfacing the `artemide` directly to python since `artemide` is made on fortran95. `Artemide` uses some features of Fortran95, such as interfaces, and indirect list declarations, which are alien to python. Also I have not found any convenient way to include several dependent Fortran modules in f2py (if you have suggestion just tell me). Therefore, I made a wrap module `harpy.f90` that call some useful functions from `artemide` with simple declarations. It contains limited set of functions useful for phenomenology, and definitely cannot replace the FORTRAN interface for deep studies.

The compilation of `harpy` is slightly more complicated.

1. In the makefile check the variable `Fpath` (in the top part of the file). Put-in the full path for fortran compiler. It is needed by f2py.

2. (optional, does not work on Mac (?)) Run `make harpy-signature` . This will create an interface file (`artemide.pyf` for all functions in the `harpy.f90`. This file is already provided in the distribution, so if you did not change `harpy.f90`, you better skip this step. For some reason, it does not work on Mac.

3. Run `make harpy`. It compiles `harpy.f90` with interface `artemide.pyf` linking to `artemide`. The result is `artemide.so`.

All files are in /harpy. Link it to python and work. There is also extra python `harpy.py` which has several most important function.

F. Constants file and version compatibility

Constants files contains the list of work-flow parameters, such as physical constants, LHA grid names, numerical setup, etc. Each constants file has version number (does not coincide with the version of artemide).

On the module initialization the constants file is read, parameter are setup. Thus, the version of constants file should be adjusted to version of artemide *if you run modules separately.*

The workflow is deferent of the initialization of `artemide` is made by the `artemide_control`. In this case, the `artemide` creates a copy of the initial constants-file (`aTMDe_temporary`), and (in the case the versions do not match) fill the absent options by default parameter. After it the lower-level modules are initialized by `aTMDe_temporary`. Naturally, in this case the versions should match (if not something is wrong with installation).

You can update the `constants-file` to an actual version by
`make update TARGET=...`
where ... is the path to `constants-file` to be updated. The updated version will have all content of the original file, + new options setup by default setting.

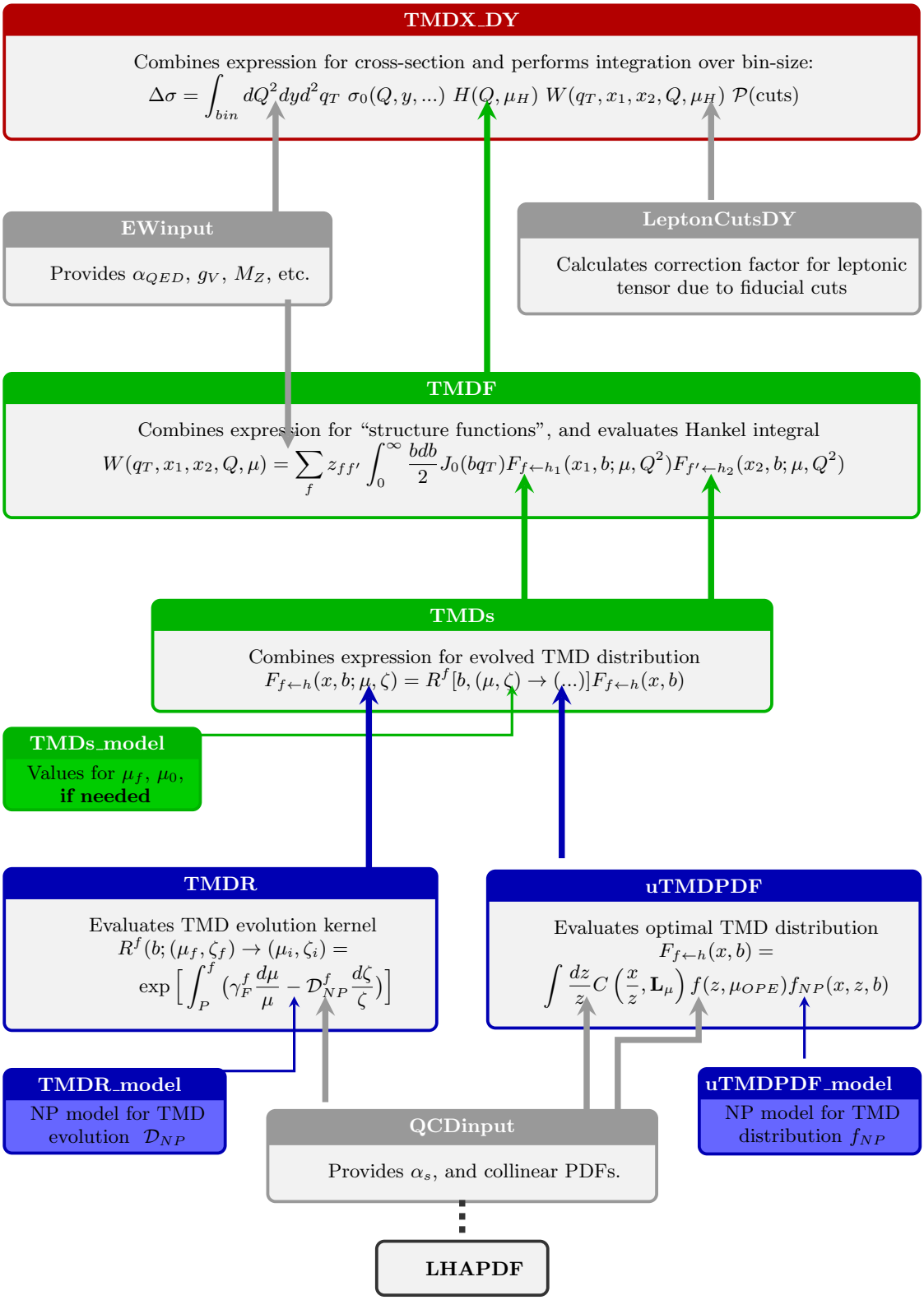


FIG. 2: Evaluation of DY cross-section by `artemide`

II. UTILITY MODULES

Utility modules are used in practically every other module of `artemide`. They include common definitions and routines that are not related to physics, but to programming. This set includes the following modules

- **aTMDe_numerics** which defines the numeric precisions (default is `dp=selected_real_kind(15, 307)`, and standard mathematical constants.
- **aTMDe_interfaces** which defines standard procedure interfaces
- **aTMDe_IO** which defines functions for text coloring, and printing in the terminal, as well as, count system for warning messages
- **aTMDe_integration** which defines elementary integration routines. Used for general purposes.
- **aTMDe_Ogata** which defines Hankel transform of 1D function. The computation is by Ogata quadratures [5].
- **aTMDe_invMatrix** which defines the operation of matrix inverse. The main part of this code has been taken from https://github.com/Beliavsky/Matrix_Inversion/blob/main/linear_solve.f90.

III. ATMDE_SETUP MODULE

The module aTMDe_setup creates and modifies the constants-file. **It is a stand-alone module, which does not require the rest modules** (however, it is called by aTMDe_control).

List of commands optional parameters are shown in blue.

Command	Sec.	Short description
artemide_Setup_Default(order)	III A	The main command which initializes variables by default values corresponded to a particular order.
artemide_Setup_fromFile(file,prefix,order)	III A	The main command which initialize variables by pre-saved values in file. prefix is path to the file. order is the order of default version of parameters (used if versions of files are incompatible).
CreateConstantsFile(file,prefix)	-	Write a new constants-file according to current modification.
CheckConstantsFile(file,prefix)	-	Function (logical). Compare the version of constants-file. Returns .true. if version of file ¿= version of artemide, .false. otherwise.
artemide_include(arg1,arg2,...,arg10)	-	Include the modules into the initialization procedure. arg is (string) with the module name.
Set_outputLevel(level,numMessages)	-	Set the level of artemide-messages to (int)level. 0= only critical, 1=+warnings, 2=+module evaluation information, 3=+ details. Default=2. (integer)numMessages set number of non-critical Warnings of the same type to show (prevent spaming by same messages).
Set_uPDF(hadron,setName,replica)	V A	Assign the hadron for uPDF with number hadron(int) a PDF set setName(string) in LHAPDF library. It will be initialized in with replica replica(int)(default =0).
Set_uFF(hadron,setName,replica)	V A	Assign the hadron for uFF with number hadron(int) a FF set setName(string) in LHAPDF library. It will be initialized in with replica replica(int)(default =0).
Set_lpPDF(hadron,setName,replica)	V A	Assign the hadron for (unpolarized)PDF that is used by lpTMDPDF with number hadron(int) a PDF set setName(string) in LHAPDF library. It will be initialized in with replica replica(int)(default =0).
Set_quarkMasses(mC,mB,mT)	-	Set values for pole quark masses, charm, bottom and top (real), which determine N_f-thresholds. Default mC=1.4, mB=4.75, mT=173.
Set_EWparameters(alphaInv,massZ,massW,widthZ,widthW, massH,widthH,vevHIGGS, sin2ThetaW,UD,US,UB,CD,CS,CB)	-	Set parameters of electro-weak theory. alphaInv=inverse $\alpha_{QED}(M_Z)$. UD,US,UB,CD,CS,CB are elements of CKM matrix. Masses and widths are in GeV.
Set_TMDR_order(order)	??	Set the perturbative order of anomalous dimensions used for evolution. order=string(8)
Set_TMDR_evolutionType(num)	??	Set the type of evolution solution used to (int)num.
Set_TMDR_lengthNParray(num)	??	Set the length of λ_{NP} for $\mathcal{D}_{NP}$ to num(int).
Set_uTMDPDF(hadron,setName,replica)	V A ??	Assign the hadron for uTMDPDF with number hadron(int) a PDF set setName(string) in LHAPDF library. It will be initialized in with replica replica(int)(default =0). Automatically calls for Set_uPDF.
Set_uTMDPDF_order(order)	??	Set the perturbative order of convolution. order=string(8)
Set_uTMDPDF_gridEvaluation (prepareGrid,includeGluon)	??	Set the trigger to prepare the grid, and to include the gluons in the grid (default=.false.). Both logical.
Set_uTMDPDF_lengthNParray(num)	??	Set the length of λ_{NP} for uTMDPDF NP model to num(int).
Set_uTMDFF(hadron,setName,replica)	V A ??	Assign the hadron for uTMDFF with number hadron(int) a FF set setName(string) in LHAPDF library. It will be initialized in with replica replica(int)(default =0). Automatically calls for Set_uFF.

Set_uTMDFF_order(order)	??	Set the perturbative order of convolution. order=string(8)
Set_uTMDFF_gridEvaluation (prepareGrid,includeGluon)	??	Set the trigger to prepare the grid, and to include the gluons in the grid (default=.false.). Both logical.
Set_uTMDFF_lengthNParray(num)	??	Set the length of λ_{NP} for uTMDFF NP model to num(int).
Set_lpTMDPDF(hadron,setName,replica)	V A ??	Assign the hadron for lpTMDPDF with number hadron(int) a PDF set setName(string) in LHAPDF library. It will be initialized in with replica replica(int)(default =0). Automatically calls for Set_lpPDF.
Set_lpTMDPDF_order(order)	??	Set the perturbative order of convolution. order=string(8)
Set_lpTMDPDF_gridEvaluation (prepareGrid,includeGluon)	??	Set the trigger to prepare the grid, and to include the gluons in the grid (default=.true.). Both logical.
Set_lpTMDPDF_lengthNParray(num)	??	Set the length of λ_{NP} for lpTMDPDF NP model to num(int).

A. aTMDe_Setup

TO BE WRITTEN

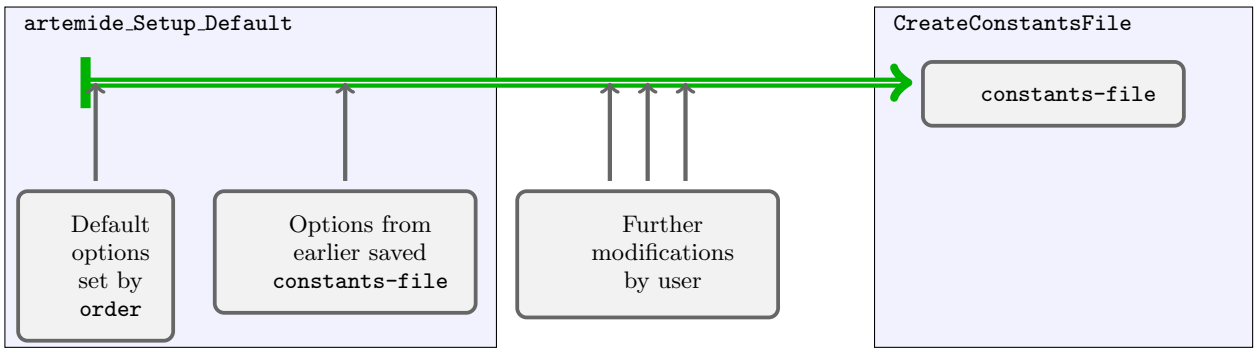


FIG. 3: Scheme of creation and modification of constants-file by aTMDe_setup.

IV. ATMDE_CONTROL MODULE

The module aTMDe_setup is used to coordinate the operation of other modules. It does not bring any new features, just operates other modules in proper order. It is only for convenience.

List of commands optional parameters are shown in blue.

Command	Sec.	Short description
artemide_Initialize(file,prefix,order)	III A	The command which initialize modules according to constants-file created by artemide_Setup_fromFile(file,prefix,order) .
artemide_ShowStatistics()		Shows some information.
artemide_SetNPparameters(lambdaNP)	IV A	Receive a (real*8)list of NP parameters, split it according to current setup and passes NP parameters to appropriate modules. Reset modules counters.
artemide_SetNPparameters_TMDR(lambdaNP)	IV A	Reset NP parameters of TMDR-module by (real*8) list lambdaNP. Reset modules counters.
artemide_SetNPparameters_uTMDPDF(lambdaNP)	IV A	Reset NP parameters of uTMDPDF-module by (real*8) list lambdaNP. Reset modules counters.
artemide_SetNPparameters_uTMDFF(lambdaNP)	IV A	Reset NP parameters of uTMDFF-module by (real*8) list lambdaNP. Reset modules counters.
artemide_SetNPparameters_lpTMDFF(lambdaNP)	IV A	Reset NP parameters of lpTMDFF-module by (real*8) list lambdaNP. Reset modules counters.
artemide_SetReplica_TMDR(num)	IV A	Reset NP parameters of TMDR-module by values corresponding to replica (int)num. Reset modules counters.
artemide_SetReplica_uTMDPDF(num)	IV A	Reset NP parameters of uTMDPDF-module by values corresponding to replica (int)num. Reset modules counters.
artemide_SetReplica_uTMDFF(num)	IV A	Reset NP parameters of uTMDFF-module by values corresponding to replica (int)num. Reset modules counters.
artemide_SetReplica_lpTMDPDF(num)	IV A	Reset NP parameters of lpTMDPDF-module by values corresponding to replica (int)num. Reset modules counters.
artemide_SetScaleVariations(c1,c2,c3,c4)	-	Change the value of scale-variation constants $c_1 - c_4$.
artemide_GetReplicaFromFile(file,rep,array)	-	Read the .rep file (path is file), and search for the replica (int)rep. Return the (real*8,allocatable)array of NP parameters. **This command can work without initialization of artemide-control**, in this case, no check of consistency is performed and warning raised.
artemide_NumOfReplicasInFile(file)	-	Read the .rep file (path is file), and return the number of replicas saved in it.

A. Passing non-perturbative parameters

The important part of the initialization is the number of NP parameters for each TMD distributions under consideration. Each TMD-evaluating module (say, uTMDPDF, uTMDFF, etc.) requires n_i number of parameters. The numbers are specified in constants-file. The number must be greater then zero $n_i > 0$ for any used module, i.e. f_{NP} is at least 1-parametric (if it is not so, just do not use the parameter in the definition of f_{NP}, but keep $n_i > 0$). These numbers are read during the initialization procedure, and allocate the memory.

The set of particular values of these parameters can be done by several ways.

Option I: Set all values in a single call
`call artemide_SetNPParameters(lambda)`
where

$\{\lambda_i\}$ real*8(1:$\sum_i n_i$) The set of parameters which define the non-perturbative functions f_{NP} within modules. It is split into parts and send to corresponding modules. I.e. lambda(1:n_0) →`uTMDR`, lambda(n_0 + 1:n_0 + n_1) →`uTMDPDF`, lambda($n_0 + n_1 + 1$:$n_0 + n_1 + n_2$) →`uTMDFF` (fixed order).

Option II: Set value for particular function. For it call `artemide_SetNPparameters_???(lambdaNP)`, where `???` is module name, e.g.

`artemide_SetNPparameters_uTMDPDF(lambdaNP)`

will set parameters for unpolarized TMDPDF.

Option III: You can use presaved values of λ_{NP} for a given function. They are provided by model file (if provided). To set NP-input given by integer `num`, call `artemide_SetReplica_???(num)`, where `???` is module name, e.g.

`artemide_SetReplica_uTMDPDF(num)`

will set parameters for unpolarized TMDPDF.

V. QCDINPUT MODULE

The module QCDinput gives an interface to external function provided by the user, such as PDF, FF, values of alpha-strong. By default the input for these functions is done via the LHAPDF tables. Originally, artemide used the LHAPDF library [4], which however caused a number of inconveniences, such as difficulty in the installation, problems with simultaneous usage of many tables, and parallel access to the tables. Starting from the version 3.01 artemide is detached from the LHAPDF library, but continue to relay on LHAPDF-tables as input for the QCD parameters. The tables are read and interpolated by the original artemide sub-modules (LHA_alpha and LHA_PDF). The details on these modules are presented below.

For the technical reasons there is a **hard-coded limit for the maximum number of hadrons**. This limit can be changed but it requires modification of the code. The default limitation is the following:

- uPDF 3 hadrons,
- uFF 6 hadrons,
- lpPDF 1 hadron,
- gPDF 2 hadrons (helicity PDFs),
- hPDF 2 hadrons (transversity PDFs).

List of available commands optional parameters are shown in blue.

Command	Description
QCDinput_Initialize(file,prefix)	Subroutine to initialize anything what is needed. (string) file is the name of constants-file, which contains initialization information. (string)prefix is appended to file if provided.
As(Q)	Returns the (real*8) value of $\alpha_s(Q)/(4\pi)$. Q is (real*8).
xPDF(x,Q,hadron)	Returns the (real*8(-5:5)) value of $xf(x,Q)$ for given hadron. x, Q are (real*8), hadron is (integer).
xFF(x,Q,hadron)	Returns the (real*8(-5:5)) value of $xd(x,Q)$ for given hadron. x, Q are (real*8), hadron is (integer).
x_hPDF(x,Q,hadron)	Returns the (real*8(-5:5)) value of $xg_1(x,Q)$ for given hadron (g_1=helicity distribution). x, Q are (real*8), hadron is (integer).
mCHARM	(real*8)Constant for mass of charm quark.
mBOTTOM	(real*8)Constant for mass of bottom quark.
QCDinput_SetPDFreplica(num,h)	Changes the unpolarized PDF replica number for hadron h to rep.
QCDinput_SetFFreplica(num,h)	Changes the unpolarized FF replica number for hadron h to rep.
QCDinput_SetlpPDFreplica(num,h)	Changes the replica of PDF associated with linearly polarized gluon PDF number for hadron h to rep.
QCDinput_SethPDFreplica(num,h)	Changes the replica of helicity PDF number for hadron h to rep.

A. Initialization

TOBE WRITTEN

B. `LHA_alpha` module

The module reads the info-file of the LHAPDF set and extracts the information about α_s. For the moment the only possible input is with "`AlphaS_Type=ipol`". The interpolation is done by cubic-interpolation in logarithm scale, for each mass-threshold subgrid (in accordance of the description of LHAPDF library). Compared that the deviation from the original LHA library and `artemide` is better that 10^{-6} (i.e. at the level of float-definition fluctuations).

Below the $Q_{\min}$ I perform the extrapolation by the following formula

$$\alpha_s(Q) = \frac{\alpha_0\alpha_1 \ln(Q_1/Q_0)}{(\alpha_0-\alpha_1)\ln(Q/Q_0)+\alpha_1\ln(Q_1/Q_0)}, \tag{5.1}$$

where $Q_0 = Q_{\min}$ and Q_1 is the next-to-$Q_{\min}$ node, and $\alpha_{0,1} = \alpha_s(Q_{0,1})$. This formula extrapolate $1/\alpha$ logarithmically in the logarithm scale. It gives the values of Λ_{eff}

$$\Lambda_{\text{eff}} = Q_0 \exp\left(-\frac{\alpha_1\ln(Q_1/Q_0)}{\alpha_0-\alpha_1}\right), \tag{5.2}$$

where the coupling constant diverges. The minimal possible Q is truncated to $\min(0.4\text{GeV}, \Lambda_{\text{eff}} + 0.1\text{GeV})$ to avoid problems with low energy.

C. `LHA_PDF` module

The module reads info-file and the PDF-table and interpolates using the log-qubic interpolation (instead of cubic-Hermite interpolator from LHAPDF-library). There is a small-difference between interpolating results but they are smaller that 10^{-4} (relative). I.e. far inside of the pdf uncertainties.

Below the $Q_{\min} = Q_0$ the extrapolation is done by the formula

$$F(x,Q) = F(x,Q_0)(Q/Q_0)^{\Gamma(Q)}, \qquad \Gamma(Q) = 2+\sqrt{\frac{Q}{Q_0}}(\gamma(x)-2), \tag{5.3}$$

where $F(x,Q) = xf(x,Q)$, and $\gamma(x)$ is

$$\gamma(x) = \frac{\ln F(Q_1) - \ln F(Q_0)}{\ln(Q_1/Q_0)} \approx \frac{d\ln F(Q)}{d\ln Q}. \tag{5.4}$$

This is in the agreement of formulas (4,5) of ref.[4]. Below 0.4GeV the extrapolation terminated and error is called.

VI. EWINPUT MODULE

`EWinput` module contains definitions of various physical parameters of elector-weak interactions, such as masses of Z and W bosons, CKM matrix, code for evolution of α_{QED} etc.
TOBE WRITTEN

A. $\alpha_{\rm QED}$

This implementation has been added in ver.2.03. In earlier versions the code for $\alpha_{\rm QED}$ had a bug that generated slightly lower values (independently on the values of μ).

The QED coupling constant is set by the one-loop QED evolution with LO matching at thresholds. In order to tune it finer, the value of β_0 is modified by $\beta_0 \to \beta_0 + \delta\beta$, where $\delta\beta$ is found by exact implementation of two boundary values, $\alpha_{\rm QED}(m_\tau)$ and $\alpha_{\rm QED}(m_Z)$.

The LO evolution is

$$\alpha(\mu) = \frac{1}{\alpha_0^{-1} + 2\beta_0 \ln(\mu/\mu_0)}, \tag{6.1}$$

where $\beta_0 = -(3\pi)^{-1} N_{eff}$ with

$$N_{eff} = \sum_{\substack{\text{active}\\ \text{leptons}}} 1 + N_c \sum_{\substack{\text{active}\\ \text{quarks}}} e_q^2. \tag{6.2}$$

In artemide I account the following threasholds

μ	$\mu < m_e(\sim m_u, m_d)$	$\mu < m_\mu(\sim m_s)$	$\mu < m_c$	$\mu < m_\tau$	$\mu < m_b$	$\mu < m_t$	$\mu > m_t$
N_{eff}	0	$\frac{8}{3}$	4	$\frac{16}{3}$	$\frac{19}{3}$	$\frac{20}{3}$	8

So the actual formula for α_{QED} is

$$\alpha_{\rm QED}(\mu) = \frac{1}{\alpha_Z^{-1} + 2\bar\beta \left[N_{eff}(\mu) \ln\left(\frac{\mu}{m_{th}}\right) + \sum_{m_i \in m_{th}} N_{eff} \ln\left(\frac{m_i}{m_{i+1}}\right)\right]}, \tag{6.3}$$

where m_{th} are all threasholds from μ till M_Z. The constant $\bar\beta$ is defined by exact matching at M_Z and m_τ. For default values $\bar\beta/\beta_0 \sim 1.002\%$, so it accounts for really tiny effects.

VII. TMD EVOLUTION MODULES

The TMD evolution has several implementation. The artemide v3 (and higher) is hard-coded to use the fixed-μ evolution (see ref.[6]). This choice is made after many test performed in earlier versions or artemide (where the type of evolution was selected by user), and the demonstration that they are equivalent within perturbative uncertainty.

The fixed-μ scale corresponds to evolution along ζ is with fixed-μ solution to the special equi-potential line, and evolution along μ along equi-potential line (this part is unity by definition). The evolution factor reads

$$R[b,(\mu_1,\zeta_1)\to(\mu_2,\zeta_2)] \;=\; \frac{R[b,(\mu_1,\zeta_1)\to(\mu_1,\zeta_{\mu_1})]}{R[b,(\mu_2,\zeta_2)\to(\mu_2,\zeta_{\mu_2})]}, \tag{7.1}$$

$$R[b,(\mu,\zeta)\to(\mu,\zeta_\mu)] \;=\; \exp\left(-\mathcal{D}(\mu,b)\ln\left(\frac{\zeta}{\zeta_\mu(b)}\right)\right), \tag{7.2}$$

where $\mathcal{D}$ is the Collins-Soper kernel, and $\zeta_\mu(b)$ is the special ζ-line.

The Collins-Soper kernel is defined as

$$\mathcal{D}^f(\mu,b)=\mathcal{D}^f_{\rm pert}(\mu_{\rm OPE},b)+\int_{\mu_{\rm OPE}}^{\mu}\frac{d\mu'}{\mu'}\Gamma^f(\mu')+\mathcal{D}^f_{\rm NP}(b), \tag{7.3}$$

where $\mathcal{D}^f_{\rm pert}$ is the high-scale perturbative part, and $\mathcal{D}_{\rm NP}$ is the non-perturbative model (e.g. $g_K b^2$)

The module for evolution includes three modules

- `TMD_AD` this module contains collection of all anomalous dimensions and related quantities. Logically it is equivalent to `_OPE` modules.
- `TMDR_model` this module contains the user-defined non-perturbative input.
- `TMDR` this module interface `TMD_AD` and `TMDR_model` into the evolution factor, and perform the general control.

The initialization is happened thorough the `TMDR`-module.

At small values of parameter $Q=Q_0$ the point (Q,Q^2) crosses the ζ-lines. The value of Q_0 depends on b. The dangerous situation is then hard scale of the process Q is smaller then Q_0 at large $b=b_\infty$. In this case the evolution kernel $R[b_\infty,(Q,Q^2)\to\zeta_\mu]>1$, which is generally implies that it grows to infinity. However, it happens only at small values of Q. E.g. at NNLO the typical value of Q_0 is ~ 1.5GeV. That should be taken into account during consideration of low-energy experiment and especially their error-band, since the point (c_2Q,Q^2) could cross the point at larger values of Q.

The function `LowestQ()` returns the values (real*8(1:3)) $\{Q_{-1},Q_0,Q_{+1}\}$, which are solution of equation $Q^2=\zeta_{cQ}(b)$, for (fixed bu) large values of b. Q_{-1} corresponds to $c=0.5$, Q_0 corresponds to $c=1$ and Q_{+1} corresponds to $c=2$.

1. INI-parameters

Parameters for TMD-evolution modules (section `*3`).

Entry	Type	Description
A.p1	str	Order of the evolution *standardized*. Could be LO, NLO,.. (see table below for details)
A.p2	T/F	The flag to override the *standard* evolution definition. If T, than instead of A.p1 the orders defined in A.p3 – A.p6 are used.
A.p3	str	Order of $\Gamma_{\rm cusp}$. LO=$a_s\Gamma_0$, NLO=$a_s\Gamma_0 + a_s^2\Gamma_1$, etc.
A.p4	str	Order of γ_V. LO=0, NLO=$a_s\gamma_1$, etc.
A.p5	str	Order of $\mathcal{D}_{\rm pert}$. LO=0, NLO=$a_s\mathbf{L} + d^{(1,0)}$, etc.
A.p6	str	Order of ζ_μ (exact solution). LO=a_s^0*(exact solution), etc.
B.p1	int	Number of NP parameters.
C.p1	real	Tolerance general (used for check of variable modifications, etc. Does not affect integration precision). (default =10^{-6})
C.p2	real	Small-b value at which the evolution is frozen (default =10^{-6})
C.p3	real	Parameter for smoothing the small-b value of exact ζ-line (see below) (default =0.01).

The table of correspondence between `standard` definition and particular definitions

`order`	$\Gamma_{\rm cusp}$	γ_V	$\mathcal{D}$ *	$\mathcal{D}_{\rm resum}$	ζ_μ **
`LO`	a_s^1	a_s^0	a_s^0	a_s^0	a_s^0
`NLO`	a_s^2	a_s^1	a_s^1	a_s^1	a_s^2
`NNLO=N2LO`	a_s^3	a_s^2	a_s^2	a_s^2	a_s^3
`NNNLO=N3LO`	a_s^4	a_s^3	a_s^3	a_s^2	a_s^4
`N4LO`	a_s^5	a_s^4	a_s^4	a_s^3	a_s^{5***}

* $\mathcal{D}_{\rm resum}$ starts from a_s^0, it already contains Γ_0.

** Definition of ζ_μ is correct only in the natural ordering, i.e. `LO`,`NLO`,`NNLO`. Proper definition in `+` orders would make the function too heavy. The resumed version has the same counting.

*** The finite part =0, since $d^{(5,0)}$ is unknown.

Notes:

- For very small-values of b the value of b is fixed.
- For very large positive values of $\mathcal{D}$ (i.e. for $b > 25\text{GeV}^{-1}$ or so), the exact solution for ζ-line diverges exponentially at 3- and 4- loops. It is because the coefficient infront of rising terms is negative. To prevent the divergence, the corresponding terms are expanded in a_s up to needed power (α_s^6).

2. Functions

`XX`=name of module (e.g. `uTMDPDF`, `uTMDFF`, etc.)

Entry	Type	Description
`TMDR_Initialize(file,prefix)`	subroutine	Initialize the module from INI-file.
`TMDR_SetScaleVariation(c1)`	subroutine	Sets globally a new value for constant c_1.
`TMDR_SetNPparameters(lambda)`	subroutine	Set a new vector of NP parameters $\lambda_{\rm NP}$. The new vector must be same size as declared at the setup.
`TMDR_CurrentNPparameters()`	real(:)	Returns the current vector for $\lambda_{\rm NP}$.
`CS_kernel(mu,b,f)`	real	Returns the value Collins-Soper kernel at scale μ, point b for flavor f.
`zetaNP(mu,b,f)`	real	Returns the value ζ_μ for give CS kernel at scale μ, point b for flavor f.
`TMDR_Rzeta(b,muf,zetaf,f)`	real	Return the factor R for evolution from the point (μ,ζ) down to the special evolution line.
`TMDR_R(b,muf,zetaf,mui,zetai,f)`	real	Return the factor R for evolution from the point (μ_f,ζ_f) to (μ_i,ζ_i).
`LowestQ()`	real	Attempts to compute the lowest available scale Q, for which the evolution converges.

A. `TMD_AD` module

The module `TMD_AD` give access to various perturbative anomalous dimensions needed for TMD evolution. This module is essential part of `TMDR` module, and thus initialized by it.

List of available commands

Command	Sec.	Short description
`TMD_AD_Initialize(oC,oV,oD,oDr,oZ)`	??	Initialization of module. The (integer) parameters (`oC,oV,oD,oDr,oZ`) are orders of definition for anomalous dimensions ($\Gamma_{\rm cusp}$, γ_V, $\mathcal{D}_{\rm pert}$, $\mathcal{D}_{\rm resum}$, $\zeta_{\rm pert}$).
`GammaCusp(mu,f)`		$=\Gamma^f_{\rm cusp}(\mu)$
`gammaV(mu,f)`		$=\gamma^f_V(\mu)$
`Dpert(mu,bT,f)`		$=\mathcal{D}^f_{\rm pert}(b,\mu)$
`Dresum(mu,bT,f)`		$=\mathcal{D}^f_{\rm resum}(b,\mu)$
`zetaMUpert(mu,bT,f)`		$=\zeta^f_{\rm pert}(b,\mu)$
`zetaSL(mu,rad,f)`		$=\zeta^f_{\rm exact}(\mathcal{D},\mu)$ (Note, the argument.)
`RADEvolution(mu0,mu1,f)`		$=\int_{\mu_0}^{\mu_1}\frac{d\mu^2}{\mu^2}\frac{\Gamma(\mu)}{2}$ The evolution constant for $\mathcal{D}$

Notes:

- The integral over μ is done using exact analytical formula by decomposition of β-function over roots.
- The exact expression for ζ-line is numerically unstable at very small-b. It is due to its double-exponential nature which requires exponentiation and cancellation of large-numbers. However, it could be computed analytically in this regime as expansion. Thus, to stabilize the computation of expression the following formula is used

$$\zeta_\mu = e^{-b^2/S}\zeta_{\rm pert} + (1-e^{-b^2/S})\zeta_{\rm exact}. \tag{7.4}$$

  Here, S is the smoothing parameter. It should be small enough to guaranty non-impact of approximation (approx. $S<0.05$).

B. TMDR_model

This module gives the expression for $\mathcal{D}$. It could be modified by user, and one of modules that form a TMD model (together with models for TMD-distributions).
Making this module follow the interface

```
public::  ModelInitialization(NPlength)
public::  ModelUpdate(NParray)
real(dp),public::  DNP(b,f)
real(dp),public::  bSTAR(b)
real(dp),public::  muOPE(b,c1)
```

`ModelInitialization(NPlength)`: this subroutine is called during the initialization of `TMDR`. The `integer,intent(in)::NParray(:)` is the length of array of NP-parameters. Use this subroutine to make preparation of the model.

`ModelUpdate(NParray)`: this subroutine is called during the `TMDR_setNPparameters(NParray)`. The `real(dp),intent(in)::NParray(:)` is the array of NP-parameters passed to `TMDR_setNPparameters`. Use this subroutine to recompute model parameters (if needed), save NP-parameters (if needed), etc.

`DNP(b,f)`: the function returns the value, $\mathcal{D}^f_{\rm NP}(b)$. It contains only NP part.

`bSTAR(b)`: the function returns the value, $b^*(b)$ used instead of b within $\mathcal{D}_{\rm pert}$

`muOPE(b,c1)`: the function returns the value, $\mu_{\rm OPE}$ used as reference scale of $\mathcal{D}_{\rm pert}$. The parameter c_1 is the prescription used during the scale variation.

Important: the module `TMDR_model` is compiled before `TMDR`, and thus can use only lower-level modules: `TMD_AD`, `QCDinput`, etc.

Important: There is an option to switch to the CSS-like computation (formula (2.14) in 1803.11089).

VIII. MODULES COMPUTING TMD DISTRIBUTIONS

There are many TMD distributions each computed in a separate module. All TMD distributions are split in the following classes:

- **Pure Twist-2 OPE** These are `uTMDPDF`, `uTMDFF`, `lpTMDPDF`. These TMD distributions are defined by twist-2 collinear distribution at small-b
- **Pure Twist-3 OPE** These are `SiversTMDPDF`, `BoerMuldersTMDPDF`. These TMD distributions are defined by twist-3 collinear distribution at small-b
- **Twist-2 & Twist-3 OPE** These are `wgtTMDPDF`, `wglTMDPDF`. These TMD distributions are defined by twist-2 (WW-term) and twist-3 collinear distribution at small-b
- **Pure-parton TMDs** These are `eeTMDFF`. These TMD distributions are defined by a pure parton (i.e. jet), and thus their small-b expansion is just a number (i.e. does not contain any hadron matrix elements).

These lists are to be extended in the future.

Each module consists of several sub-modules, which have names as `TMDname_X`. The `X` corresponds to

- `OPE` this module computes the convolution of the twist-2 (or twist-3) coefficient function with corresponding PDF.
- `model` this module contains all information about NP-part and other free-parameters (such as $\mu_{\rm OPE}$ of the given TMD distribution. **These modules could be edited by user, to implement/update/fit TMDs.**

The most part of modules are identical. There are a few differences in the implementation. So, first the general structure is presented, and then particularities.

Different OPE's are known up to different orders. Consequently, the modules implement them at different orders. The orders are denoted as

$$\begin{aligned}
\texttt{NA} &:\to \text{no OPE}\\
\texttt{LO} &:\to C=\mathcal{O}(a_s^0)\\
\texttt{NLO} &:\to C=\mathcal{O}(a_s^1)\\
\texttt{NNLO}=\texttt{N2LO} &:\to C=\mathcal{O}(a_s^2)\\
\texttt{NNNLO}=\texttt{N3LO} &:\to C=\mathcal{O}(a_s^3).
\end{aligned}$$

- The `NA` option is used to remove any relation to collinear distributions. In this case $TMD(x,b)=f_{\rm NP}(x,b)$
- If the coefficient function at LO/NLO/.. is zero, the convolution is also zero. E.g. linearly polarized gluons start at $\sim a_s$, the convolution is zero at `LO`.

List of available orders for different distributions

```
                uTMDPDF : NA, LO, NLO, N2LO, N3LO
                 uTMDFF : NA, LO, NLO, N2LO, N3LO
              lpTMDPDF : NA, LO, NLO, N2LO
        wgtTMDPDF[tw2] : NA, LO, NLO
        wgtTMDPDF[tw3] : NA
        wglTMDPDF[tw2] : NA, LO, NLO
        wglTMDPDF[tw3] : NA
      SiversTMDPDF[tw3] : NA
BoerMuldersTMDPDF[tw3] : NA
               eeTMDFF : NA, LO, NLO, N2LO, N3LO
```

The functions are computed by the following rule

$$\begin{aligned}
\texttt{uTMDPDF} &= [C \times f]_{\rm tw2}(x,b) F_{\rm NP}(x,b), && (8.1)\\
\texttt{uTMDFF} &= [C \times d]_{\rm tw2}(x,b) F_{\rm NP}(x,b), && (8.2)\\
\texttt{lpTMDPDF} &= [C \times f_{\rm lp}]_{\rm tw2}(x,b) F_{\rm NP}(x,b), && (8.3)\\
\texttt{wgtTMDPDF} &= [C \times g]_{\rm tw2}(x,b) F_{\rm NP}(x,b) + [C \times T]_{\rm tw3}(x,b) F^{\rm tw3}_{\rm NP}(x,b), && (8.4)\\
\texttt{wglTMDPDF} &= [C \times h]_{\rm tw2}(x,b) F_{\rm NP}(x,b) + [C \times T]_{\rm tw3}(x,b) F^{\rm tw3}_{\rm NP}(x,b), && (8.5)\\
\texttt{SiversTMDPDF} &= [C \times T]_{\rm tw3}(x,b) F^{\rm tw3}_{\rm NP}(x,b), && (8.6)\\
\texttt{BoerMuldersTMDPDF} &= [C \times E]_{\rm tw3}(x,b) F^{\rm tw3}_{\rm NP}(x,b), && (8.7)\\
\texttt{eeTMDFF} &= C(b) F_{\rm NP}(b), && (8.8)
\end{aligned}$$

where f (`uPDF`), d (`uFF`), $f_{\rm lp}$ (`lpPDF`) g (`gPDF`), h (`hPDF`) are collinear distributions of twist-2 (taken from corresponding module), and $[C \times h]_{\rm tw2}$ is corresponding convolution; T and E are collinear distributions of twist-3, and $[C \times T]_{\rm tw3}$, $[C \times E]_{\rm tw3}$ are corresponding convolution; $F_{\rm NP}(x,b)$ and $F^{\rm tw3}_{\rm NP}(x,b)$ are non-perturbative functions to be fit. NP functions are defined in the `XX_model` modules, and share the same $\lambda_{\rm NP}$ (array of parameters). This structure correctly reproduces the small-b properties of TMD-distributions. The convolutions are computed in separate modules see sec.VIII B, VIII C.

The modules computes the optimal TMDs, and also evolved TMDs. Generally a TMD distribution is given by the expression

$$F_f(x,b;\mu,\zeta) = R_f[b,(\mu,\zeta) \to (\mu,\zeta_\mu)] \tilde{F}_f(x,b), \tag{8.9}$$

where R is the TMD evolution kernel, $\tilde{F}$ is an optimal TMD distribution.

1. INI-parameters

Parameters universal for all such modules which include only tw2 or tw3 OPE (mixed case see below).

Entry	Type	Description
A.p1	T/F	Include gluon into the computation. Usually gluon is computed by a separate rule, and switching off this option leads to ~ 2 increase in the computation velocity.
A.p2	in	Number of hadrons. Specify maximum required number of hadrons.
B.p1	str	Order of the coefficient function. Could be N/A, LO, NLO,.. (see table below for details)
B.p2	T/F	Trigger to prepare the grid (T=prepare). Preparation is done on the initialization stage.
B.p3	T/F	Trigger the test of the grid (T=run the test). Take the same time as computation of the grid (by a single processor). Requires A.p2=T. Example Result is shown in sec.VIII B 4.
C.p1	int	Number of NP parameters.
C.p2	real	$B_{\text{MAX-ABS}}$, a large number such that $b > B_{\text{MAX-ABS}}$ the TMD is returned 0. (default 100).
D.p1	real	Tolerance for integration (default $=10^{-6}$)
D.p2	real	Tolerance general (used for check of variable modifications, etc. Does not affect integration precision). (default $=10^{-6}$)
D.p3	int	Maximum number of iteration for adaptive integration. (default $=10^4$)
E.p1	real	$x_{\min}$ for the grid (see (**??**))
E.p2	real	p_x for the grid (see (**??**))
E.p3	real	$b_{\min}$ for the grid (see (**??**)). [It is not related to parameter $b_{\min}$ of b^*-prescription!]
E.p4	real	$b_{\max}$ for the grid (see (**??**)) [It is not related to parameter $b_{\max}$ of b^*-prescription!]
E.p5	integer	Number of nodes in the x-grid
E.p6	integer	Number of nodes in the b-grid
F.p1	real	Tolerance use in the Fourier integral to the k_T-space (default $=10^4$)
F.p2	real	Step for the Ogata quadrature (default $=10^{-3}$)
F.p3	real	Minimal k_T to compute. Below this number distribution is frozen. (default $=10^{-4}$)
G.p1	real	Tolerance use in the Fourier integral for TMM (default $=10^4$)
G.p2	real	Step for the Ogata quadrature for TMM computation (default $=10^{-3}$)
G.p3	real	Minimal μ to compute. Below this number the program terminates. (default =0.8GeV)

- `wgtTMDPDF` has extra section F and G, which are analogous to sections B and C, but responsible for twist-3 convolution. The section F,G,I is correspondingly sections H,I.
- `eeTMDFF` does not have most part of entries, because it is $x-$independent, and thus expressions are very simple and there is no necesity to build grids.

2. Functions

`XX`=name of module (e.g. `uTMDPDF`, `uTMDFF`, etc.)

Entry	Type	Description
`XX_Initialize(file,prefix)`	subroutine	Initialize the module from INI-file.
`XX_SetPDFreplica(rep,h)`	subroutine	Passes to `XX_OPE_SetPDFreplica`
`XX_SetScaleVariation(c4)`	subroutine	Passes to `XX_OPE_SetScaleVariation`
`XX_SetLambdaNP(lambda)`	subroutine	Set a new vector of NP parameters $\lambda_{\rm NP}$. The new vector must be same size as declared at the setup.
`XX_CurrentLambdaNP()`	real(:)	Returns the current vector for $\lambda_{\rm NP}$.
`XX_lowScale5(x,bT,h)`	real(-5:5)	TO BE REMOVED Returns the value of TMD distributions for given (x,b) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa. I.e. $F(-h)=F(h)(5:-5:-1)$
`XX_inB(x,bT,h)`	real(-5:5)	Returns the value of optimal TMD distributions for given (x,b) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa. I.e. $F(-h)=F(h)(5:-5:-1)$
`XX_inB(x,bT,mu,zeta,h)`	real(-5:5)	Returns the value of evolved TMD distributions (see (8.9)) for given (x,b,μ,ζ) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa. I.e. $F(-h)=F(h)(5:-5:-1)$. μ is in GeV. ζ in GeV^2.
`XX_inKT(x,kT,h)`	real(-5:5)	Returns the value of optimal TMD distributions in momentum space (see sec.VIII D) for given (x,k_T) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa. I.e. $F(-h)=F(h)(5:-5:-1)$
`XX_inKT(x,bT,mu,zeta,h)`	real(-5:5)	Returns the value of evolved TMD distributions (see (8.9)) in momentum space (see sec.VIII D) for given (x,k_T,μ,ζ) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa. I.e. $F(-h)=F(h)(5:-5:-1)$. μ is in GeV. ζ in GeV^2.
`XX_TMM_G(x,mu,h)`	real(-5:5)	Returns the value of TMM $\mathcal{G}_{n,n}$ (see sec.VIII E) for given (x,μ) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa.
`XX_TMM_X(x,mu,h)`	real(-5:5)	Returns the value of TMM $\mathcal{G}_{n+1,n}$ (see sec.VIII E) for given (x,μ) (reals) and hadron h. The value is the vector over flavors. For $h<0$ the quark-distributions are replaced by anti-quark distributions and vice versa.

Functions unique for particular modules

Entry	Type	Description
`uPDF_uPDF(x1,x2, bt,mu,zeta,h1,h2)`	real(-5:5)	Returns the value of product $f_q f_{\bar{q}}$ evolved unpolarized TMD distributions for given (x,b) (reals) and hadrons h_1 and h_2. The value is the vector over flavors. If $h_{1,2}$ have the same sign the product is $f_q f_{\bar{q}}$, if $h_{1,2}$ have different signs the product is $f_q f_q$. This combination is used to compute Drell-Yan-type processes, and a bit faster than a direct multiplication.

A. _model modules

This module gives the expression for f_{NP}, $\mu_{\rm OPE}$ and b^*. It could be written by user, and one of modules that form a TMD model (together with models for TMD-evolution).
Making this module follow the interface

```
public::  ModelInitialization
public::  ModelUpdate
real(dp),public,dimension(-5:5)::  FNP
real(dp),public::  bSTAR
real(dp),public::  muOPE
```
If there is a twist-3 part of small-b matching, there are extra functions
```
real(dp),public,dimension(-5:5)::  FNP_tw3
real(dp),public::  bSTAR_tw3
real(dp),public::  muOPE_tw3
```
which contribute to the twist-3 term.

`ModelInitialization(NParray)`: this subroutine is called during the initialization of `XX` module. The `integer,intent(in)::NPlength(:)` is the size of the array of NP-parameters. Use this subroutine to make preparation of the model.

`ModelUpdate(NParray)`: this subroutine is called during the `XX_setNPparameters(NParray)`. The `real(dp),intent(in)::NParray(:)` is the array of NP-parameters passed to `TMDR_setNPparameters`. Use this subroutine to recompute model parameters (if needed), save NP-parameters (if needed), etc.

`FNP(x,b,hadron,lambdaNP)`: the function returns $f_{NP}^{f,h}(x,b,\lambda)$.

`bSTAR(b,x,y)`: the function returns $b^*(b)$ function.

`mu_OPE(b,x,y,c4)`: the function returns $\mu_{\rm OPE}(z,b)$ function.
Important: the module `XX_model` is compiled before `XX`, and thus can use only lower-level modules: `TMD_AD`, `QCDinput`, etc.

B. _OPE modules (tw2)

OPE modules compute the following convolution

$$[C\times f](x) = \int_x^1 \frac{dy}{y} C(y,\mathbf{L}) f\left(\frac{x}{y},\mu_{\rm OPE}\right) \tag{8.10}$$

$$= \frac{1}{x}\int_x^1 dy C(y,\mathbf{L}) F\left(\frac{x}{y},\mu_{\rm OPE}\right), \tag{8.11}$$

where $F(x)=xf(x)$ and

$$\mathbf{L} = \ln\left(\frac{\mu_{\rm OPE}^2 b^2}{4e^{-2\gamma_E}}\right). \tag{8.12}$$

OPE modules are initialized automatically from the corresponding TMD module.

Extra notes:

- `eeTMDFF` does not have `_OPE` sub-module, because its OPE is elementary, and written directly in the main code.
- The convolution is convergent, but the convolution kernel has singularity at $y\to 1$. To stabilize the integral, it is computed as

$$\int_x^1 dy(...) = \int_x^{y_C} dy(...) + \int_{y_C}^1 dy(...). \tag{8.13}$$

The first term is computed by the adaptive algorithm. The second term is computed by the approximate formulas

$$\int_{y_C}^{1} dy \frac{f(x/y)-f(x)}{1-y} = f(x/y_C) - f(x),$$
$$\int_{y_C}^{1} dy \frac{f(x/y)-f(x)}{1-y} \ln(1-y) = (f(x/y_C) - f(x))(\ln(1-y_C) - 1),$$
$$\int_{y_C}^{1} dy \frac{f(x/y)-f(x)}{1-y} \ln^2(1-y) = (f(x/y_C) - f(x))(2 - \ln(1-y_C) + 2\ln^2(1-y_C)),$$
$$\int_{y_C}^{1} dy f(x/y)g(y) = f(x)\int_{y_C}^{1} dy g(y),$$

where $g(y)$ is integrable, and the last integral is taken explicitly.

- The parameter $y_C = 1-10^{-4}$ (**for a moment**) is fixed. It provides the accuracy of the last integral (6-7 digits), which is far better than.
- For $x \sim y_C$ the accuracy drops. In this case, the values of y_C is recomputed by formula $y_C = (100+x)/101$ (i.e. it is 2-digits closer to 1 than x).
- **The definition of TMDFF does not perfectly match the definition of collinear FF. There is a relative factor $1/z^2$.** So the convolution reads

$$D(z,b) = \int_z^1 \frac{dy}{y} \mathbb{C}\left(\frac{z}{y}, \mathbf{L}_\mu\right) \frac{d(y,\mu)}{y^2} = \frac{1}{z^3}\int_z^1 dy [y^2 C]\,(y, \mathbf{L}_\mu)\, D(\frac{z}{y}, \mu), \tag{8.14}$$

where $D(z) = zd(z)$.

1. *Functions*

`XX`=name of module (e.g. `uTMDPDF`, `uTMDFF`, etc.)

Entry	Type	Description
`XX_OPE_Initialize(file,prefix)`	subroutine	Initialize the module from INI-file.
`XX_resetGrid()`	subroutine	Force recomputation of the grid. Only if griding is ON.
`XX_OPE_testGrid()`	subroutine	Run the test of grid (see sec.VIII B 4). Only if griding is ON.
`XX_OPE_SetPDFreplica(rep,h)`	subroutine	Switch the PDF replica to `rep`(int) for hadron `h` (int). Recompute the grid (if griding is ON). If the new replica is as before nothing is done.
`XX_OPE_SetScaleVariation(c4)`	subroutine	Change the scale variation parameter c_4 (it enters the definition of $\mu_{\rm OPE}$). Recompute the grid (if griding is ON). The parameter (real)`c4` is restricted to $0.1 < c_4 < 10$.
`XX_OPE_convolution(x,b,h,G)`	dp_real(-5:5)	Returns (8.10) for (x,b)=((real)`x`, (real)`b`), and hadron number (int)`h`. In the case of `NA` returns 1. If grid is computed restore from the grid. The flag (T/F)`G` is optional. It indicates the inclusion of gluon into the result. Note, that if gluons were not included into the grid they are zero.
`uTMDPDF_X0_AS(x,mu,mu0,h,G)`	dp_real(-5:5)	Computes the AS$[\mathcal{X}_0[f_1]]$ as defined in [????.????]. This is asymptotic term required to determine the second moment of TMDPDF. (x,μ,μ_0)=((real)`x`, (real)`mu`, (real)`mu0`), and hadron number (int)`h`. In the case of `NA` returns 0. The flag (T/F)`G` is optional. It indicates the inclusion of gluon into the result. For a moment this function is done only for uTMDPDF, but it will be updated to all twist-TMDs

2. *Model parameters.*

The OPE is fully deterministic formula, but it contains the free scale, and often one implement b^*-modification. These parameters are written in `model`-module. Namely:

- `bSTAR(bT,x,y)` = is the function used instead of b in **L** in eqn.(8.10).
- `muOPE(bt,x,y,c4)` = is the function used for $\mu_{\rm OPE}$.

In both cases, the variables x and y are defined as in eqn.(8.10). The parameter c_4 is the scale variation parameter.

3. *Scale variation.*

TO BE WRITTEN

4. *Grid*

For fitting procedure one often needs to evaluate TMDs multiple times. For example, for fit performed in [1] the evaluation of singe χ^2 entry requires $\sim 16 \times 10^6$ calls of `uTMDPDF_...` So, every call of `uTMDPDF_..` at NNLO order, requires ~ 200 calls of pdfs, depending on x, b and λ's, in order to evaluate the integral over y. In such situations, it is much faster to make a grid of TMD distributions for a given set of non-pertrubative parameters (i.e. the grid is in x and b), and then use this grid for the interpolation of TMD values. The calculation of a grid is not a very fast procedure, nonetheless, for large computation (number of TMD calls $> 10^4 - 10^5$) is more efficient.

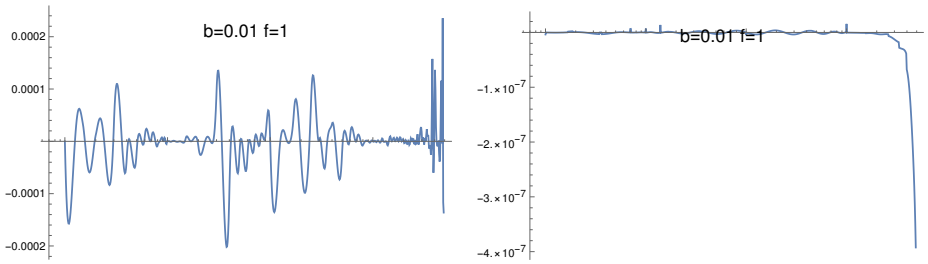


FIG. 4: Example of precision extraction from the grid. (left) with LHAPDF (right) with smooth input function (similar to PDF). Both cases the N^{3}LO convolution with matching coefficient is included.

The grid used in the artemide 3.0 is Chebyschev grid similar to the one described in [2112.09703]. Along each direction the phase-space is split into subgrids $[x_0, x_1, ..., x_n]$ ($x_n = 1$) and $[b_0, b_1, ..., b_k]$. Within each subgrid there some amount of node, distributed as $\ln(x)$ and $\ln(b)$ over Chebyschev points. The interpolation is done by barycentric formula.

- For $x < x_0$ and $x > 1$ the error is returned.
- For $b < b_0$ and $b > b_k$ the value at b_0 or b_k is returned.

All these parameters can be changed in `constants` file. However, we recommend to use default values, since they were checked. The check could be performed by the call `TestGrid()` subroutine of corresponding module. It computes values for central points of grids-cells from grid and from direct computation. This check can be also called by option in the `constants` file.

WARNING!
LHAPDF produces the non-smooth interpolation (based on cubic splines), while the Chebychev interpolation is much smoother. Therefore, at some points the difference between Chebyshev and LHAPDF can the significant (up to 50% and more if absolute value of function is small). Practically, one cannot push avarage precision better than 10^{-4}. In a bulk, it is acceptable. If the grid is tested at smooth input function one can easily reach precision $\sim 10^{-8}$.

Extra Notes:

- Grid is computed by parallel do (if OMP is enabled).

5. Technical notes

Convolution

The convolution integral reads

$$I(x) = \int_x^1 \frac{dz}{z} C(z) f\left(\frac{x}{z}\right) f_{NP}(x, z), \tag{8.15}$$

where the function $C(z)$ has a general form

$$C(z) = C_0 \delta(1-z) + (C_1(z))_+ + C_2(z). \tag{8.16}$$

Here the plus-distribution is undestood in the usual way

$$(C_1(z))_+ = C_1(z) - \delta(1-z) \int_0^1 dy C_1(y) dy. \tag{8.17}$$

In NNLO coefficient function there are only possible two $(..)_+$ terms, $1/(1-z)$ and $\ln(1-z)/(1-z)$.

In order to simplify the integration we rewrite

$$I(x) = \frac{1}{x}\int_x^1 dz C(z)\tilde{f}\left(\frac{x}{z}\right) f_{NP}(x,z), \qquad \tilde{f}(z) = zf(z). \tag{8.18}$$

Then the integral is split as

$$I(x) = \frac{1}{x}\Big\{ I_2(x) + C_0\tilde{f}\,(x)\, f_{NP}(x,1) + \tilde{f}\,(x)\, f_{NP}(x,1)\int_0^x dz C_1(z)\Big\}, \tag{8.19}$$

where

$$I_2(x) = \int_x^1 dz\Big[C_1(z)\left(\tilde{f}\left(\frac{x}{z}\right) f_{NP}(x,z) - \tilde{f}\,(x)\, f_{NP}(x,1)\right) + C_1(z)\tilde{f}\left(\frac{x}{z}\right) f_{NP}(x,z)\Big] \tag{8.20}$$

Presumably, the term $\sim C_0$ give the dominant contribution w, since it is ~ 1 whereas the other terms $\sim a_s$. Therefore, it serves as the estimation of the integral value, with respect to which the integration convergence is calculated. The convolution integral is evaluated by the G7K15 rule adaptively with given tolerance with respect to w.

The integral I is calculated in the procedure `Common_lowScale50` and `Common_lowScale5`, which is common for all twist-2 terms. The integral I_2 is calculated in the iterative procedures `MellinConvolutionVectorPart50` and `MellinConvolutionVectorPart5`, which is common for all twist-2 terms.

In the case of TMDFF we have the coefficient function with the structure $C(z)/z^2$ (plus-distribution, δ, etc, are multiplied by $1/z^2$), and collinear function $f(z)/z^2$. In this case it is convenient to rewrite

$$I(x) = \int_x^1 \frac{dz}{z}\frac{C(z)}{z^2}\frac{f\left(\frac{x}{z}\right)}{(x/z)^2} f_{NP}(x,z) = \frac{1}{x^3}\int_x^1 dz C(z)\hat{f}\left(\frac{x}{z}\right) f_{NP}(x,z), \qquad \hat{f}(z) = zf(z). \tag{8.21}$$

Since the common-code calculates the integral PDF-like convolution, I divide by factor $1/x^2$ all output of the common-code.

Coefficient functions

The coefficient function is spit to 6 terms.

$$C_{q\leftarrow q}, \quad C_{q\leftarrow g}, \quad C_{g\leftarrow q}, \quad C_{g\leftarrow g}, \quad C_{q\leftarrow \bar{q}}, \quad C_{q\leftarrow \bar{q}}, \quad C_{q\leftarrow q'}, \tag{8.22}$$

where $C_{q\leftarrow q}$ and $C_{q\leftarrow \bar{q}}$ contain *non-singlet* part only. The singlet contribution is given in $C_{q\leftarrow q'}$. So, the convolution for a quark (e.g. $q = u$) reads

$$F_u = C_{q\leftarrow q}\otimes q_u + C_{q\leftarrow g}\otimes q_g + C_{q\leftarrow \bar{q}}\otimes q_{\bar{u}} + C_{q\leftarrow q'}\otimes \sum_{f\in q,\bar{q}} q_f. \tag{8.23}$$

For gluon

$$F_g = C_{g\leftarrow g}\otimes q_g + C_{g\leftarrow q}\otimes \sum_{f\in q,\bar{q}} q_f. \tag{8.24}$$

The kernels are split into regular, singular and δ-parts, where singular contains $(\ln(1-x)/(1-x)))^n_+$ terms, δ contains only $\delta(1-x)$ terms, and regular is the rest. Singular and δ-terms are present only for $C_{q\leftarrow q}$ and $C_{g\leftarrow g}$.

The regular part is decomposed further, into singular (but integrable) terms at $x\to 0$ and $x\to 1$, and the finite part. The singular terms are computed exactly, while the finite part is fitted to linear combination of several functions. The basis of fit is such that the expressions are exact for coefficients which do not contain PolyLog functions. These are NLO, leading log [at all orders], large-Nf [at all orders], and some others. The full basis for the regular part is

$$\Big\{\underbrace{\ln\bar{x}, \ln^2\bar{x}, \ln^3\bar{x}, \ln^4\bar{x}, \ln^5\bar{x}, \frac{1}{x}, \frac{\ln x}{x}, \frac{\ln^2 x}{x}, \ln x, \ln^2 x, \ln^3 x, \ln^4 x, \ln^5 x,}_{\substack{\text{singular-integrable}\\ \text{exact}}} \tag{8.25}$$

$$\underbrace{1, x}_{\text{exact}}, x\bar{x}, x\left(\frac{\ln x}{1-x}+1\right), \frac{x\ln^2 x}{1-x}, \frac{x\ln^3 x}{1-x}, x\ln x, x\ln^2 x, x\ln^3 x, x\ln^4 x, x\bar{x}\ln x, x\bar{x}\ln^2 x, x\bar{x}\ln^3 x,$$

$$\bar{x}\ln\bar{x}, \bar{x}\ln^2\bar{x}, \bar{x}\ln^3\bar{x}, x\bar{x}\ln\bar{x}, x\bar{x}\ln^2\bar{x}, x\bar{x}\ln^3\bar{x}, \ln x\ln\bar{x}, x\ln x\ln\bar{x}, x\bar{x}\ln x\ln\bar{x}, \frac{\ln\bar{x}(1-x^2+2\ln x)}{1-x}, \frac{\bar{x}(\ln x + x)}{x}\Big\}.$$

Decomposed in this basis the coefficient functions are very presice. The typical relative precision is $10^{-8} - 10^{-10}$ *for the finite part* (other terms are exact). The worst presicion happen at small $x \sim 10^{-5}$ for $g \leftarrow g$ channel where relative precision is $\sim 10^{-5} - 10^{-6}$.

Similar ansatz is used for uTMDFF, with some term removed (which are sensitive to small-x region). Precision in the case of TMDFF is slightly worse $10^{-6} - 10^{-8}$. The worst precision is for N^{3}LO $q \to q'$ channel, where it is $\sim 10^{-4} - 10^{-6}$. However, it is very good precision which guaranties about 10-12 presice digits for the physical convolution.

Computation of AS$[\mathcal{X}_0]$

The function AS$\mathcal{X}_0$ is defined in [2402.01836] and reads

$$\begin{aligned}\mathrm{AS}[\mathcal{X}_0[F]] \;=\; & \alpha_s\Big\{P_1 + \alpha_s\left[P_2 + (C_1 - P_1)\otimes(P_1 - \beta_0)\right] \\ & +\alpha_s^2\Big[P_3 + (C_1 - P_1)\otimes(P_2 - \beta_1) + (C_2 - P_2)\otimes(P_1 - 2\beta_0) \\ & - (C_1 - P_1)\otimes(P_1 - \beta_0)\otimes(P_1 - 2\beta_0)\Big] + \mathcal{O}\left(\alpha_s^3\right)\Big\}\otimes f.\end{aligned} \tag{8.26}$$

I could be obtained by computation of ordinary optimal small-b convolution with replacement of finite terms by 0, and logarithm terms by $\mathbf{L}_b^k \to -k!$. Thus, the function can be computed as ordinary convolution with the coefficient function X defined as (up to N^{3}LO)

$$X = \frac{3}{4}C[\mathbf{L}_b = 0] - \frac{2}{3}C[\mathbf{L}_b = 1] - \frac{1}{12}C[\mathbf{L}_b = 4]. \tag{8.27}$$

C. `_OPE` modules (tw3)

OPE modules compute the following convolution

$$[C \times T](x) \;=\; \int [dy] C_{\mathrm{tw3}}(y_{1,2,3}), \mathbf{L}) T\left(y_{1,2,3}, \mu_{\mathrm{OPE}}\right), \tag{8.28}$$

where T is a twist-three function (qgq-correlator) and

$$\mathbf{L} = \ln\left(\frac{\mu_{\mathrm{OPE}}^2 b^2}{4e^{-2\gamma_E}}\right). \tag{8.29}$$

OPE modules are initialized automatically from the corresponding TMD module.

Twist-3 convolution is not implemented yet. All function are placeholders. The only working order is NA.

1. *Functions*

`XX`=name of module (e.g. `uTMDPDF`, `uTMDFF`, etc.)

Entry	Type	Description
`XX_OPE_Initialize(file,prefix)`	subroutine	Initialize the module from INI-file. (common with tw2-part)
`XX_tw3_resetGrid()`	subroutine	Force recomputation of the grid. Only if griding is ON.
`XX_OPE_tw3_testGrid()`	subroutine	Run the test of grid (see sec.VIII B 4). Only if griding is ON.
`XX_OPE_tw3_SetPDFreplica(rep,h)`	subroutine	Switch the PDF replica to `rep`(int) for hadron `h` (int). Recompute the grid (if griding is ON). If the new replica is as before nothing is done.
`XX_OPE_tw3_SetScaleVariation(c4)`	subroutine	Change the scale variation parameter c_4 (it enters the definition of $\mu_{\rm OPE}$). Recompute the grid (if griding is ON). The parameter (real)`c4` is restricted to $0.1 < c_4 < 10$.
`XX_OPE_tw3_convolution(x,b,h,G)`	dp_real(-5:5)	Returns (8.28) for (x,b)=((real)`x`, (real)`b`), and hadron number (int)`h`. In the case of `NA` returns 1. If grid is computed restore from the grid. The flag (T/F)`G` is optional. It indicates the inclusion of gluon into the result. Note, that if gluons were not included into the grid they are zero.

2. Model parameters.

The OPE is fully deterministic formula, but it contains the free scale, and often one implement b^*-modification. These parameters are written in `model`-module. Namely:

- `bSTAR(bT,x,y)` = is the function used instead of b in $\mathbf{L}$ in eqn.(8.10).
- `muOPE(bt,x,y,c4)` = is the function used for $\mu_{\rm OPE}$.

In both cases, the variables x and y are defined as in eqn.(8.10). The parameter c_4 is the scale variation parameter.

3. Scale variation.

Not implemented yet

4. Grid

Not implemented yet

D. Fourier to k_T space

We define

$$F(x,\boldsymbol{k}_T;\mu,\zeta) = \int \frac{d^2\boldsymbol{b}}{(2\pi)^2} F(x,\boldsymbol{b};\mu,\zeta) e^{-i(\boldsymbol{k}_T\boldsymbol{b})}. \tag{8.30}$$

Since all evaluated TMDs depends only on the modulus of $\boldsymbol{b}$, within the module we evaluate Hankel transformations. These transformations depends on the kind of TMD. Generally the integrals are

$$F(x,|\boldsymbol{k}_T|;\mu,\zeta) = \frac{M^{2n}}{n!}\int \frac{|b|d|\boldsymbol{b}|}{2\pi}\left(\frac{|\boldsymbol{b}|}{|k_T|}\right)^n J_n(|\boldsymbol{b}||\boldsymbol{k}_T|)F(x,|\boldsymbol{b}|;\mu,\zeta), \tag{8.31}$$

where the parameter M is the mass-scale (defined globally for all modules). The value of n dependents on the type of the TMD distributions. Particularly, there are

- For `uTMDPDF`, `uTMDFF`

$$n = 0, \qquad F(x, |\boldsymbol{k}_T|; \mu, \zeta) = \int \frac{d|\boldsymbol{b}|}{2\pi} |\boldsymbol{b}| J_0(|\boldsymbol{b}||\boldsymbol{k}_T|) F(x, |\boldsymbol{b}|; \mu, \zeta),$$

- For `SiversTMDPDF`, `BoerMuldersTMDPDF`, `wgtTMDPDF`

$$n = 1, \qquad F(x, |\boldsymbol{k}_T|; \mu, \zeta) = \frac{M^2}{k_T} \int \frac{d|\boldsymbol{b}|}{2\pi} |\boldsymbol{b}|^2 J_1(|\boldsymbol{b}||\boldsymbol{k}_T|) F(x, |\boldsymbol{b}|; \mu, \zeta),$$

- For `lpTMDPDF`

$$n = 2, \qquad F(x, |\boldsymbol{k}_T|; \mu, \zeta) = \frac{M^4}{2k_T^2} \int \frac{d|\boldsymbol{b}|}{2\pi} |\boldsymbol{b}|^3 J_2(|\boldsymbol{b}||\boldsymbol{k}_T|) F(x, |\boldsymbol{b}|; \mu, \zeta),$$

Clenshaw-Curtis-Levin algorithm. The following algorithm provides sufficient precision in the large range of k_T. It is a mixture of Clenshaw-Curtis quadrature and Levin differential-equation approach. First split the integral into ranges

$$\{B_0, ..., B_N\}, \tag{8.32}$$

where b_0 is sufficiently small (default $B_0 = 10^{-6}$) and B_n is sufficiently large (default $B_n \sim 25 - 100$). Then the integral is split as

$$F(q) = \int_0^\infty db J_0(bq) f(b) = \sum_{n=1}^{N} \int_{B_{n-1}}^{B_n} db J_0(bq) f(b) + \Delta_0 + \Delta_N, \tag{8.33}$$

where

$$\Delta_0 = \int_0^{B_0} db J_0(bq) f(b), \qquad \Delta_N = \int_{B_N}^\infty db J_0(bq) f(b).$$

Then for each sector one can estimate integral using different approaches. Here, I use the approaches that are quadratures over the nodes of Chebishev polynomials with

$$t_i = \cos \frac{i\pi}{K}, \tag{8.34}$$

where N is the size of the grid. So, for each interval $[B_{n-1}, B_n]$ one defines a grid with (K+1) points (ultimate points enters two sub-grids). The points are distributed logarithmically. So

$$b_{nk} = \exp\left(\delta_n t_k + \bar{b}_n\right), \qquad \delta_n = \frac{B_n - B_{n-1}}{2}, \qquad \bar{b}_n = \frac{B_n + B_{n-1}}{2}, \tag{8.35}$$

where $n \in [1, N]$ and $k \in [0, K]$. The integral is then defined as

$$\int_{B_{n-1}}^{B_n} db J_0(bq) f(b) = \sum_{k=0}^{K} w_{nk}(q) f(b_{nk}), \tag{8.36}$$

where $w_{nk}(q)$ is precomputed by different algorithm.

The *Clenshaw-Curtis* quadrature leads to

$$\int_{B_{n-1}}^{B_n} db J_0(bq) f(b) = \int_{-1}^{1} dt \delta_n b(t) J_0(b(t)) f(b(t)) = \sum_{k=0}^{K} w_{nk}^{\mathrm{CC}}(q) f(b_{nk}), \tag{8.37}$$

where

$$w_{nk}^{\mathrm{CC}}(q) = \delta_n b_{nk} J_0(b_{nk}) a_k, \qquad a_k = \frac{4\beta_k}{K} \sum_{l=0,2,..}^{K} \frac{\beta_l}{1 - l^2} \cos \frac{kl\pi}{K}, \tag{8.38}$$

with $\beta_i = 1/2$ for $i = 1$ or $i = K$ and $\beta_i = 1$ otherwise.

The *Levin* approach consists in the consideration of the differential system *(to be written later)*, which can be bootstraped on the grid, and leads to the system of linear equations. In case of present integral it requires the inversion of the Matrix

$$A(q) = \begin{pmatrix} \frac{D}{b\delta} & qI \\ -qI & \frac{D}{b\delta} - \frac{I}{b} \end{pmatrix}, \tag{8.39}$$

where D is the derivative matrix for the Chebyshev interpolation, and $1/b$ is understood as division by b_i for corresponding row. The inverse of this matrix is made by Crout's method (the routine is derived from `https://github.com/Beliavsky/Matrix_Inversion`). Then the weights are

$$w^{\mathrm{L}}_{nk}(q) = J_0(B_n q)A^{-1}_{1k} - J_0(B_{n-1}q)A^{-1}_{K+1,k} + J_1(B_n q)A^{-1}_{K+2,k} - J_1(B_{n-1}q)A^{-1}_{2K+2,k}. \tag{8.40}$$

Note, that it can be written as vector sum

$$\boldsymbol{w}^{\mathrm{L}}_{n}(q) = J_0(B_n q)\boldsymbol{A}^{-1}_{1} - J_0(B_{n-1}q)\boldsymbol{A}^{-1}_{K+1} + J_1(B_n q)\boldsymbol{A}^{-1}_{K+2} - J_1(B_{n-1}q)\boldsymbol{A}^{-1}_{2K+2}, \tag{8.41}$$

where $\boldsymbol{A}_k$ is [1,K] part of the k'th row.

This approaches are used in different ranges.

- *Oscillatory integral* If the Bessel function is already oscillating in the range. In this case the integral is computed by Levin method.
- *Stable integral* If the Bessel function is not yet oscillating in the range. In this case the Clenshaw-Curtis quadrature is used.

Basically, for small-(bq) one should use CC quadrature, and for other cases Levin quadrature. To distinguish these case I use the inequality

$$w_{nk}(q) = \begin{cases} w^{\mathrm{CC}}_{nk}(q), & \text{if } B_n q < j_0\alpha, \\ w^{\mathrm{L}}_{nk}(q), & \text{otherwise,} \end{cases} \tag{8.42}$$

where j_0 is the first zero of Bessel function, and α is a parameter of order 1 (default $\alpha = 1$).

Finally, the correction to the integral could be estimated. The tale correction is known [cite]

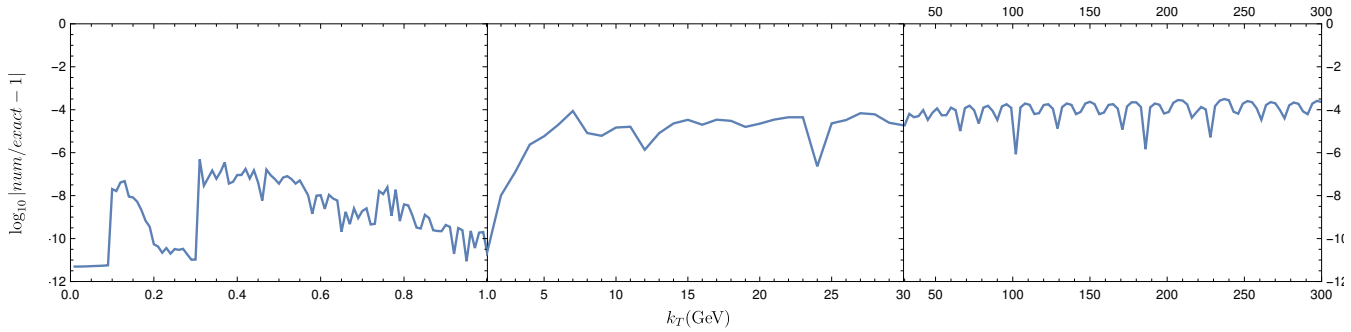
$$\Delta_N \sim f(B_N), \tag{8.43}$$

so using sufficiently large B_N we get very small correction. The first point correction can be estimated assuming smooth and non-singular behavior of $f(b)$

$$\Delta_0 = \int_0^{B_0} dbJ_0(bq)f(b) \simeq \int_0^{B_0} dbJ_0(bq)b\frac{f(B_0)}{B_0} = \frac{J_1(B_0 q)}{q}f(B_0) \simeq B_0\frac{f(B_0)}{2}. \tag{8.44}$$

Practically, I add ($n = 1$, $k = K$ corresponds to B_0 node)

$$w_{1K} \to w_{1K} + \frac{J_1(B_0 q)}{q}. \tag{8.45}$$



Example of Hankel transform. Test function is $f(b) = be^{-2.1b^2}(1 + 6.3b^2 + 1.4b^6)$. Shown is logarithm of ratio to the exact evaluation. Here the sub-grids are $B = \{0.00001, 0.01, 0.2, 2., 8., 25.\}$ with $K = 16$.

E. TMM

In ref.[2402.01836] [7] the following TMMs are defined

$$\mathcal{G}_{n,n}[F] = \mathcal{G}[F] = \frac{1}{n!}\int_0^\infty db\mu \left(\frac{\mu b}{2}\right)^n J_{n+1}(\mu b) F(x,b) \tag{8.46}$$

$$\begin{aligned}\mathcal{G}_{n+1,n}[F] = \mathcal{X}[F] &= \frac{1}{2M^2 n!}\int_0^\infty db\mu^3 \left(\frac{\mu b}{2}\right)^n \frac{(n+1)J_{n+1}(\mu b) - J_{n+3}(\mu b)}{n+2} F(x,b) \\ &= \frac{1}{2M^2 n!}\int_0^\infty db\mu^3 \left(\frac{\mu b}{2}\right)^n \left(J_{n+1}(\mu b) - \frac{J_{n+2}(\mu b)}{(\mu b)}\right) F(x,b)\end{aligned} \tag{8.47}$$

The correspond to the cut-integral over TMD and particular TMMs. The value of n is defined by the type of TMD (see sec.VIII D).

IX. TMDS MODULE

The module TMDs joins the lower modules and performs the evaluation of various TMD distributions in the ζ-prescription. Generally a TMD distribution is given by the expression

$$F_f(x,b;\mu,\zeta) = R_f[b,(\mu,\zeta)\to(\mu_i,\zeta_{\mu_i})]\tilde{F}_f(x,b), \tag{9.1}$$

where R is the TMD evolution kernel, $\tilde{F}$ is a TMD distribution at low scale. The scale μ_i is dependent on the evolution type, and could be out of use. Note, that TMDs initializes the lower modules automatically. Therefore, no special initializations should be done.

List of available commands

Command	Type	Sec.	Short description
`TMDs_Initialize(file,prefix)`	subrout.	-	Initialization of module. (string) `file` is the name of `constants-file`, which contains initialization information. (string)`prefix` is appended to `file` if provided.
`TMDs_SetScaleVariations(c1,c3)`	subrout.	IX D	Set new values for the scale-variation constants.
`uTMDPDF_5(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Unpolarized TMD PDF (gluon term undefined)
`uTMDPDF_50(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Unpolarized TMD PDF (gluon term defined)
`uTMDFF_5(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Unpolarized TMD FF (gluon term undefined)
`uTMDFF_50(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Unpolarized TMD FF (gluon term defined)
`lpTMDPDF_50(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Linearly polarized gluon TMD PDF (quarks are zero)
`SiversTMDPDF_5(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Sivers TMD PDF (gluon term undefined)
`SiversTMDPDF_50(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	Sivers TMD PDF (gluon term defined)
`wgtTMDPDF_5(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	worm-gear T TMD PDF (gluon term undefined)
`wgtTMDPDF_50(x,b,mu,zeta,h)`	(real*8(-5:5))	IX B	worm-gear T TMD PDF (gluon term defined)
`uTMDPDF_5(x,b,h)`	(real*8(-5:5))	IX B	Unpolarized TMD PDF at optimal line (gluon term undefined)
`uTMDPDF_50(x,b,,h)`	(real*8(-5:5))	IX B	Unpolarized TMD PDF at optimal line (gluon term defined)
`uTMDFF_5(x,b,h)`	(real*8(-5:5))	IX B	Unpolarized TMD FF at optimal line(gluon term undefined)
`uTMDFF_50(x,b,h)`	(real*8(-5:5))	IX B	Unpolarized TMD FF at optimal line (gluon term defined)
`lpTMDPDF_50(x,b,,h)`	(real*8(-5:5))	IX B	Linearly polarized gluon TMD PDF at optimal line (quarks are zero)
`SiversTMDPDF_5(x,b,h)`	(real*8(-5:5))	IX B	Sivers TMD PDF at optimal line (gluon term undefined)
`SiversTMDPDF_50(x,b,,h)`	(real*8(-5:5))	IX B	Sivers TMD PDF at optimal line (gluon term defined)
`wgtTMDPDF_5(x,b,h)`	(real*8(-5:5))	IX B	Worm gear T TMD PDF at optimal line (gluon term undefined)
`wgtTMDPDF_50(x,b,,h)`	(real*8(-5:5))	IX B	Worm gear T TMD PDF at optimal line (gluon term defined)
`BoerMuldersTMDPDF_5(x,b,h)`	(real*8(-5:5))	IX B	Sivers TMD PDF at optimal line (gluon term undefined)
`BoerMuldersTMDPDF_50(x,b,,h)`	(real*8(-5:5))	IX B	Sivers TMD PDF at optimal line (gluon term defined)
`uPDF_uPDF(x1,x2,b,mu,zeta,h1,h2)`	(real*8(-5:5))	IX C	Product of Unpolarized TMD PDF $f_{q\leftarrow h_1}(x_1)f_{\bar{q}\leftarrow h_1}$ at the same scale (gluon term undefined)

List of functions which must be provided by a model code

Input	Short description
`mu_LOW(b)`	The value of μ_i used in the evolutions of type 1 and 2 (improved $\mathcal{D}$ and γ). See [6].
`mu0(b)`	The value of μ_0 used in the evolution of type 1 (improved $\mathcal{D}$). See [6].

A. Definition of low-scales

The low scales μ_i and μ_0 are defined in the functions `mu_LOW(bt)` and `mu0(bt)` which can be found in the end of `TMDs.f90` code. Modify it if needed.

B. Evaluating TMDs

The expression for unpolarized TMD PDF is obtained by the functions
(real*8(-5:5))`uTMDPDF_5(x,b,mu,zeta,h)`
where

`x` (real*8) Bjorken-x $(0 < x < 1)$

`b` (real*8) Transverse distance $(b > 0)$ in GeV

`mu` (real*8) The scale μ_f in GeV. Typically, $\mu_f = Q$.

`zeta` (real*8) The scale ζ_f in GeV^2. Typically, $\zeta_f = Q^2$.

`h` (integer) The hadron type. If $h < 0$, the hadron is anti-hadron.

This function return the vector real*8(-5:5) for $\bar{b}, \bar{c}, \bar{s}, \bar{u}, \bar{d}, ?, d, u, s, c, b$.

- Gluon contribution in `uTMDPDF_5` is undefined, but taken into account in the mixing contribution. The point is that evaluation of gluons slow down the procedure approximately by 40%, and often is not needed. To calculate the full flavor vector with the gluon TMD, call `uTMDPDF_50(x,b,mu,zeta,h)`, where all arguments defined in the same way.
- The other TMDs, such as unpolarized TMDFF, transversity, etc. are obtained by similar function see the table in the beginning of the section.
- Each function has version without parameters `mu` and `zeta`. It corresponds to the evaluation of a TMS at optimal line [6]. Practically, it just transfers the outcome of corresponding TMD module, e.g.module `uTMDPDF`, see sec.**??**.
- For $h < 0$ the quark-distributions are replaced by anti-quark distributions and vice versa. I.e. $F(-h) = F(h)(5 : -5 : -1)$ [it is done in TMDs-Modules]

C. Products of TMDs

The the evaluation of majority of cross-sections one needs the product of two TMDs at the same scale. There are set of functions which return these products. They are slightly faster then just evaluation and multiplication, due to the flavor blindness of the TMD evolution. The function have common interface
(real*8(-5:5)) `uPDF_uPDF(x1,x2,b,mu,zeta,h1,h2)`
where

`x1,x2` (real*8) Bjorken-x's $(0 < x < 1)$

`b` (real*8) Transverse distance $(b > 0)$ in GeV

`mu` (real*8) The scale μ_f in GeV. Typically, $\mu_f = Q$.

`zeta` (real*8) The scale ζ_f in GeV^2. Typically, $\zeta_f = Q^2$.

`h1,h2` (integer) The hadron's types.

The function return a product of the form $F_{f_1 \leftarrow h_1}(x_1, b; \mu, \zeta) F_{f_2 \leftarrow h_2}(x_2, b; \mu, \zeta)$, where $f_{1,2}$ and the type of TMDs depend on the function.

D. Theoretical uncertainties

`TMDs_SetScaleVariations(c1,c3,c4)` changes the scale multiplicative factors c_i (see [6], sec.6). The default set of arguments is (1,1,1), i.e. the scales as they given in corresponding functions. This subroutine changes $c1$ and $c3$ constants and call corresponding subroutines for variation of $c4$ in TMD defining packages. Note, that in some types of evolution particular variations absent.

X. TMDF MODULE

This module evaluates the structure functions, that are universally defined as

$$F(Q^2, q_T, x_1, x_2, \mu, \zeta_1, \zeta_2) = \int_0^\infty \frac{bdb}{2} b^n J_n(bq_T) \sum_{ff'} z_{ff'}(Q^2) F_1^f(x_1, b; \mu, \zeta_1) F_2^{f'}(x_2, b; \mu, \zeta_2), \tag{10.1}$$

where

- Q^2 is hard scale.
- q_T is transverse momentum in the factorization frame. It coincides with measured q_T in center-mass frame for DY, but $q_T \sim p_T/z$ for SIDIS.
- x_1 and x_2 are parts of collinear parton momenta. I.e. for DY $x_{1,2} \simeq Qe^{\pm y}/\sqrt{s}$, while for SIDIS $x_2 \sim z$. It can also obtain power correction, ala Nachmann variables.
- μ is the hard factorization scale $\mu \sim Q$
- $\zeta_{1,2}$ are rapidity factorization scales. In the standard factorization scheme $\zeta_1\zeta_2 = Q^4$. It can also be modified by power corrections.
- f, f' are parton flavors.
- $z_{ff'}$ is the process related function. E.g. for photon DY on $p + \bar{p}$, $z_{ff'} = \delta_{ff'}|e_f|^2$.
- n The order of Bessel transformation is defined by structure function. E.g. for unpolarized DY $n = 1$. For SSA's $n = 1$. In general for angular modulation $\sim \cos(n\theta)$.
- $F_{1,2}^f$ TMD distribution (PDF or FF) of necessary polarization and flavor.

Some structure functions are multiplied by a constant with mass dimension (e.g. Sivers asymmetry). This constant (called global mass scale = M) is defined globally in the constants file.

List of available commands

Command	Type	Sec.	Short description
`TMDF_Initialize(file,prefix)`	subrout.	X A	Initialization of module. (string) `file` is the name of `constants-file`, which contains initialization information. (string)`prefix` is appended to `file` if provided.
`TMDF_ShowStatistic())`	subrout.	–	Print current statistic on the number of calls.
`TMDF_ResetCounters()`	subrout.	–	Reset intrinsic counters. E.g.release convergenceIS-lost state.
`TMDF_F(Q2,qT,x1,x2,mu,zeta1,zeta2,N)`	(real*8)	X B	Evaluates the structure function

A. Initialization

Prior the usage module is to be initialized (once per run) by
`call TMDF_Initialize(file)`
It reads `constants-file` and initialize it-self accordingly.

B. Evaluating Structure functions

The value of the structure function is obtained by
(real*8)`TMDF_F(Q2,qT,x1,x2,mu,zeta1,zeta2,N)`
where

`Q2` (real*8) hard scale in GeV^2.

`qT` (real*8) modulus of transverse momentum in the factorization frame in GeV, $q_T > 0$

`x1` (real*8) x passed to the first TMD distribution ($0 < x1 < 1$)

`x2` (real*8) x passed to the first TMD distribution ($0 < x2 < 1$)

`mu` (real*8) The hard scale μ in GeV. Typically, $\mu = Q$.

`zeta1` (real*8) The scale ζ_f in GeV^2 for the first TMD distribution. Typically, $\zeta_f = Q^2$.

`zeta2` (real*8) The scale ζ_f in GeV^2 for the second TMD distribution. Typically, $\zeta_f = Q^2$.

`N` (integer) The number of process.

The function returns the value of

$$F^N(Q^2, q_T, x_1, x_2, \mu, \zeta_1, \zeta_2) = \int_0^\infty \frac{bdb}{2} b^n J_n(bq_T) \sum_{ff'} z^N_{ff'}(Q^2) F_1^f(x_1, b; \mu, \zeta_1) F_2^{f'}(x_2, b; \mu, \zeta_2). \tag{10.2}$$

The parameter n depends on the argument `N` and uniformly defined as

$n = 0$	for `N` < 10000
$n = 1$	for `N` $\in [10000, 20000]$
$n = 2$	for `N` $\in [20000, 30000]$
$n = 3$	for `N` > 30000

The particular values of $z_{ff'}$ and $F_{1,2}$ are given in the following table. User function can be implemented by code modification.

Notes on the integral evaluation:

- The integral is uniformly set to 0 for $q_T < 10^{-7}$.
- The integrand is uniformly set to 0 for $b > 10^3$.
- If for any element of evaluation (including TMD evolution factors and convolution integrals, and the integral of the structure function it-self) obtained divergent value. The trigger is set to ON. In this case, the integral returns uniformly large value 10^{100} for all integrals without evaluation. The trigger is reset by new values of NP parameters. It is done in order to cut the improper values of NP parameters in the fastest possible way, which speed up fitting procedures.
- The Fourier is made by Ogata quadrature, which is double exponential quadrature. I.e.

$$\int_0^\infty \frac{bdb}{2} b^n I(b) J_n(q_T b) \simeq \frac{1}{q_T^{n+2}} \sum_{k=1}^{\infty} \tilde{\omega}_{nk} b_{nk}^{n+1} I\left(\frac{b_{nk}}{q_T}\right), \tag{10.3}$$

  where

$$b_{nk} = \frac{\psi(\tilde{h}\tilde{\xi}_{nk})}{\tilde{h}} = \tilde{\xi}_{nk} \tanh(\frac{\pi}{2} \sinh(\tilde{h}\tilde{\xi}_{nk}))$$

$$\tilde{\omega}_{nk} = \frac{J_n(\tilde{b}_{nk})}{\tilde{\xi}_{nk} J^2_{n+1}(\tilde{\xi}_{nk})} \psi'(\tilde{h}\tilde{\xi}_{nk})$$

  Here, $\tilde{\xi}_{nk}$ is k'th zero of $J_n(x)$ function. Note, that $\tilde{h} = h/\pi$ in the original Ogata's notation.
- The convergence properties of the quadrature essentially depends on h. I have found that $h \sim q_T$. For accurate evolution of integrals at different values of q_T I set $h = h_0 s$, where s is defined for intervals of q_T.
- The sum over k is restricted by $N_{\max}$ where $N_{\max}$ is hard coded number, $N_{\max} = 200$.

- The sum over k is evaluated until the sum of absolute values of last four terms is less then *tolerance*. If $M > N_{\max}$ the integral declared divergent, and the trigger is set to ON.

WARNING!
The error for Ogata quadrature is defined by parameters h and M(number of terms in the sum over k). In the parameter M quadrature is double-exponential, i.e. converges fast as M approaches $N_{\max}$. And the convergence of the sum can be simply checked. In the parameter h the quadrature is quadratic (the convergence is rather poor). The convergence to the true value of integral is very expensive especially at large q_T (it requires the complete reevaluation of integral at all nodes). Unfortunately, $N_{\max} \sim h^{-1}$, and has to find the balance value for h. In principle, TMD functions decays rather fast, and suggested default value $h = 0.005$ is trustful. **Nonetheless, we suggest to test other values (*/2) of h to validate the obtained values in your model.**
The adaptive check of convergence will implemented in the future versions.

C. Enumeration of structure functions

List of enumeration of structures functions
N<10000

N	$z_{ff'}$	F_1	F_2	Short description	Gluon	LP	KPC
0 9999 9998	–	–	–	Test cases (see X D)	no	✓	✓
1	$\delta_{\bar{f}f'}\lvert e_f\rvert^2$	f_1	f_1	(unpol.)$h_1 + h_2 \to \gamma$	no	✓	✓
2	$\delta_{\bar{f}f'}\frac{(1-2\lvert ef\rvert s_w^2)^2 - 4e_f^2 s_w^4}{8 s_w^2 c_w^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to Z$	no	✓	✓
3	$\delta_{\bar{f}f'}\frac{z_{ll'} z_{ff'}}{\alpha_{\rm em}^2}$ given in (2.8) of [1]	f_1	f_1	(unpol.)$h_1 + h_2 \to Z + \gamma$	no	✓	✓
4	$\frac{1}{4s_w^2}\frac{\lvert V_{ff'}\rvert^2}{4s_w^2}\frac{Q^4}{(Q^2 - M_W^2)^2 + \Gamma_W^2 M_W^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to W^+$	no	✓	
5	$\frac{1}{4s_w^2}\frac{\lvert V_{ff'}\rvert^2}{4s_w^2}\frac{Q^4}{(Q^2 - M_W^2)^2 + \Gamma_W^2 M_W^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to W^-$	no	✓	
6	$\frac{1}{4s_w^2}\frac{\lvert V_{ff'}\rvert^2}{4s_w^2}\frac{Q^4}{(Q^2 - M_W^2)^2 + \Gamma_W^2 M_W^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to W^\pm$	no	✓	
7	$\frac{\lvert V_{ff'}\rvert^2}{4s_w^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to W^+$ (for narrow-width approx.)	no	✓	
8	$\frac{\lvert V_{ff'}\rvert^2}{4s_w^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to W^-$ (for narrow-width approx.)	no	✓	
9	$\frac{\lvert V_{ff'}\rvert^2}{4s_w^2}$	f_1	f_1	(unpol.)$h_1 + h_2 \to W^\pm$ (for narrow-width approx.)	no	✓	
10	$f_g(x_1)f_g(x_2) + h_1^\perp(x_1)h_1^\perp(x_2)$	-	-	(unpol.)$h_1 + h_2 \to H$ Elementary Higgs production	yes	✓	

11	$f_g(x_1)f_g(x_2)$	f_1	f_1	(unpol.)$h_1 + h_2 \to H$ Elementary Higgs production. Only unpol.TMDPDF term	yes	✓	
12	$h_{1g}^{\perp}(x_1)h_{1g}^{\perp}(x_2)$	$h_1^{\perp}$	$h_1^{\perp}$	(unpol.)$h_1 + h_2 \to H$ Elementary Higgs production. Only linearly pol.TMDPDF term	yes	✓	
20	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	f_1	f_1	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim f_1 f_1$ part of the angular coefficient Σ_0. Also part the part of cut Z-boson production $\sim \mathcal{P}_0$	yes		✓
21	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	f_1	f_1	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim f_1 f_1$ part of the angular coefficient Σ_1. Also part the part of cut Z-boson production $\sim \mathcal{P}_1$	yes		✓
22	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	f_1	f_1	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim f_1 f_1$ part of the angular coefficient Σ_2. Also part the part of cut Z-boson production $\sim \mathcal{P}_2$	yes		✓
23	$\Delta^{GG'} z_{-\ell}^{GG'} z_{-f}^{GG'} f_1 f_1$	f_1	f_1	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim f_1 f_1$ part of the angular coefficient Σ_3. Also part the part of cut Z-boson production $\sim \mathcal{P}_3$	yes		✓
24	$\Delta^{GG'} z_{-\ell}^{GG'} z_{-f}^{GG'} f_1 f_1$	f_1	f_1	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim f_1 f_1$ part of the angular coefficient Σ_4. Also part the part of cut Z-boson production $\sim \mathcal{P}_4$	yes		✓
29	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	f_1	f_1	(unpol.)$h_1+h_2 \to Z+\gamma$; $\sim f_1 f_1$ part of the angular coefficient $\Sigma_{UU} + \Sigma_0/2$. Used in normalization of some angular coefficients.	yes		✓
30	$\Delta^{GG'} z_{+\ell}^{GG'} r_{+f}^{GG'} h_1^{\perp} h_1^{\perp}$	$h_1^{\perp}$	$h_1^{\perp}$	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim h_1^{\perp} h_1^{\perp}$ part of the angular coefficient Σ_0. Also part the part of cut Z-boson production $\sim \mathcal{P}_0$	yes		✓
31	$\Delta^{GG'} z_{+\ell}^{GG'} r_{+f}^{GG'} h_1^{\perp} h_1^{\perp}$	$h_1^{\perp}$	$h_1^{\perp}$	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim h_1^{\perp} h_1^{\perp}$ part of the angular coefficient Σ_1. Also part the part of cut Z-boson production $\sim \mathcal{P}_1$	yes		✓
32	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} h_1^{\perp} h_1^{\perp}$	$h_1^{\perp}$	$h_1^{\perp}$	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim h_1^{\perp} h_1^{\perp}$ part of the angular coefficient Σ_2. Also part the part of cut Z-boson production $\sim \mathcal{P}_2$	yes		✓
35	$\Delta^{GG'} z_{+\ell}^{GG'} r_{-f}^{GG'} h_1^{\perp} h_1^{\perp}$	$h_1^{\perp}$	$h_1^{\perp}$	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim h_1^{\perp} h_1^{\perp}$ part of the angular coefficient Σ_5. Also part the part of cut Z-boson production $\sim \mathcal{P}_5$	yes		✓
36	$\Delta^{GG'} z_{+\ell}^{GG'} r_{-f}^{GG'} h_1^{\perp} h_1^{\perp}$	$h_1^{\perp}$	$h_1^{\perp}$	(unpol.)$h_1 + h_2 \to Z + \gamma$; $\sim h_1^{\perp} h_1^{\perp}$ part of the angular coefficient Σ_6. Also part the part of cut Z-boson production $\sim \mathcal{P}_6$	yes		✓
101	$R_{\bar{f}f'}^{Cu} e_f e_{f'}$ see (3.1) of [1]	f_1	f_1^{Cu}	(unpol.)$h_1 + Cu \to \gamma$ where the isostructure of Cu is simulated from $hadron = 1$, (used to describe E288 experiment)	no	✓	✓

102	$R^{^2H}_{\bar{f}f'} e_f e_{f'}$ see (3.1) of [1] with $Z=1$ and $A=2$	f_1	$f_1^{^2H}$	(unpol.)$h_1 +^2 H \to \gamma$ where thes isostructure of ^2H is simulated from $hadron=1$, used to describe E772 experiment	no	✓	✓
103	$R^{W}_{ff'} e_f e_{f'}$ see (3.1) of [1] with $Z=74$ and $A=183$	f_1	f_1^W	(unpol.)$h_1 + W \to \gamma$, where the isostructure of W(tungsten) is simulated from $hadron = 1$, used to describe E537 experiment	no	✓	✓
2001	$\delta_{ff'}\|e_f\|^2$	f_1	d_1	(unpol)$h_1 + \gamma \to h_2$	no	✓	
2002	$\delta_{ff'}\|e_f\|^2$	f_1^d	d_1	(unpol)$d^{(h_1)} + \gamma \to h_2$. Here, $d^{(h_1)}$ is the isoscalar target prepared from the hadron h_1. I.e. $u, d \to (u+d)/2$. For h_1=proton, this simulates the deuteron target.	no	✓	
2003	$\delta_{f\bar{f}'}\|e_f\|^2$	f_1^n	d_1	(unpol)$n^{(h_1)} + \gamma \to h_2$ Here, $n^{(h_1)}$ is the iso-conjugated target prepared from the hadron h_1. I.e. $u \leftrightarrow d$. For h_1=proton, this simulates the neutron target.	no	✓	
2101	$\delta_{ff'}\|e_f\|^2$	f_1	$d_1^{h_{1+2}}$	(unpol)$h_1 + \gamma \to h_{1+2}$, where $h_{1+2} = 1+2$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2102	$\delta_{ff'}\|e_f\|^2$	f_1	$d_1^{h_{1+2+3}}$	(unpol)$h_1 + \gamma \to h_{1+2+3}$, where $h_{1+2+3} = 1+2+3$ Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2103	$\delta_{ff'}\|e_f\|^2$	f_1^d	$d_1^{h_{1+2}}$	(unpol)$d^{(h_1)} + \gamma \to h_{1+2}$, where $h_{1+2} = 1+2$ and d is isoscalar target $u, d \to (u+d)/2$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2104	$\delta_{ff'}\|e_f\|^2$	f_1^d	$d_1^{h_{1+2+3}}$	(unpol)$d^{(h_1)} + \gamma \to h_{1+2+3}$, where $h_{1+2+3} = 1+2+3$ and d is isoscalar target $u, d \to (u+d)/2$. Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2105	$\delta_{ff'}\|e_f\|^2$	f_1^n	$d_1^{h_{1+2}}$	(unpol)$n^{(h_1)} + \gamma \to h_{1+2}$, where $h_{1+2} = 1+2$ and n is neutron target ($u \leftrightarrow d$). Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2106	$\delta_{ff'}\|e_f\|^2$	f_1^n	$d_1^{h_{1+2+3}}$	(unpol)$n^{(h_1)} + \gamma \to h_{1+2+3}$, where $h_{1+2+3} = 1+2+3$ and n is neutron target ($u \leftrightarrow d$). Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value).	no	✓	

2107	$\delta_{ff'}\|e_f\|^2$	f_1	$d_1^{h_{1+2}}$	(unpol)$h_1+\gamma\to h_{1+2}$, where $h_{1+2}=1+2$. Same as 2101 but computed strictly with produced hadron **3 and 4** (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2108	$\delta_{ff'}\|e_f\|^2$	f_1^d	$d_1^{h_{1+2}}$	(unpol)$d^{(h_1)}+\gamma\to h_{1+2}$, where $h_{1+2}=1+2$ and d is isoscalar target $u,d\to(u+d)/2$. Same as 2103 but computed strictly with produced hadron **3 and 4** (accounts only the sign of h_2 argument ignoring its value).	no	✓	
2109	$\delta_{ff'}\|e_f\|^2$	f_1^n	$d_1^{h_{1+2}}$	(unpol)$n^{(h_1)}+\gamma\to h_{1+2}$, where $h_{1+2}=1+2$ and n is neutron target ($u\leftrightarrow d$). Same as 2105 but computed strictly with produced hadron **3 and 4** (accounts only the sign of h_2 argument ignoring its value).	no	✓	

10000⩽N<20000

N	$z_{ff'}$	F_1	F_2	Short description	Gluon	LP	KPC
10000						✓	
19999	–	–	–	Test cases (see X D)	no	✓	
19998						✓	
10001	$-M\delta_{\bar f f'}\|e_f\|^2$	$-(f_{1T}^\perp)^p$	f_1	(Sivers)$h_1^\uparrow+h_2\to\gamma$	no	✓	
10003	$-M\delta_{\bar f f'}\frac{z_{ll'}z_{ff'}}{\alpha_{\rm em}^2}$ given in (2.8) of [1]	$-(f_{1T}^\perp)^p$	f_1	(Sivers)$h_1^\uparrow+h_2\to Z+\gamma$	no	✓	
10004	$-M\frac{1}{4s_w^2}\frac{\|V_{ff'}\|^2}{4s_w^2}\frac{Q^4}{(Q^2-M_W^2)^2+\Gamma_W^2M_W^2}$	$-(f_{1T}^\perp)^p$	f_1	(Sivers)$h_1^\uparrow+h_2\to W^+$	no	✓	
10008	$-M\frac{1}{4s_w^2}\frac{\|V_{ff'}\|^2}{4s_w^2}\frac{Q^4}{(Q^2-M_W^2)^2+\Gamma_W^2M_W^2}$	$-(f_{1T}^\perp)^p$	f_1	(Sivers)$h_1^\uparrow+h_2\to W^-$	no	✓	
10011	$+M\delta_{\bar f f'}\|e_f\|^2$	f_1^h	$-(f_{1T}^\perp)^p$	(Sivers)$h+p^\uparrow\to\gamma$	no	✓	
12001	$-M\delta_{ff'}\|e_f\|^2$	$(f_{1T}^\perp)^{h_1}$	$d_1^{h_2}$	(Sivers)$h_1^\uparrow+\gamma\to h_2$	no	✓	
12002	$-M\delta_{ff'}\|e_f\|^2$	$(f_{1T}^\perp)^d$	$d_1^{h_2}$	(Sivers)$d^\uparrow+\gamma\to h_2$. Here, d is isoscalar target $(p+n)_{h_1}/2$.	no	✓	
12003	$-M\delta_{ff'}\|e_f\|^2$	$(f_{1T}^\perp)^n$	$d_1^{h_N}$	(Sivers)$n^\uparrow+\gamma\to h_N$. Here, n is neutron target $n=h_1(u\leftrightarrow d)$.	no	✓	
12101	$-M\delta_{ff'}\|e_f\|^2$	$(f_{1T}^\perp)^p$	$d_1^{h_{1+2}}$	(Sivers)$p^\uparrow+\gamma\to h_{1+2}$, where $h_{1+2}=h_1+h_2$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	

12102	$-M\delta_{ff'}\lvert e_f\rvert^2$	$(f_{1T}^{\perp})^p$	$d_1^{h_{1+2+3}}$	(Sivers)$p^{\uparrow}+\gamma\to h_{1+2+3}$, where $h_{1+2+3}=h_1+h_2+h_3$. Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
12103	$-M\delta_{ff'}\lvert e_f\rvert^2$	$(f_{1T}^{\perp})^d$	$d_1^{h_{1+2}}$	(Sivers)$d^{\uparrow}+\gamma\to h_{1+2}$, where $h_{1+2}=h_1+h_2$ and d is isoscalar target $(p+n)/2$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
12104	$-M\delta_{ff'}\lvert e_f\rvert^2$	$(f_{1T}^{\perp})^d$	$d_1^{h_{1+2+3}}$	(Sivers)$d^{\uparrow}+\gamma\to h_{1+2+3}$, where $h_{1+2+3}=h_1+h_2+h_3$ and d is isoscalar target $(p+n)/2$. Computed strictly with produced hadron 1,2 and 3 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
12105	$-M\delta_{ff'}\lvert e_f\rvert^2$	$(f_{1T}^{\perp})^n$	$d_1^{h_{1+2}}$	(Sivers)$n^{\uparrow}+\gamma\to h_{1+2}$, where $h_{1+2}=h_1+h_2$ and n is neutron target $n=p(u\leftrightarrow d)$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
12106	$-M\delta_{ff'}\lvert e_f\rvert^2$	$(f_{1T}^{\perp})^n$	$d_1^{h_{1+2+3}}$	(Sivers)$n^{\uparrow}+\gamma\to h_{1+2+3}$, where $h_{1+2+3}=h_1+h_2+h_3$ and n is neutron target $n=p(u\leftrightarrow d)$. Computed strictly with produced hadron 1,3 and 2 (accounts only the sign of h_2 argument ignoring its value).	no	✓	
$\downarrow\downarrow\downarrow$ Enumeration for F_{LT} in SIDIS is same as for F_{UT} with $(12\to 13)$ $\downarrow\downarrow\downarrow$							
13001	$+M\delta_{ff'}\lvert e_f\rvert^2$	$(g_{1T}^{\perp})^{h_1}$	$d_1^{h_2}$	(wgt)$h_1^{\uparrow}+\gamma\to h_1$	no	✓	
13002	$+M\delta_{ff'}\lvert e_f\rvert^2$	$(g_{1T}^{\perp})^d$	$d_1^{h_2}$	(wgt)$d^{\uparrow}+\gamma\to h_2$. Here, d is isoscalar target $(p+n)_{h_1}/2$.	no	✓	
13003	$+M\delta_{ff'}\lvert e_f\rvert^2$	$(g_{1T}^{\perp})^d$	$d_1^{h_2}$	(wgt)$n^{\uparrow}+\gamma\to h_2$. Here, n is neutron target $n=h_1(u\leftrightarrow d)$.	no	✓	
13101	$+M\delta_{ff'}\lvert e_f\rvert^2$	$(g_{1T}^{\perp})^h$	$d_1^{h_{1+2}}$	(wgt)$p^{\uparrow}+\gamma\to h_{1+2}$, where $h_{1+2}=1+2$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value)	no	✓	
13102	$+M\delta_{ff'}\lvert e_f\rvert^2$	$(g_{1T}^{\perp})^h$	$d_1^{h_{1+2+3}}$	(wgt)$p^{\uparrow}+\gamma\to h_{1+2+3}$, where $h_{1+2+3}=h_1+h_2+h_3$. Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value)	no	✓	
13103	$+M\delta_{ff'}\lvert e_f\rvert^2$	$(g_{1T}^{\perp})^d$	$d_1^{h_{1+2}}$	(wgt)$d^{\uparrow}+\gamma\to h_{1+2}$, where $h_{1+2}=h_1+h_2$ and d is isoscalar target $(p+n)_h/2$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value)	no	✓	

13104	$+M\delta_{ff'}\|e_f\|^2$	$(g_{1T}^{\perp})^d$	$d_1^{h_{1+2+3}}$	(wgt)$d^{\uparrow}+\gamma\to h_{1+2+3}$, where $h_{1+2+3}=h_1+h_2+h_3$ and d is isoscalar target $(p+n)/2$. Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value)	no	✓	
13105	$+M\delta_{ff'}\|e_f\|^2$	$(g_{1T}^{\perp})^n$	$d_1^{h_{1+2}}$	(wgt)$n^{\uparrow}+\gamma\to h_{1+2}$, where $h_{1+2}=h_1+h_2$ and n is neutron target $n=p(u\leftrightarrow d)$. Computed strictly with produced hadron 1 and 2 (accounts only the sign of h_2 argument ignoring its value)	no	✓	
13106	$+M\delta_{ff'}\|e_f\|^2$	$(f_{1T}^{\perp})^n$	$d_1^{h_{1+2+3}}$	(wgt)$n^{\uparrow}+\gamma\to h_{1+2+3}$, where $h_{1+2+3}=h_1+h_2+h_3$ and n is neutron target $n=p(u\leftrightarrow d)$.Computed strictly with produced hadron 1, 2 and 3 (accounts only the sign of h_2 argument ignoring its value)	no	✓	
13200	$+M\delta_{\bar{f}f'}\frac{\bar{z}_{ll'}z_{ff'}}{\alpha_{\rm em}^2}$ given in (2.8) of [1]	$(g_{1T}^{\perp})^p$	$(-1)^f f_1$	G_{TU}^1 cross-section in $h_1^{\uparrow}+h_2\to Z+\gamma$	no	✓	
13201	$+M\frac{1}{4s_w^2}\frac{\|V_{ff'}\|^2}{4s_w^2}\frac{Q^4}{(Q^2-M_W^2)^2+\Gamma_W^2M_W^2}$	$(g_{1T}^{\perp})^p$	$(-1)^f f_1$	G_{TU}^1 cross-section in $h_1^{\uparrow}+h_2\to W^+$	no	✓	
13202	$+M\frac{1}{4s_w^2}\frac{\|V_{ff'}\|^2}{4s_w^2}\frac{Q^4}{(Q^2-M_W^2)^2+\Gamma_W^2M_W^2}$	$(g_{1T}^{\perp})^p$	$(-1)^f f_1$	G_{TU}^1 cross-section in $h_1^{\uparrow}+h_2\to W^-$	no	✓	

20000⩽N<30000

N	$z_{ff'}$	F_1	F_2	Short description	Gluon req.
20000 29999 29998	–	–	–	Test cases (see X D)	no

30000⩽N

N	$z_{ff'}$	F_1	F_2	Short description	Gluon req.
30000 39999 39998	–	–	–	Test cases (see X D)	no

Notes on parameters and notation:
s_w^2, M_Z, Γ_Z, etc. are defined in `constants`
f_1-unpolarized TMDPDF
d_1-unpolarized TMDFF
$f_{1T}^{\perp}$-Sivers function
$g_{1T}^{\perp}$-wgt-function
$(-1)^f$ is (+1) for f=quark, and (-1) for f=anti-quark.
Everywhere proton is expected to be hadron number 1.

D. Tests

The test options are called by process number $N+0$, $N+9999$ and $N+9998$. In these case, the integrand set by a fixed function. In the case $N+0$

$$N+0: \qquad \sum zFF \to e^{-0.2b}. \tag{10.4}$$

The cases $N+9999$ and $N+9998$ are dependent on parameters of the function:

$$N+9999: \qquad \sum zFF \to e^{-\mu b}(1+x_1 b^2+x_2 b^4), \tag{10.5}$$

$$N+9998: \qquad \sum zFF \to e^{-\mu b^2}(1+x_1 b^2+x_2 b^4). \tag{10.6}$$

In all cases the Hankel integral could be evaluated. They read

$$9999: \qquad F=\frac{1}{2\mu^2(1+X)^{3/2}}\left(1+\frac{x_1}{\mu^2}\frac{6-9X}{(1+X)^2}+\frac{15x_2}{\mu^4}\frac{8-40X+15X^2}{(1+X)^4}\right), \tag{10.7}$$

$$19999: \qquad F=\frac{3\sqrt{X}}{2\mu^3(1+X)^{5/2}}\left(1+\frac{5x_1}{\mu^2}\frac{4-3X}{(1+X)^2}+\frac{105x_2}{\mu^4}\frac{8-20X+5X^2}{(1+X)^4}\right), \tag{10.8}$$

$$29999: \qquad F=\frac{15X}{2\mu^4(1+X)^{7/2}}\left(1+\frac{21x_1}{\mu^2}\frac{2-X}{(1+X)^2}+\frac{63x_2}{\mu^4}\frac{48-80X+15X^2}{(1+X)^4}\right), \tag{10.9}$$

$$39999: \qquad F=\frac{105X^{3/2}}{2\mu^5(1+X)^{9/2}}\left(1+\frac{9x_1}{\mu^2}\frac{8-3X}{(1+X)^2}+\frac{495x_2}{\mu^4}\frac{16-20X+3X^2}{(1+X)^4}\right), \tag{10.10}$$

where $X=q_T^2/\mu^2$.

$$9998: \qquad F=\frac{e^{-Y}}{4\mu}\left(1+\frac{x_1}{\mu}(1-Y)+\frac{x_2}{\mu^2}(2-4Y+Y^2)\right), \tag{10.11}$$

$$19998: \qquad F=\frac{e^{-Y}\sqrt{Y}}{4\mu^{3/2}}\left(1+\frac{x_1}{\mu}(2-Y)+\frac{x_2}{\mu^2}(6-6Y+Y^2)\right), \tag{10.12}$$

$$29998: \qquad F=\frac{e^{-Y}Y}{4\mu^2}\left(1+\frac{x_1}{\mu}(3-Y)+\frac{x_2}{\mu^2}(12-8Y+Y^2)\right), \tag{10.13}$$

$$39998: \qquad F=\frac{e^{-Y}Y^{3/2}}{4\mu^{5/2}}\left(1+\frac{x_1}{\mu}(4-Y)+\frac{x_2}{\mu^2}(20-10Y+Y^2)\right), \tag{10.14}$$

where $Y=q_T^2/(4\mu)$.

XI. TMDX_DY MODULE

This module evaluates cross-sections with the Drell-Yan-like kinematics. I.e. it expects the following kinematic input, (s, Q, y) which defines the variables x's, etc.

The general structure of the cross-section is

$$\Delta\sigma(q_T) = \int_{bin} dX \; prefactor2 \; \times \sum_n F_n, \tag{11.1}$$

where

$$dX = dQ^2 dy dq_T^2.$$

So, for a small bin

$$\Delta\sigma(q_T) \approx \Delta X \frac{d\sigma}{dX}, \qquad \Delta X = (Q^2_{\max} - Q^2_{\min})(y_{\max} - y_{\min})(q^2_{T\max} - q^2_{T\min}). \tag{11.2}$$

Prefactor2 accumulates the definition of the phase-space, and general process. It has the form

$$prefactor2 = (\text{phase-space Jacobian}) \times H(Q, \mu_H) \tag{11.3}$$

The F_n is a contribution specific for a channel. Some processes have several channels, which could be computed together or separately (e.g. $\sim f_1 f_1$ and $\sim h_1^\perp h_1^\perp$). Generally it has a form

$$F_n = (\text{cuts for lepton pair})_n \times F_n(Q, q_T, x_1, x_2, \mu, \zeta_1, \zeta_2), \tag{11.4}$$

where F is a product of TMDs and corresponding weight defined in (10.1) or (13.1). There are following feature of current implementation

- In the current version the scaling variables are set as

  $$\mu^2 = \zeta_1 = \zeta_2 = Q^2. \tag{11.5}$$

  Currently, it is hard coded and could not be easily modified. However, there is a possibility to vary the value of μ as $\mu = c_2 Q$, where c_2 is variation parameter (see sec.XI E).

- For the definition of *(cuts for lepton pair)*-function see [1]. It is evaluated within module `LeptonCutsDY.f90`. The presence of cuts, and their parameters are set by the `SetCuts` subroutine.

- The expression for the hard factor H is taken from [8]. It is the function of $\ln(Q/\mu_H)$ and $a_s(\mu_H)$). Since in the current realization $\mu_H = Q$, the logarithm is replaced by $\ln(c_2)$, where c_2 is the variation constant.

The cross-section can be computed by two different means. Using LP factorization or using KPC-resummed factorization. The selection is done by the parameter A.p2. One behavior totally exclude another.

This section is to be updated by definition of kinematics

List of available commands

Command	Type	Sec.	Short description
`TMDX_DY_Initialize(order)`	subrout.	XI A	Initialization of module.
`TMDX_DY_ShowStatistic()`	subrout.	–	Print current statistic on the number of ca
`TMDX_DY_ResetCounters()`	subrout.	–	Reset intrinsic counters of the module.
`TMDX_DY_SetScaleVariation(c2)`	subrout.	–	Set new value for the scale-variation consta
`xSec_DY(X,proc,s,qT,Q,y,iC,CutP,Num)`	subrout.	XI C	Evaluates cross-section completely integra the bin.
`xSec_DY_List(X,proc,s,qT,Q,y,iC,CutP,Num,doP)`	subrout.	XI C	Evaluates cross-section completely integra the bin over the list.
`xSec_DY_List_APPROXIMATE(X,proc,s,qT,Q,y,iC,CutP,Num)`	subrout.	XI C	Evaluates cross-section completely integra the bin over the list (using several points).
`xSec_DY_List_BINLESS(X,proc,s,qT,Q,y,iC,CutP,Num)`	subrout.	XI C	Evaluates cross-section at the point over th

A. Initialization

Prior the usage module is to be initialized (once per run) by
`call TMDX_DY_Initialize(file)`
The order of used coefficient function is set in `constant-file` by `'LO'`(a_s^0), `'NLO'`(a_s^1),`'NNLO'` or `'N2LO'`(a_s^2), `'N3LO'`(a_s^3), `'N4LO'`(a_s^4). The value in brackets shows the maximum included perturbative order in the hard coefficient function.

B. The parameters of cross-section

Process: The process is encoded by integer numbers `(/p1,p2,p3,p4,p5,.../)` where

`p1` (integer) Defines the $prefactor2$ that contains the phase space elements and the universal part of factorization formula.

`p2` (integer) Defines the type of hadron 1.

`p3` (integer) Defines the type of hadron 2.

`p4` (integer) Defines the structure function F. See sec.X C.

Note: In artemide v.2.07 and earlier the process was encoded by 3 numbers, and did not contained the types of hadrons.

Kinematics: Each bin is defined by s (Mandelstam variable, GeV^2), Q (virtuality, GeV, [min,max]), y (rapidity, [min,max]) and q_T (transverse momentum, GeV, [min,max]). If the order of limits is inverted the result is 0.

Lepton cuts: The fiducial cuts on the lepton pair are defined by two variables: logical `iC` (If `inc=.true.` the evaluation of cuts will be done, otherwise it will be ignored.). And the array of four reals `(p1,p2,eta1,eta2)`, such that the cuts are defined as

$$|l_T| < \texttt{p1}, \qquad |l'_T| < \texttt{p2}, \qquad \texttt{eta1} < \eta, \eta' , \texttt{eta2}, \tag{11.6}$$

where l and l' are the momenta of produced leptons, with l_T's being their transverse components and η's being their rapidities. For accurate definition of the cut-function see sec.2.6 of [1] (particularly equations (2.40)-(2.42)).

C. Cross-section evaluation

The main subroutine is called `xSec_DY` and it have the following interface:

```
xSec_DY(X,process,s,qT,Q,y,includeCuts,CutParameters,Num)
```

where

- `X` is (real*8) value of cross-section.
- `process` (integer)array `(p1,p2,p3,p4)`.
- `s` is Mandelshtam variable s
- `qT` is (real*8)array `(qtmin,qtmax)`
- `Q` is (real*8)array `(Qmin,Qmax)`
- `y` is (real*8)array `(ymin,ymax)`
- `includeCuts` is logical
- `CutParameters` is (real*8) array `(k1,k2,eta1,eta2)` **OPTIONAL**
- `Num` is even integer that determine number of section in q_T integral **OPTIONAL**

Note, that optional variables could be omit during the call.

IMPORTANT: Practically, the call of this function coincides with `X` evaluated by the following set of commands

```
call TMDX_DY_SetProcess(process)
call TMDX_DY_XSetup(s,any,any)
call SetCuts(includeCuts,k1,k2,eta1,eta2)
call CalcXsec_DY_PTint_Qint_Yint (X,qtmin,qtmax,Qmin,Qmax,yMin,yMax,Num)
```

There is also an analogous subroutine that evaluate cross-section by list. It is called `xSec_DY_List` and it have the following interface:

```
xSec_DY_List(X,process,s,qT,Q,y,includeCuts,CutParameters,Num,doP)
```

All variables are analogues to those of `xSec_DY`, but should come in lists, i.e. `X` is (1:n), `process` is (1:n,1:4), `s` is (1:n), `qT`,`Q`,`y` are (1:n,1:2), `includeCuts` is (1:n), `CutParameters` is (1:n,1:4), `Num` is (1:n), `doP` is logic-single. **The arguments `Num` and `doP` are OPTIONAL arguments. The argument `CutParameters` must be presented**. This command compiled by OPENMP, so runs in parallel on multi-core computers. If `doP` is `.true.` than the list is split to runs of identical (except q_T) bins and integrated by a faster routine (default=False).
In addition there is a subroutine `xSec_DY_List_APPROXIMATE` with the same interface:

```
xSec_DY_List_APPROXIMATE(X,process,s,qT,Q,y,includeCuts,CutParameters,Num)
```

It does the same job as `xSec_DY_List`, however using simplified bin-integration routines (non-adaptive). It is several times faster, but has limited numerical precision.

Finally, there is an also analogous subroutine that evaluate cross-section by list without bin-integration. It is called `xSec_DY_List_BINLESS` and it have the following interface:

```
xSec_DY_List_BINLESS(X,process,s,qT,Q,y,includeCuts,CutParameters,Num)
```

All variables are analogues to those of `xSec_DY_List`, but should come in lists of central values, i.e. `X` is (1:n), `process` is (1:n,1:4), `s` is (1:n), `qT`,`Q`,`y` are (1:n), `includeCuts` is (1:n), `CutParameters` is (1:n,1:4), and `Num` is (1:n). **Only the `Num` argument is OPTIONAL arguments. The argument `CutParameters` must be presented**. This command compiled by OPENMP, so runs in parallel on multi-core computers.

Extra notes:

- The integrations over Q and y are adaptive Simpsons. We have found that it is the fastest (adaptive) method for typical cross-sections with tolerance $10^{-3} - 10^{-4}$. Naturally, it is not suitable for higher precision, which however is not typically required.
- The integration over p_T is not adaptive, since typically p_T-bins are rather smooth and integral is already accurate if approximated by 4-8 points (default minimal value is set in INI-file, and is automatically increased for larger bins). For unexceptionally large q_T-bins, or very rapid behavior we suggest to use overloaded versions with manual set of N (number of integral sections).
- For very small value of q_T the cross-section is frozen at $q_{T(abs\ min)}$, which is defined in INI-file (10^{-4} by default). It is done to avoid badly converging integrals.
- The evaluation can be done in parallel. The list commands are parallel by default. Basically, in this case, individual values for the list of cross-sections are evaluated in parallel. Unfortunately the parallel scaling-rate is not very high, on 8 processors it is about 400%-500% only. The parallelization is made with OPENMP library. To make the parallel computation possible, add `-fopenmp` option in the compilation instructions. The maximum number of allowed threads is set `constants-file`.

D. `LeptonCutsDY`

The calculation of cut prefactor is made in `LeptonCutsDY.f90`. It has two public procedures: `SetCutParameters`, and `CutFactor`.

- The subroutine `SetCutParameters(kT,eta1,eta2)` set a default version of cut parameters: $p_{1,2} < k_T$ and $\eta_1 < \eta < \eta_2$.
- The overloaded version of the subroutine `SetCutParameters(k1,k2,eta1,eta2)` set a default version of asymmetric cut parameters. $p_1 < k_1$,$p_2 < k_2$ and $\eta_1 < \eta < \eta_2$.
- Function `CutFactor(qT,Q_in,y_in, CutParameters)` calculates the cut prefactor at the point q_T, Q, y. The argument `CutParameters` is **optional**. If it is not present, cut parameters are taken from default values (which are set by `SetCutParameters`). `CutParameters` is array `(/ k1,k2,eta1,eta2/)`. The usage of global definition for `CutParameters` is not recommended, since it can result into running condition during parallel computation. This is interface is left fro compatibility with earlier version of `artemide`.

E. Variation of scale

TO BE WRITTEN

F. Partitioning of continuous p_T bins integration

There is a variant of the command for the speed-up of multiple computation of bin-integrations vs. q_T. It is based on the facts that (1) the q_T-shape is very smooth, (2) often one needs to evaluate many small q_T bins.

Given the sequence of bins in q_T $\{\{a_1, b_1\}, \{a_2, b_2\}, ..., \{a_k, b_k\}\}$ with $a_i < b_i$ (all other parameters for cross-section are the same), one can make the interpolation function in the complete range of$(\min(a), \max(b))$, and then integrate

the interpolation function. The tests with Chebyshev polynomial interpolation have shown that precision of such procedure is 5-6 digits for N=10 interpolation. The procedure is following:

1. Fix N (degree-1 of interpolation), and matrix M and nodes t,

$$M_{ij} = \frac{2\beta_i\beta_j}{N}\cos\left(\frac{ij\pi}{N}\right), \qquad t_i = \cos\left(\frac{i\pi}{N}\right), \tag{11.7}$$

   where $\beta_i = 1/2$ for $i = 1$ or N, and $\beta_i = 1$ for other i.

2. Compute a vector of cross-section at nodes

$$\boldsymbol{S} = \sigma\left(\frac{b-a}{2}t_i + \frac{b+a}{2}\right), \qquad a = \min(a), \qquad b = \max(b). \tag{11.8}$$

3. Then for each bin the integral over bin reads

$$\int_{a_i}^{b_i}\sigma dq_T = \frac{b-a}{2}\left(\boldsymbol{R}\left(\frac{2b_i-a-b}{b-a}\right) - \boldsymbol{R}\left(\frac{2a_i-a-b}{b-a}\right)\right)\cdot M\cdot\boldsymbol{S}, \tag{11.9}$$

   where

$$\boldsymbol{R}(x) = \{x, \frac{x^2}{2}, ..., \frac{1}{2}\left(\frac{T_{k+1}(x)}{k+1} - \frac{T_{k-1}(x)}{k-1}\right), ...\}. \tag{11.10}$$

   With T being the Chebyshev polynomial.

So, this algorithm requires only $N+1$ evaluations of cross-section.

However, this algorithms works well only in the vicinity of q_T-peak, so it is limited to the q_T-bins in the range $q_T < q_{T,MAX}$, where $q_{T,MAX}$ is specified in the constants file.

G. Options for evaluation of DY-like cross-section

1. π^2-resummation

The coefficient function of DY-like cross-section $|C_V|^2$ is evaluated at $(-q^2)$ (space-like momentum). Thus, it has contributions $\sim \pi^2$, which could be large, especially in the case of Higgs-boson production, see discussion in refs.Ahrens:2009cxz, Ahrens:2008qu. These corrections could be resummed by RG technique [9]. So,

$$|C_V(-q^2)|^2 \to U_\pi|C_V(q^2)|^2, \tag{11.11}$$

where U_π is the resummation exponent for πa_s-correction. At LO it reads [9]

$$U_\pi = \exp\left(\frac{\Gamma_0}{2a_s\beta_0^2}[2a\arctan a - \ln(1+a^2)]\right), \qquad a = \pi a_s. \tag{11.12}$$

To use the π-resummed coefficient function switch the corresponded option in `constants-file` (Option 9.A.p3).

2. Power corrections

There are many sources of power corrections. For a moment there is no systematic studies of power corrections for TMD factorization. Nonetheless I include some options in `artemide`, and plan to make more systematic treatment in the future.

Exact values of $x_{1,2}$ for DY: The TMD distributions enter the cross-section with x_1 and x_2 equal to

$$\begin{aligned} x_1 &= \frac{q^+}{p_1^+} = \frac{Q}{\sqrt{s}}e^y\sqrt{1+\frac{\boldsymbol{q}_T^2}{Q^2}} \simeq \frac{Q}{\sqrt{s}}e^y, \\ x_1 &= \frac{q^+}{p_1^+} = \frac{Q}{\sqrt{s}}e^{-y}\sqrt{1+\frac{\boldsymbol{q}_T^2}{Q^2}} \simeq \frac{Q}{\sqrt{s}}e^{-y}. \end{aligned} \tag{11.13}$$

Traditionally, the corrections $\sim q_T/Q$ are dropped, since they are power corrections. Nonetheless, they could be included since they have different sources in comparison to power corrections to the factorized cross-section. Usage of one or another version of $x_{1,2}$ is switched in `constants-file` (Option 9.A.p2).

Exact values for hard scale for DY: The TMD hard coefficient function contains $\ln(|2q^+q^-|/\mu^2)$ which is usually approximated by $\ln(Q^2)$. Similarly, the *zeta*-scales are normalized as $\zeta\bar{\zeta} = (2q^+q^-)^2$, which is usually approximated by Q^4. So, the exact values of scales are

$$\mu = \sqrt{Q^2 + \boldsymbol{q}_T^2}, \qquad \zeta_1 = \zeta_2 = Q^2 + \boldsymbol{q}_T^2. \tag{11.14}$$

Traditionally, the corrections $\sim q_T/Q$ are dropped, since they are power corrections. Nonetheless, they could be included since they have different sources in comparison to power corrections to the factorized cross-section. Usage of one or another version of *scales* is switched in `constants-file` (Option 9.A.p4).

H. Enumeration of processes (LP case)

List of enumeration for `p2`

p2	$prefactor2$	Short description
0	1	Test case
1	$\frac{4\pi}{9}\frac{\alpha_{\rm em}^2(Q)}{sQ^2}\lvert C_V^{DY}(c_2Q)\rvert^2 R \times \mathrm{cut}(q_T)$	DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$. $\mathrm{cut}(q_T)$ is the weight for the lepton tensor with fiducial cuts (see sec.2.6 in [1]). Corresponds to DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$. The result is in pb/GeV^2
2	$\frac{2\sqrt{Q^2+\boldsymbol{q}_T^2}}{\sqrt{s}}\cosh y\ \frac{4\pi}{9}\frac{\alpha_{\rm em}^2(Q)}{sQ^2}\lvert C_V^{DY}(c_2Q)\rvert^2 R \times \mathrm{cut}(q_T)$	DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$. $\mathrm{cut}(q_T)$ is the weight for the lepton tensor with fiducial cuts (see sec.2.6 in [1]). The result is in pb/GeV^2. **Attention!:** this process corresponds to computation with the $dX = dx_F dQ^2 dq_T^2$ where $x_F = \frac{2\sqrt{Q^2+\boldsymbol{q}_T^2}}{\sqrt{s}}\sinh y$ is Feynman x. In this case the integration y is preplaced by the integration over x_F (i.e. limits of integration are treated as for x_F).
3	$\frac{4\pi^2}{3}\frac{\alpha_{\rm em}(Q)}{s} Br_Z \lvert C_V^{DY}(c_2Q)\rvert^2 R \times \mathrm{cut}(q_T)$	DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$ for the Z-boson production in the narrow width approximation. $Br_Z = 0.03645$ is Z-boson branching ration to leptons. $\mathrm{cut}(q_T)$ is the weight for the lepton tensor with fiducial cuts (see sec.2.6 in [1]). The result is in pb/GeV^2
4	$\frac{4\pi^2}{3}\frac{\alpha_{\rm em}(Q)}{s} Br_W \lvert C_V^{DY}(c_2Q)\rvert^2 R \times \mathrm{cut}(q_T)$	DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$ for the W-boson production in the narrow width approximation. $Br_W = 0.1086$ is W-boson branching ration to leptons. $\mathrm{cut}(q_T)$ is the weight for the lepton tensor with fiducial cuts (see sec.2.6 in [1]). Use together with `p3`=13,...,18. In this case, Q is be M_Z. The result is in pb/GeV^2
5	$\frac{2}{\pi}\frac{\pi m_H^2 a_s^2(Q)}{36 s v^2} C_t^2(m_t, Q)\lvert C_g(Q)\rvert^2 \lvert A(x_t)\rvert^2$	Cross-section $\frac{d\sigma}{dy d(q_T^2)}$ for the exclusive Higgs-boson production. The result is in pb/GeV^2. For definition of functions see [10].

Here $R = 0.3893379 \cdot 10^9$ the transformation factor from GeV to pb.

I. Enumeration of processes (KPC case)

List of enumeration for `p2`

p2	$prefactor2$	Short description
0	1	Test case
1	$\frac{2\pi^2}{9}\frac{\alpha_{\rm em}^2(Q)}{s}\lvert C_V^{DY}(c_2Q)\rvert^2 R \times \mathrm{cut}(q_T)$	DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$. $\mathrm{cut}(q_T)$ is the weight for the lepton tensor with fiducial cuts (see sec.2.6 in [1]). Corresponds to DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$. The result is in pb/GeV^2
2	$\frac{2\sqrt{Q^2+\boldsymbol{q}_T^2}}{\sqrt{s}}\cosh y\ \frac{2\pi^2}{9}\frac{\alpha_{\rm em}^2(Q)}{s}\lvert C_V^{DY}(c_2Q)\rvert^2 R \times \mathrm{cut}(q_T)$	DY-cross-section $\frac{d\sigma}{dQ^2 dy d(q_T^2)}$. $\mathrm{cut}(q_T)$ is the weight for the lepton tensor with fiducial cuts (see sec.2.6 in [1]). The result is in pb/GeV^2. **Attention!:** this process corresponds to computation with the $dX = dx_F dQ^2 dq_T^2$ where $x_F = \frac{2\sqrt{Q^2+\boldsymbol{q}_T^2}}{\sqrt{s}}\sinh y$ is Feynman x. In this case the integration y is preplaced by the integration over x_F (i.e. limits of integration are treated as for x_F).
3	—	Absent
4	—	Absent
5	$\frac{\pi m_H^2 a_s^2(Q)}{36 s v^2} Q^2 C_t^2(m_t, Q)\lvert C_g(Q)\rvert^2 \lvert A(x_t)\rvert^2$	Cross-section $\frac{d\sigma}{dy d(q_T^2)}$ for the exclusive Higgs-boson production. The result is in pb/GeV^2. For definition of functions see [10]. The correct formula is to be defined, this is LP formula multiplied by large-Q translation factor.

Here $R = 0.3893379 \cdot 10^9$ the transformation factor from GeV to pb.

XII. TMDX_SIDIS MODULE

This module evaluates cross-sections with the SIDIS-like kinematics. I.e. it expects the following kinematic input, (Q, x, z) or (Q, y, z) or (x, y, z), that are equivalent.

The general structure of the cross-section is

$$\frac{d\sigma}{dX} = d\sigma(q_T) = \int [bin]\ prefactor2\ \times F, \tag{12.1}$$

where

$$dX = dx dQ^2 d\boldsymbol{p}_T^2. \tag{12.2}$$

For further specifications of parameters see sec.XII F.

List of available commands

Command	Type	Sec.	Short description
`TMDX_SIDIS_Initialize(file,prefix)`	subrout.	XI A	* Initialization of module.
`TMDX_SIDIS_ResetCounters()`	subrout.	–	* Reset intrinsic counters of module.
`TMDX_SIDIS_ShowStatistic()`	subrout.	–	* Print current statistic on the number of calls.
`TMDX_SIDIS_SetScaleVariation(c2)`	subrout.	–	* Set new values for the scale-variation constants.
`xSec_SIDIS(X,process,s,pT,z,x,Q)`	subrout.	XII A	Evaluates cross-section completely integrated over the bin.
`xSec_SIDIS_List(X,process,s,pT,z,x,Q)`	subrout.	XII A	Evaluates cross-section completely integrated over the bin over the list.

*)The structure of the module repeats the structure of `TMDX_DY` module, with the main change in the kinematic definition. Most part of routines has the same input and output with only replacement `_DY`→`_SIDIS`. We do not comment such commands. They marked by $*$ in the following table.

A. Cross-section evaluation

The computation of SIDIS cross-section is called by the following subroutine:

```
xSec_SIDIS(X,process,s,pT,z,x,Q,incC,cuts,masses)
```

where

- `X` is (real*8) value of cross-section.
- `process` is (integer)array `(p1,p2,p3,p4)` (described below).
- `s` is (real*8) Mandelshtam variable s
- `pT` is (real*8) array `(qtmin,qtmax)`
- `z` is (real*8) array `(zmin,zmax)`
- `x` is (real*8) array `(xmin,xmax)`
- `Q` is (real*8) array `(Qmin,Qmax)`
- `incC` is (logical) flag to include cuts.
- `cuts` is (real*8) array `(ymin,ymax,W2min,W2max)` that parameterizes kinematic cuts.
- `masses` is (real*8) array `(mt,mp)` which is $(M_{target}, m_{produced})$ in GeV OPTIONAL. If not specified, considered as zeros.

There is also analogous subroutine that evaluate cross-section by list. It is called `xSec_SIDIS_List` and it have the following interface:

```
xSec_SIDIS_List(X,process,s,pT,z,x,Q,incC,cuts,masses)
```

All variables are analogues to those of `xSec_SIDIS`, but should come in lists, i.e. `X` is (1:n), `process` is (1:n,1:4), `s` is (1:n), `pT`,`x`,`Q`,`z` are (1:n,1:2). This command compiled by OPENMP, so runs in parallel on multi-core computers.

In addition to these main subroutines the following sub-routines are defined

```
xSec_SIDIS_BINLESS(proc,s,pt,z,x,Q,m1,m2)
```

Compute cross-section without integration over bin (and also ignoring any possible bin-cut options). All kinematic variables are (real*8).

There harpy-interface prevents using optional variables. Thus there are separate copies of the function with argument `masses` being obligatory:

```
xSec_SIDIS_List_forharpy(xx,process,s,pT,z,x,Q,doCut,Cuts,masses)
  xSec_SIDIS_BINLESS_List_forharpy(xx,process,s,pT,z,x,Q,masses)
```

B. Enumeration of processes

The enumeration of processes for SIDIS is analogous to the DY case. Namely, each process is defined by 4 integer numbers

```
[p1, p2, p3, p4]
```

with

- p_1 indicates type of $prefactor2$ to use. Depending on this prefactor also the integral over phase space may differ. See the table below for details.

- p_2 is the number of the target hadron (i.e. the hadron associated with TMDPDF)
- p_3 is the number of the produced hadron (i.e. the hadron associated with TMDFF)
- p_4 indicates the process (structure function). See the table below for details.

List of enumeration for $prefactor2$

p1

p1	$prefactor2$	Short description
0	1	For tests and cross-checks.
1	R_C	The cross-section $\frac{d\sigma}{dxdQ^2dzd(p_\perp^2)}$. The factor R_C is defined below.
2	$\frac{Q^2}{y}R_C$	The cross-section $\frac{d\sigma}{dxdydzd(p_\perp^2)}$, in this case integration over Q^2 is replaced by integration over y.
3	$\frac{x}{y}R_C$	The cross-section $\frac{d\sigma}{dydQ^2dzd(p_\perp^2)}$, in this case integration over x is replaced by integration over y.

The factor R_C is standard normalization of the cross-section (however without factor $1+\gamma^2/2x$, which is obsolete)

$$R_C = \frac{2\pi\alpha_{\rm em}^2(Q)}{Q^4\sqrt{1-\gamma^2\boldsymbol{\rho}_\perp^2}}\frac{y^2}{1-\varepsilon}|C_V^{SIDIS}(c_2Q)|^2c_0^{(un)}R_t \quad [\text{pb/GeV}^2], \tag{12.3}$$

with $R_t = 0.3893379\times10^9$ the transformation factor from GeV to pb, and

$$c_0^{(un)} = 1+\frac{\boldsymbol{p}_\perp^2}{(zQ)^2}\frac{\varepsilon-\frac{\gamma^2}{2}}{1-\gamma^2\rho^2} = 1+\frac{\boldsymbol{q}_T^2}{Q^2}\frac{\varepsilon-\frac{\gamma^2}{2}}{1+\gamma^2}.$$

Note, that different parts of this formula can be switched on/off by parameters of the power-corrections.

C. Kinematic cuts

Some experiments put extra constraints on the measurement. Typically, such constraint has the form

$$y_{\min} < y < y_{\max}, \qquad W^2_{\min} < W^2 < W^2_{\max}. \tag{12.4}$$

In this case the integration over x and Q^2 has additional restrictions. Say, if one integrates over a bin: $x_{\min} < x < x_{\max}$ and $Q_{\min} < Q < Q_{\max}$ the boundaries of the bin should be modified as (here integration over x is before the integration over Q)

$$\hat{x}_{\min}(Q) < x < \hat{x}_{\max}(Q), \qquad \hat{Q}^2_{\min} < Q^2 < \hat{Q}^2_{\max}. \tag{12.5}$$

where

$$\begin{aligned}
\hat{x}_{\min}(Q) &= \max\{x_{\min}, \frac{Q^2}{y_{\max}(s-M^2)}, \frac{Q^2}{Q^2+W^2_{\max}-M^2}\},\\
\hat{x}_{\max}(Q) &= \min\{x_{\max}, \frac{Q^2}{y_{\min}(s-M^2)}, \frac{Q^2}{Q^2+W^2_{\min}-M^2}\},\\
\hat{Q}^2_{\min} &= \max\{Q^2_{\min}, x_{\min}y_{\min}(s-M^2), \frac{x_{\min}}{1-x_{\min}}(W^2_{\min}-M^2)\},\\
\hat{Q}^2_{\max} &= \min\{Q^2_{\max}, x_{\max}y_{\max}(s-M^2), \frac{x_{\max}}{1-x_{\max}}(W^2_{\max}-M^2)\}.
\end{aligned}$$

If some cut is not required, then set corresponding parameter to limiting value, such as $y_{\min}=0$, $y_{\max}=1$, $W^2_{\min}=0$, $W^2_{\max}\gg Q^2$.

Important: If due to cuts the minimal and maximum values of parameter are inverted, the resulting cross-section is immediately set to zero.

D. Power corrections

There are many sources of power corrections. For a moment there is no systematic studies of of effect of power corrections for TMD factorization. Nonetheless we include some options in `artemide`, and plan to make systematic treatment in the future.

Kinematic corrections to the variables/phase space. There are three sources of kinematic corrections to the variables

$$\frac{p_\perp}{Q}, \qquad \frac{M}{Q}, \qquad \frac{m}{Q},$$

where M is the target mass, m is the mass of the fragmented hadron. These terms appear in many places of the SIDIS expression (see sec.XII F, for some minimal details), even without accounting for power-suppressed terms in the factorization formula. The nice feature of these correction is that they could be accounted exactly. The majority of these terms appears during of the Lorentz transformation of factorization frame (where the factorization is performed) to the laboratory frame (where the measurement is made).

Exact values for hard scale for SIDIS: The TMD hard coefficient function contains $\ln(|2q^+q^-|/\mu^2)$ which is usually approximated by $\ln(Q^2)$. Similarly, the *zeta*-scales are normalized as $\zeta\bar{\zeta} = (2q^+q^-)^2$, which is usually approximated by Q^4. So, the exact values of scales are

$$\mu = \sqrt{Q^2 - \boldsymbol{q}_T^2}, \qquad \zeta_1 = \zeta_2 = Q^2 - \boldsymbol{q}_T^2. \tag{12.6}$$

Traditionally, the corrections $\sim q_T/Q$ are dropped, since they are power corrections. Nonetheless, they could be included since they have different sources in comparison to power corrections to the factorized cross-section. Usage of one or another version of *scales* is switched in `constants-file` (Option 10.A.p6).

The accounting of these corrections is switched on/off in `constants`.

E. Bin-integration routines

The inclusion of bin-integration is essential for comparison with experiment. The integration over a bin is automatically included in the calculation of cross-section (although routines for unintegrated calculation are also presented). Typically, one does not need an extreme precision from such integration i.e. 4-5 digit precision is typically more then enough to reach the experimental precision. Also the cross-section is relatively slow-function (exception is Z-boson peak). Therefore, as a based method for bin-integration I have selected the adaptive Simpson integration rule. By default, the adaptive Simpson rule is used for integration over z, x, Q^2, and fixed-number-of-points Simpson method for p_T-integration. Note, one can switch to G7 method (faster but not presice) by modification of the pre-compile parameter in the beginning of the code.

The integration is made for phase-space element $dQ^2dxd\boldsymbol{p}_\perp^2$. I.e.

$$\Delta\sigma = \int dz \int dQ^2dxd\boldsymbol{p}_\perp^2 \; d\sigma = \int_{z_{\min}}^{z_{\max}} dz \int_{Q_{\min}}^{Q_{\max}} 2Q\; dQ \int_{x_{\min}}^{x_{\max}} dx \int_{p_{\min}}^{p_{\max}} 2|\boldsymbol{p}_\perp|d|\boldsymbol{p}_\perp|\; d\sigma. \tag{12.7}$$

For processes with $dx \to dy$, or $dQ^2 \to dy$ the integration is done in the same manner but with limits of integration over Q or x re-computed at values of y.

Notes:

- If the limits of integration are inverted (i.e. $\int_a^b$ with $a > b$) the integral is set to zero. It is made intentionally, because the cuts may shift the limits upside-down, which corresponds to the completely-eliminated case.
- The central value is computed at the mean of the bin. I.e. for bin $[a, b]$ the central value is computed at $(a+b)/2$.

F. SIDIS theory

In many aspects the theory for SIDIS is more complicated then for Drell-Yan. Here I collect the main definition which were used in the artemide. For a more detailed description see ref.[11]. Note, that some parts of definition were derived by me, since I have not found them in the literature.

1. Kinematics

The process is

$$H(P) + l(l) \to l(l') + h(p_h) + X, \tag{12.8}$$

with

$$P^2 = M^2, l^2 = l'^2 = m_l^2 \simeq 0, \qquad p_h^2 = m^2. \tag{12.9}$$

The standard variables used for the description of SIDIS are

$$q = l - l' \qquad \Rightarrow \qquad Q^2 = -q^2, \qquad x = \frac{Q^2}{2(Pq)}, \qquad y = \frac{(Pq)}{(Pl)}, \qquad z = \frac{(Pp_h)}{(Pq)}. \tag{12.10}$$

There are is also a set of power variables used to define power-corrections

$$\gamma = \frac{2Mx}{Q}, \qquad \rho = \frac{m}{zQ}, \qquad \boldsymbol{\rho}_\perp^2 = \frac{m^2 + \boldsymbol{p}_\perp^2}{z^2 Q^2}. \tag{12.11}$$

The variables x, y and Q^2 are dependent: $xy(s - M^2) = Q^2$. The phase-volume $dxdQ^2$ can be expressed via $dxdy$ or $dydQ^2$:

$$dxdQ^2 = \frac{Q^2}{y} dxdy = \frac{x}{y} dydQ^2. \tag{12.12}$$

A phase space point is totally characterized by the following numbers:

$$\{s, x, Q^2, z, \boldsymbol{p}_T^2\}. \tag{12.13}$$

Alternatively, one can replace $x \leftrightarrow Q^2 \leftrightarrow y$.

2. Cross-section

The cross-section for SIDIS has the general form

$$\frac{d\sigma}{dxdQ^2 dz d\psi d\phi_h d\boldsymbol{p}_{h\perp}^2} = \frac{\alpha_{\rm em}^2(Q)}{Q^6} \frac{y^2}{8z} \frac{L_{\mu\nu} W^{\mu\nu}}{\sqrt{1 + \boldsymbol{\rho}_\perp^2 \gamma^2}}, \tag{12.14}$$

where $L_{\mu\nu}$ is the lepton tensor, and $W_{\mu\nu}$ is the hadron tensor. The hadron tensor contains many term accompanied by polarization angles.

The TMD factorization is performed in the factorization frame with respect to $\boldsymbol{q}_T^2 \ll Q^2$. The factorization variable $|\boldsymbol{q}_T|$ is related to the measured variable $\boldsymbol{p}_{h\perp}$ as

$$|\boldsymbol{q}_T| = \frac{|\boldsymbol{p}_{h\perp}|}{z} \sqrt{\frac{1 + \gamma^2}{1 - \gamma^2 \rho^2}}, \qquad \Leftrightarrow \qquad |\boldsymbol{p}_{h\perp}| = z|\boldsymbol{q}_T| \sqrt{\frac{1 - \gamma^2 \rho^2}{1 + \gamma^2}}. \tag{12.15}$$

In the factorization frame, the unpolarized part of the hadron tensor

$$W^{\mu\nu} = \frac{-z_1 g_T^{\mu\nu}}{\pi} \int |\boldsymbol{b}| d|\boldsymbol{b}| J_0(|\boldsymbol{b}||\boldsymbol{q}_T|) \sum_f e_f^2 |C_V(-Q^2, \mu^2)|^2 F_1^f(x_1, \boldsymbol{b}) D_1^f(z_1, \boldsymbol{b}) + \dots, \tag{12.16}$$

where dots denote, polarized terms, and power corrections. The LP variables x_1 and z_1 are

$$x_1 = -\frac{2x}{\gamma^2} \left(1 - \sqrt{1 + \gamma^2 \left(1 - \frac{\boldsymbol{q}_T^2}{Q^2} \right)} \right), \tag{12.17}$$

$$z_1 = -z \frac{1 - \sqrt{1 + \left(1 - \frac{\boldsymbol{q}_T^2}{Q^2} \right) \gamma^2}}{\gamma^2} \frac{1 + \sqrt{1 - \rho^2 \gamma^2}}{1 - \frac{\boldsymbol{q}_T^2}{Q^2}}. \tag{12.18}$$

Making the convolution with unpolarized leptonic tensor, we get the following expression for the cross-section

$$\frac{d\sigma}{dxdQ^2dzd\boldsymbol{p}_{h\perp}^2} = 2\pi\frac{\alpha_{\rm em}^2}{Q^4}\frac{1}{\sqrt{1+\boldsymbol{\rho}_\perp^2\gamma^2}}\frac{y^2}{1-\varepsilon}\frac{z_1}{z}\left\{1+\left(\varepsilon-\frac{\gamma^2}{2}\right)\frac{\boldsymbol{\rho}_\perp^2-\rho^2}{1-\gamma^2\rho^2}\right\} \tag{12.19}$$
$$\int\frac{|\boldsymbol{b}|d|\boldsymbol{b}|}{2}J_0\left(\frac{|\boldsymbol{b}||\boldsymbol{p}_{h\perp}|}{z}\sqrt{\frac{1+\gamma^2}{1-\gamma^2\rho^2}}\right)\sum_f e_f^2|C_V(-\overline{Q}^2,\mu^2)|^2F_1^f(x_1,\boldsymbol{b})D_1^f(z_1,\boldsymbol{b}),$$

where

$$\varepsilon=\frac{1-y-\frac{\gamma^2y^2}{4}}{1-y+\frac{y^2}{2}+\frac{y^2\gamma^2}{4}}. \tag{12.20}$$

The value $\overline{Q}^2=-2q^+q^-$, note that $\zeta\bar{\zeta}=\overline{Q}^4$.

Clearly, this LP expression contains a set of power corrections. It is a standard problem of TMD factorization, where the limit of LP is not very standardly defined. Thus, there are options to switch off/on these corrections.

- **Transverse momentum corrections**: (option p_1 in section 10.C) Include terms proportional to $\boldsymbol{\rho}_\perp^2$.
- **Target mass corrections**: (option p_2 in section 10.C) Include terms proportional to γ^2.
- **Product mass corrections**: (option p_3 in section 10.C) Include terms proportional to ρ^2.
- **Exact LP values of x_1 and z_1**: (option p_4 in section 10.C) Uses values of x_1 and z_1 (if option is False $x_1\to x$, $z_1\to z$)
- **Exact LP value for factorization scale**: (option p_5 in section 10.C) Uses exact LP value for $\overline{Q}^2=-2q^+q^-=Q^2-\boldsymbol{q}_T^2$ (if option is False $\overline{Q}^2=Q^2$)

Note, that corrections are included in the order from top to bottom. For example, terms $\sim\gamma^2\rho^2=0$ if p_2 =F and p_3 =T.

XIII. KPC IN TMD FACTORIZATION

The TMD factorization with KPCs is computed using different route. The primary problem is that one must compute these correction in the momentum space. It essentially complicates the computation, since the evolution is done in the position space.

The module `TMDFint_KPC_DY` realizes the computation of the following integral

$$\mathcal{C}_{KPC}[K,FF] = 4p_1^+p_2^+\int d\xi_1 d\xi_2 \int d^4k_1 d^4k_2 \delta^{(4)}(q-k_1-k_2)\delta(k_1^2)\delta(k_2^2) \tag{13.1}$$

$$\delta(k_1^+ - \xi_1 p_1^+)\delta(k_2^- - \xi_2 p_2^-) K \times FF, \tag{13.2}$$

where K is a kinematic kernel, that is resulted from the convolution of tensors, and FF is the product of TMD distributions.

This function has the following interface

```
KPC_DYconv(Q,qT,x1,x2,mu,proc1,proc2)
```

where `Q,qT,x1,x2,mu` are usual variables, and `proc1` is (int,int,int)-array with first item being the number of FF, the second and third items being number of hadrons; `proc2` is the enumeration of factors K.

A. Realization of convolution

The integral for convolution can be presented as

$$\mathcal{P}\mathcal{C}_{KPC}[K,FF] = \mathcal{P}\int_0^{\pi/2} d\alpha \int_0^{\pi} d\theta \frac{K\times FF}{2}, \tag{13.3}$$

where

$$\xi_1 = \frac{x_1}{2}\left(1+S+\sqrt{\Lambda}\right), \tag{13.4}$$

$$\xi_2 = \frac{x_2}{2}\left(1-S+\sqrt{\Lambda}\right), \tag{13.5}$$

$$\boldsymbol{k}_1^2 = \frac{\tau^2}{4}\left(1-\Lambda+2S+S^2\right), \tag{13.6}$$

$$\boldsymbol{k}_2^2 = \frac{\tau^2}{4}\left(1-\Lambda-2S+S^2\right), \tag{13.7}$$

with

$$S = \frac{(\boldsymbol{q}_T\boldsymbol{\Delta})}{\tau^2} = \sqrt{\sin\alpha\frac{\boldsymbol{q}_T^2}{\tau^2}}\cos\theta, \qquad \Lambda = \frac{\lambda(\tau^2,\boldsymbol{k}_1^2,\boldsymbol{k}_2^2)}{\tau^4} = \left(1-\frac{\boldsymbol{q}_T^2}{\tau^2}\right)(1-\sin\alpha). \tag{13.8}$$

The factor $\mathcal{P}$ is the fiducial-cut factor (multiplied within the `TMDX_DY` part).

B. Enumeration of factors processes

N	K	FF	$\mathcal{P}$	Short description
1	1	$e_q^2 f_1 f_1$	$\mathcal{P}_0$	Unpolarized DY
2	1 ($=K_1$)	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	$\mathcal{P}_0$	Z-boson (without interference with γ)
3	1 ($=K_1$)	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	$\mathcal{P}_0$	Z-boson main term $\sim\mathcal{P}_0$

20	$1-\frac{\tau^2\Lambda}{Q^2}$	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	$\mathcal{P}_0$	$f_1 f_1$ part of angular coefficient A_0. Also the part of Z-boson production $\sim \mathcal{P}_0$.
21	$\frac{\tau^2 S\sqrt{\Lambda}}{Q q_T}$	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	$\mathcal{P}_1$	$f_1 f_1$ part of angular coefficient A_1. Also the part of Z-boson production $\sim \mathcal{P}_1$.
22	$\frac{Q^4+\Lambda\tau^4+Q^2\tau^2(-1+2S^2-\Lambda)}{Q^2 q_T^2}$	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	$\mathcal{P}_2$	$f_1 f_1$ part of angular coefficient A_2. Also the part of Z-boson production $\sim \mathcal{P}_2$.
23	$-2S\frac{\tau}{q_T}$	$\Delta^{GG'} z_{-\ell}^{GG'} z_{-f}^{GG'} \{f_1 f_1\}_A$	$\mathcal{P}_3$	$f_1 f_1$ part of angular coefficient A_3. Also the part of Z-boson production $\sim \mathcal{P}_3$.
24	$-2\sqrt{\Lambda}\frac{\tau}{Q}$	$\Delta^{GG'} z_{-\ell}^{GG'} z_{-f}^{GG'} \{f_1 f_1\}_A$	$\mathcal{P}_4$	$f_1 f_1$ part of angular coefficient A_4. Also the part of Z-boson production $\sim \mathcal{P}_4$.
29	$\frac{3}{2}-\frac{\tau^2\Lambda}{2Q^2}$	$\Delta^{GG'} z_{+\ell}^{GG'} z_{+f}^{GG'} f_1 f_1$	1	$f_1 f_1$ part of proportional to 1 in the cross-section ($\Sigma_{UU}+\Sigma_0/2$). Used for the normalization of some asymmetries.
30	$-\frac{\tau^2}{4M^2Q^2}(Q^2(-1+S^2+\Lambda)+(1+S^2-\Lambda)\Lambda\tau^2)$	$\Delta^{GG'} z_{+\ell}^{GG'} r_{+f}^{GG'} h_1^\perp h_1^\perp$	$\mathcal{P}_0$	$h_1 h_1$ part of angular coefficient A_0. Also the part of Z-boson production $\sim \mathcal{P}_0$.
31	$\frac{\tau^2 S\sqrt{\Lambda}}{4M^2 Q q_T}(2Q^2+(-1+S^2-\Lambda)\tau^2)$	$\Delta^{GG'} z_{+\ell}^{GG'} r_{+f}^{GG'} h_1^\perp h_1^\perp$	$\mathcal{P}_1$	$h_1 h_1$ part of angular coefficient A_1. Also the part of Z-boson production $\sim \mathcal{P}_1$.
32	$\frac{\tau^2}{4M^2Q^2q_T^2}(Q^4(-1+S^2+\Lambda)+\tau^4\Lambda(1+S^2-\Lambda)$ $+Q^2\tau^2(2S^4+(\Lambda-1)^2-3S^2(1+\Lambda)))$	$\Delta^{GG'} z_{+\ell}^{GG'} r_{+f}^{GG'} h_1^\perp h_1^\perp$	$\mathcal{P}_2$	$h_1 h_1$ part of angular coefficient A_2. Also the part of Z-boson production $\sim \mathcal{P}_2$.
35	$-\frac{\tau^3\sqrt{\Lambda}}{4M^2Qq_T^2}(Q^2(1+S^2-\Lambda)+\tau^2(-1+S^2+\Lambda))$	$i\Delta^{GG'} z_{+\ell}^{GG'} r_{-f}^{GG'} \{h_1^\perp h_1^\perp\}_A$	$\mathcal{P}_5$	$h_1 h_1$ part of angular coefficient A_5. Also the part of Z-boson production $\sim \mathcal{P}_5$.
36	$-\frac{\tau^3 S}{4M^2Q^2q_T}(Q^2(-1+S^2-\Lambda)+2\Lambda\tau^2)$	$i\Delta^{GG'} z_{+\ell}^{GG'} r_{-f}^{GG'} \{h_1^\perp h_1^\perp\}_A (= F_{11})$	$\mathcal{P}_6$	$h_1 h_1$ part of angular coefficient A_6. Also the part of Z-boson production $\sim \mathcal{P}_6$.
101	1	$R_{\bar{f}f'}^{Cu} e_f e_{f'}$	$\mathcal{P}_0$	Unpolarized DY on Cu
102	1	$R_{\bar{f}f'}^{^2H} e_f e_{f'}$	$\mathcal{P}_0$	Unpolarized DY on ^2H ($Z=1$ and $A=2$)
103	1	$R_{\bar{f}f'}^{W} e_f e_{f'}$	$\mathcal{P}_0$	Unpolarized DY on W ($Z=74$ and $A=183$)

Appendix A: Equivalence between process-enumeration in ver.< 3

In the versions of artemide < 3, the enumeration of processes was done with 3 integers $[\tilde{a}, \tilde{b}, \tilde{c}]$, where $\tilde{a}$ and $\tilde{b}$ determined the prefactor, and $\tilde{c}$ determined the actual process. In fact, first number was not used (extremely rarely), thus it was eliminated in version > 3. Instead, the numeration of hadrons was added. In artemide > 3 the process is enumerated by four integers $[b, c, h_1, h_2]$ where b determines the prefactor, c the type of process and h_1 and h_2 the hadrons for scattering.

The correspondence of process numbering is the following

- $\tilde{a} = 1$, other options absent
- $\tilde{b} = b$
- $\tilde{c}$ is replaced by the h_1, h_2, c according to the following table

List of available commands

artemide < 3	artemide > 3	Process	Comment
$\tilde{c}$	$[h_1, h_2, c]$		
0	[-,-,0]		Does not use TMDs.
9999	[-,-,0]	Test cases	Do not use TMDs.
9998	[-,-,0]		Does not use TMDs.
1	[1,1,1]	$p + p \to \gamma$	
2	[1,-1,1]	$p + \bar{p} \to \gamma$	
3	[1,1,2]	$p + p \to Z$	
4	[1,-1,2]	$p + \bar{p} \to Z$	
5	[1,1,3]	$p + p \to Z + \gamma$	
6	[1,-1,3]	$p + \bar{p} \to Z + \gamma$	
7	[1,1,4]	$p + p \to W^+$	
8	[1,1,5]	$p + p \to W^-$	
9	[1,1,6]	$p + p \to W^\pm$	
10	[1,-1,4]	$p + \bar{p} \to W^+$	
11	[1,-1,5]	$p + \bar{p} \to W^-$	
12	[1,-1,6]	$p + \bar{p} \to W^\pm$	
13	[1,1,7]	$p + p \to W^+$	Same as 7 with $\Gamma_W = 0$

14	[1,1,8]	$p + p \to W^-$	Same as 8 with $\Gamma_W = 0$
15	[1,1,9]	$p + p \to W^\pm$	Same as 9 with $\Gamma_W = 0$
16	[1,-1,7]	$p + \bar{p} \to W^+$	Same as 9 with $\Gamma_W = 0$
17	[1,-1,8]	$p + \bar{p} \to W^-$	Same as 10 with $\Gamma_W = 0$
18	[1,-1,9]	$p + \bar{p} \to W^\pm$	Same as 11 with $\Gamma_W = 0$
20	[1,1,10]	$p + p \to H$	
21	[1,1,11]	$p + p \to H$	Only f_1
22	[1,1,12]	$p + p \to H$	Only $f_1^\perp$
101	[2,1,1]	$\pi^- + p \to \gamma$	
102	[2,-1,1]	$\pi^- + \bar{p} \to \gamma$	
103	[-2,1,1]	$\pi^+ + p \to \gamma$	
104	[-2,-1,1]	$\pi^+ + \bar{p} \to \gamma$	
1001	[1,1,101]	$p + Cu \to \gamma$	
1002	[1,1,102]	$p +^2 H \to \gamma$	
1003	[-1,1,103]	$\bar{p} + W \to \gamma$	
1004	[2,1,103]	$\pi^- + W \to \gamma$	
polarized DY			
10001	[1,1,10001]	$p^\uparrow + p \to \gamma$	
10005	[1,1,10003]	$p^\uparrow + p \to \gamma + Z$	
10007	[1,1,10004]	$p^\uparrow + p \to W^+$	
10008	[1,1,10005]	$p^\uparrow + p \to W^-$	
10101	[2,1,10011]	$\pi^+ + p^\uparrow \to \gamma$	
10102	[2,-1,10011]	$\pi^+ + \bar{p}^\uparrow \to \gamma$	
10103	[-2,1,10011]	$\pi^- + p^\uparrow \to \gamma$	
10104	[-2,-1,10011]	$\pi^- + \bar{p}^\uparrow \to \gamma$	

13200	[1,1,13200]	$p^{\uparrow} + p \to Z + \gamma$	
13201	[1,1,13201]	$p^{\uparrow} + p \to W^{+}$	
13202	[1,1,13202]	$p^{\uparrow} + p \to W^{-}$	
SIDIS			
200n	[1,n,2001]	$p + \gamma \to h_n$	
201n	[1,n,2002]	$d + \gamma \to h_n$	
202n	[1,-n,2001]	$p + \gamma \to \bar{h}_n$	
203n	[1,-n,2002]	$d + \gamma \to \bar{h}_n$	
204n	[1,n,2003]	$n + \gamma \to h_n$	
205n	[1,-n,2003]	$n + \gamma \to \bar{h}_n$	
2101	[1,a,2101]	$p + \gamma \to h_{1+2}$	$a > 0$
2102	[1,a,2102]	$p + \gamma \to h_{1+2+3}$	$a > 0$
2103	[1,a,2103]	$d + \gamma \to h_{1+2}$	$a > 0$
2104	[1,a,2104]	$d + \gamma \to h_{1+2+3}$	$a > 0$
2105	[1,a,2105]	$n + \gamma \to h_{1+2}$	$a > 0$
2106	[1,a,2106]	$n + \gamma \to h_{1+2+3}$	$a > 0$
2111	[1,-a,2101]	$p + \gamma \to \bar{h}_{1+2}$	$a > 0$
2112	[1,?,2102]	$p + \gamma \to \bar{h}_{1+2+3}$	$a > 0$
2113	[1,?,2103]	$d + \gamma \to \bar{h}_{1+2}$	$a > 0$
2114	[1,?,2104]	$d + \gamma \to \bar{h}_{1+2+3}$	$a > 0$
2115	[1,?,2105]	$n + \gamma \to \bar{h}_{1+2}$	$a > 0$
2116	[1,?,2106]	$n + \gamma \to \bar{h}_{1+2+3}$	$a > 0$

Appendix B: Version history

Ver.3.03
- **TMDX_DY** introduced option for evaluation of sequences of q_T-bins by a faster method.
- **CollinsTMDFF**: implemented (at all levels)
- **SiversTMDPDF_OPE** Added griding option for OPE part.
- **aTMDe_math**: added
- Many external codes are re-integrated in the form of classes. These are
  - **Fourier, Moment, Fourier_byOgata** (unified in the new module **aTMDe_Ogata**),
  - **Twist2_ChGrid** (reintroduced in the new module **aTMDe_optGrid**).
  - **LHA_PDF** (reintroduced as a class **LHA_PDF**) this also removes limitation for number of hadrons.

  Corresponding corrections in interfaces are made in all dependent modules.
- **IO_functions, IntegrationRoutines, InverseMatrix** are renamed to **aTMDe_IO, aTMDe_Integration, aTMDe_invMatrix** for naming unification purposes.
- **many modules:** code cleaning, unification of interfaces.

Ver.3.02
- **wgtTMDPDF_OPE** & **wglTMDPDF_OPE**: fixed incorrect reading of order for twist-3 coefficient function.
- **aTMDe_control** & **aTMDe_setup**: added a couple of rare exceptions in the loading procedure.
- **TMDF**: the enumeration of Sivers and worm-gear related structure functions is updated.
- Integration with **snowflake**: snowflake became a part of distribution of the artemide. It is not required for functioning of the artemide code (in general), but could be used. It has few files and is compiled alongside with artemide. Makefile and harpy are updated accordingly.

Ver.3.01
- general: Computation of WW convolution is split into separate code.
- **uTMDPDF**, **uTMDFF**, **wgtTMDPDF**, **wglTMDPDF**: large-x resummation for twist-2 parts.
- **wglTMDPDF**: implemented (at all levels)
- **eeTMDFF**: implemented (at all levels)
- **BoerMuldersTMDPDF**: implemented (at all levels)
- **TMDF**: Enumeration for SIDIS processes updated.
- **TMDF**: Added check for insufficiently low-hard scale Q, μ or ζ.
- **uTMDFF**: Added Fourier by Levin transformation, and grids in k_T.
- **TMDX_DY**+`KPC`: Enumeration of processes in KPC and LP equalized. Ratios with numbers 200-300 removed.
- **TMDX_DY**: Fixed double-accounting of Jacobian in dx_F-integration
- **TMDX_SIDIS**: Fixed double-accounting of Jacobian in dy-integrations
- **TMDX_SIDIS**: Refactoring and updating into v3 paradigm.
- **TMDR**: reintroduced the CSS-like evolution factor, as a "pre-compiler" option
- **QCDinput**: removed connected to LHAPDF library. All changed to self-readers. Hardcore restriction on the number of hadrons.
- **LHA_alpha** & **LHA_PDF**: implemented

Ver.3.00
- **DY_KPC**: implemented
- **MULTIPLE CHANGES. MASSIVE REORGANIZATION OF CODE**

Ver.2.06
- **harpy** Added functions for wgt-function and also for TMDR module
- **common:tw2Convolution** Fixed bug, with incorrect estimation of the integral reminder at $x_0 \to 1$
- **All modules:** the options N2LO and N3LO can be used instead of NNLO and NNNLO
- **uTMDFF:coeffFunc** 3-loop coefficient function added.
- **uTMDPDF:coeffFunc** 3-loop coefficient function added.

– **uTMDFF:coeffFunc** Corrected missprint in the NNLO coeff.function ($\sim \pi^4\delta(\bar{z})$ term)

– **all .TMD.** Singular list updated to include $(\ln^3 \bar{x}/(1-x))_+$ terms [appear at N^{3}LO]

– **TMD_SIDIS** 4-loop coefficient function added.

– **TMD_DY** Added the check for the maximum size of the Q-bin size. If bin is too large, it is divided.

– **TMD_DY** 4-loop coefficient function added. Fixed a misprint in 3-loops (¡0.01% numerically)

– **TMD_AD:AD_Integral**, implemented `RADEvolution` function, and accompanying expressions (roots, Lagrange coefficients).

– **TMDR**, Fixed mistake with counting of orders (order of $\Gamma_{\rm cusp}$ was higher by 1).

– **TMD_AD:AD_atMu**, minor optimizations.

– **TMD_AD:AD_secondary**, added 4-loop expressions for RAD, and zeta-lines.

– **TMD_AD:AD_primary**, added 4-loop expressions for RAD, and anomalous dimensions, 5-loop $\Gamma_{\rm cusp}$. The values are taken from [2202.04660, 2205.02249, 1812.11818]

– **TMDF**, Added processes: 13001-13059, 13101-13106, 13111-13116, 13200-13202 (for A_{LT})

– **TMDs**, **TMDs_inKT**, **harpy** Added **wgtTMDPDF** routines.

– **aTMDe_control** Added **wgtTMDPDF** routines.

– **aTMDe_setup** Fixed bug with absent list of hadrons for Sivers.

– **aTMDe_setup** Modifications due to **QCDinput** and **wgtTMDPDF** updates.

– **wgtTMDPDF** Implemented (Thanks to M.Horstmann).

– **EWinput**: Fixed bug with the S-quark parameter for the interference term in Z-boson production (Thanks to S. Leal-Gomez).

– **QCDinput**: Added a check for the changing of PDF replica. It is not updated if the PDF replica is same. Updated corresponding function ins TMD's modules, such that they do not recompute grid if PDF remains the same.

– **QCDinput**: Added individual function for changing PDF replica for each TMD module. Also added argument to specify hadron. Updated corresponding function ins TMD's modules.

– **QCDinput**: Helicity PDF input is added.

– **QCDinput**: Bug in the enumeration of hadrons fixed.

– **TMDX_SIDIS**: Added option for exact factorization scales.

– **TMDX_DY**: Added option for exact factorization scales.

Ver.2.05

– **TMDX_SIDIS**: Minor optimization.

– **TMDF** Added the global mass scale used for normalization of TMD structure functions.

– **TMDF** Processes unpolarized DY, 101-104, are changed to $h+p$ (was $p+h$).

– **TMDF** Added processes for Sivers 10001,10005,10007,10008,10101-10104,12001–12059,12105,12106,12115,12116,

– **TMDF** Added processes for unpolarized SIDIS 2105,2106,2115,2116,2041-2049.

– **harpy** Update to python3.

– **harpy** Added functions for values of uTMDFF, lpTMDPDF, SiversTMDPDF, and DNP, and for TMD in kT space.

– **harpy** Added functions `Set_NPparameters` for **lpTMDPDF** and **SiversTMDPDF**.

– **IO_functions** Added a check for errors in during the file-reading.

– **aTMDe_control** Added version check for replica files

– **aTMDe_control** Some code refactoring + **SiversTMDPDF** (fixed couple of bugs in lpTMD-part)

– **aTMDe_setup** Some code refactoring + **SiversTMDPDF**

– **TMDs** Some code refactoring + **SiversTMDPDF**

– **SiversTMDPDF** Implemented.

– ***TMD**** Added `NA`-option for the order-definition.

– ***TMD**** Grid-code is moved to a separate directory.

Ver.2.04

– ***TMD****&`*TMD**_model`: Deep refactoring of the code. In particular
  * `*TMD**_model` turned to separate modules.
  * `*TMD**_modelInterface` are removed. Their functionality is totally inside the model-module.
  * The code is distributed into separate text-files.
  * Few optimization twicks.

– **TMDR**&`TMD_AD`: Deep refactoring of the code. In particular
  * `TMDR_model` turned to a separate module.
  * The definitions of $\mathcal{D}$ and ζ are moved to `TMD_AD`-module
  * Removed evolution type-4, since it coincides with type-3 at $\zeta = \zeta_{NP}$.
  * Removed function `TMDR R toSL`, since it coincides with `TMDR Rzeta` at $\zeta = \zeta_{NP}$
  * The code is distributed into separate text-files.
  * Few optimization twicks.

– **aTMDe_control**: Fixed bug in the check of length for λ_{NP} (Thanks to M.Echevarria).

– **TMDs_inKT**: The integration routine is updated. Changed the convergence check and added test cases.

– **TMDX_SIDIS**: Added integration method `'I0'` for x and z integration.

– **TMDX_SIDIS**: Fixed bug in the evaluation of cross-section without bin integration and without cuts

Ver.2.03

– **TMDR**: Added `orderZETA`=-1, which is used in the LO case.

– **TMDR**: `zeMUresum` temporary removed.

– **QEDinput** Changed routine for evaluation of $\alpha_{\rm QED}$. Previous had incorrect determination of boundary value.

– **TMDF**: Added extra test for null-value integrals in OGATA quadrature.

– **TMDF**: OGATA quadrature is updated, with different values of h for qT-intervals

– **TMDF**: Added test processes N+9999, N+9998

– All: Update of variables definition. Several “magic numbers” removed. Fixed several possible precision leaks.

– **TMDR**: Functions for derived anomalous dimensions modified. No precalculated numbers, all precalculation is explicit and done inside TMD_AD.

– **TMDR** and **TMDF**: Fixed minor leaks of precision, due to rounding errors in real to double real conversions.

– **TMDR**→**TMD_AD**: definition of anomalous dimension migrated. Definitions redone in explicit way. Fixed bug in Γ_3.

– **TMD_AD**: Implemented. This module contains definition of anomalous dimensions used in TMDR.

– **TMD_numerics**: Implemented. This module contains definition of precision, and definition of various constants.

– **All**: Some refactoring of warning generation.

– **All TMD-modules and models**: Composite TMD option is added.

Ver.2.02

– **Common**: Added utility module `IO_functions`, that contains common function for IO-control. Added added colored output.

– **All TMD-modules**: The initialization array is added to the initialization file. Now, after the initialization each module known the initial NP array.

– **aTMDe_control**: The initialization procedure read initial values of NP arrays, and (after the initialization procedure) set them.

– **aTMDe_control**: Added commands `artemide_GetReplicaFromFile` and `artemide_NumOfReplicasInFile`. The module commands `setReplica` are not used anymore.

– **aTMDe_setup**: Added command `CheckConstantsFile`.

– **TMDF**: Added a check for input values of $x, z, ...$ Now, for $x > 1$ the cross-section is evaluated to zero.

- **TMDF**: Added processes 2011-2039;2101-2104;2111-2114
- **TMDX_DY**: Added additional checks for TMDF_ConvergenceIsLost trigger. Now, each cross-section-evaluation routine returns 10^9 if convergence is lost.
- **TMDX_SIDIS**: Added α_s^3-term for hard-matching coefficient. NNNLO defined.
- **TMDX_SIDIS**: Updated fiducial cut function. Now, it includes $W^2_{\rm max}$ and needs 4-arguments.
- **TMDX_SIDIS**: Added options for accounting of q_T-correction in x_1 and z_1 irrespectively kinematic q_T-corrections.
- **TMDX_SIDIS**: Added options for selection of integration method for X- and Z- bins (SA and S5).
- **TMDX_SIDIS**: Fixed an error is sign for produced mass-correction. Added missed factor (z_1/z).
- **TMDX_SIDIS**: Fixed rare bug in determination of cuts.
- **TMDX_SIDIS**: Fixed a bug in the initialization routine (wrong line numbering for const-file.)

Ver.2.01

- **All modules:** Added a check for current input-file version. + Small corrections in massages, and bug reports.
- **TMDX_DY**: Added coefficient functions for Higgs-boson production, and corresponding process 5.
- **TMDX_DY**: Added option for coefficient function with π^2-resummation [9].
- **TMDX_DY**: Added a check: if y$\not\in [-y_0, y_0]$ the $\sigma = 0$.
- **lpTMDPDF**: Implemented.
- **TMDR**: Added expressions for gluon evolution.
- **TMDR**: Fixed numeric error in N^{3}LO resummed expression for $\mathcal{D}$ and NNLO+ expression for resummed ζ_μ.
- **TMDR**&**TMDs**: added evolution type 4. (Exact solution).
- **aTMDe_control**: fixed a bug with change of NP-parameters for individual modules
- **QCDinput**: fixed a bug with resetting of the PDF-grid
- **TMDF**: Added processes 101-104,1004, 20-22.

Ver.2.00

- **TMDX_DY**: fixed a memory leak in adaptive integration. It could cause to possible source of Segmentation fault error.
- **TMDX_DY**: Added cut for very-small p_T.
- **uTMDPDF**&**uTMDFF**: Changed priority of grid-calculation during scale-variation.
- **TMD-models**&**Twist2Convolution**: Added b^* parameter. **Model files are not compatible with earlier versions.**
- **Twist2Grid**: The routine for large-b is changed. Now it is faster and more accurate.
- **aTMDe_control**&**aTMDe_setup**: Implemented.
- **ALL MODULES**: Total change of initialization routines, and interface subroutines.

Ver.1.41

- **TMDR**&**TMDs**&**TMDF**: fixed issues that arise with some older Fortran compilers (thanks to Wen-Chen Chang).
- **TMDX_SIDIS**: Totally rewritten: processes redefined, interface redefined, integration routines rewritten, masses correction, parallelization, etc.
- **TMDX_DY**: Added extra checks.
- **uTMDFF**: The factor z^2 added as an external, see sec.VIII B 5. It should improve the convergence of "common"-convolution at small z.
- **TMDF**: Added Ogata tables for $\tilde{h} = h0.05$. They are used for integrations at smaller q_T.
- **TMDF**: Fixed potential bug in the initialization order.
- **TMDF**& higher: Fixed misprint in the name of function `TMDF_F`.
- **TMDF**: Added processes 2002-2009.
- **TMDF**&**TMDX_DY**: Added processes [?,4,2013-2018]. W-boson in the narrow width approximation.

– **TMDs**: Fixed error with the passing to NO parameters in the case of multiple distributions.

Ver.1.4
– **constants**: The format of EW input is changed. **Not compatible with older constants-file.**
– **TMDF**: Added processes 7-12. Fixed a mistake in processes 1,2,2001 (thanks to L.Zoppi & D.Gutierrez-Reyes)
– **TMDs**, **uTMDPDF** & **QCDinput**: Added `_SetPDFreplica` routine.
– **HARPY**: Implemented.
– **TMDs and sub-modules**: Added function `SetReplica`.
– **TMDs**: Added interface to optimal TMDs
– **TMDs**: Added check for length of incoming λ_{NP}
– **TMDs**: Fixed bug with incorrect gluon TMDs in functions `_50`.
– **TMDs_inKT**: Implemented.
– **TMDX_DY**: Added `xSec_DY` subroutines.
– **TMDX_DY**: Encapsulated process, and cut-parameter variables.
– **TMDX_DY**: Defined `p1`=2, which corresponds to integration over x_F. Removed old functions for x_F integrations.
– **TMDX_DY**: Fixed a bug with variation of c_2 (introduced in ver.1.3).
– **TMDX_DY**: Fixed a (potential) bug with y-symmetric processes.
– **LeptonCutsDY** : Old version of function removed, cut-parameters encapsulated into single array variable.
– **LeptonCutsDY** : asymmetric cuts in p_T are introduced.
– **LeptonCutsDY** : New function `CutFactor4`, which is analogous to `CutFactor3` but with one integral integrated analytically. Thus, it is more accurate, and faster by 5-20%
– **LeptonCutsDY** : some rearrangement of variables that makes `CutFactor4` and `CutFactor3` faster by 20%.
– **uTMDPDF** & **uTMDFF** : F_{NP} is now function of (x, z, b, h, λ). For that reason **this version incompatible with earlier versions.**
– **uTMDPDF** & **uTMDFF**: The common block of the code is extracted into a separate files. It include calculation of Mellin convolution and Grid construction.

Ver.1.32
– **TMDX_DY**: Added the routine with lists of y-bins, in addition to the lists of pt-bins.
– **TMDX_DY**: The implementation of parallel computation over the list of cross-sections.
– **LeptonCutsDY**: The kinematic variables are encapsulated.
– **TMDX_DY**: The kinematic variables are encapsulated (by the cost of small reduction of performance).
– **uTMDR**: Changed behavior at extremely small-b. Now values of b freeze at $b = 10^{-6}$.
– **uTMDPDF** & **uTMDFF**: Changed behavior at extremely small-b. Now values of b freeze at $b = 10^{-6}$.
– **TMDR**: Added NNNLO evolution (only for quarks, Γ_3 is from 1808.08981). Not tested.
– **TMDs**: Functions `RuPDFuPDF` and `antiRuPDFuPDF` are added.

Ver.1.31
– **Global**: The module `TMDF` is split out from `TMDX_..` modules.
– **Global**: Constant tables are moved to the folder \tables.
– **TMDX_..**: Change the structure of process definition.
– **TMDF**: Fixed bug with throwing exception for failed check of convergence of Ogata quadrature.
– **TMDF**: Added possibility to vary the Ogata quadrature parameters.
– **TMDX_DY**: The structure of interface to integrated cross-section simplified.
– **TMDX_DY**: Added trigger for exact power-corrected values of $x_{1,2}$.
– **uTMDPDF** & **uTMDFF**: Fixed rare error for exceptional restoration of TMD distribution from grid, then f_{NP} evaluated to zero.

Ver.1.3
– **Global**: Complete change of interface. Interface update for all modules.

– **uTMDPDF**: Added hadron dependence. `FNP` is now flavour and hadron dependent.

– **uTMDPDF**: Renormalon correction is removed. As not used.

– **TMDR**: The grid (and pre-grid) option is removed. Since it was incompatible with new interface. Also the new evolution (type 3) is faster any previous (with grids).

Ver.1.2(unpub.)
– `TMDR`: Older version is changed to `uTMDR1`. New evolution routine implemented.

– `uTMDFF`: Implemented.

– `uTMDPDF`: Fixed bug in evaluation of gluon TMDs, within the evaluation of $(..)_+$ part.

– **Global**: Removed functions for the evaluation of only 3-flavours TMDs. As outdated and not used.

– **Global**: Number of non-pertrubative parameters is now read from 'constants'-file. Module `TMDs` initialize sub-modules with accordance to this set.

– **Global**: Module `TMDX` is renamed into `TMDX_DY`, also many functions in it renamed.

– `uTMDX_SIDIS`: Implemented.

– `TMDs`: As an temporary solution introduced a rigid cut for TMD($\mu < m_q$).

– `TMDX`: Update of Ogata quadrature, with more accurate estimation of convergence.

Ver.1.1 hotfix Bugs in `uTMDPDF` and TMDR related to the evaluation of gluon TMDs fixed (thanks to Valerio Bertone).

Ver.1.1
– **Global:** The physical, numerical and option constant are moved to the file `constants`, where they are read during the initialization stage.

– `MakeGridsForTMDR`: Update of integration procedures to adaptive. Default grids accordingly updated (no significant effect).

– `uTMDPDF`: Update of the integration procedure in `uTMDPDF`, to adaptive Gauss-Kronrod (G7-K15). with special treatment of the $x \to 1$ singularity.

– `uTMDPDF`: The procedure for evaluation of TMD for individual flavour (`uTMDPDF_lowScale(f,x,b,mu)`) is removed, as outdated.

– `uTMDPDF`: Removed argument μ, from `uTMDPDF_...(x,b)`. Added function `mu_OPE(b)`, which is used as μ-definition for TMDs.

– `uTMDPDF`: Optional griding of TMDs is added. See sec.**??**

– `TMDR`: fixed potential error in the "close-to-Landau-pole" exception.

– `TMDs`: fixed potential error in the evaluation of the gluon evolution factor.

– `TMDX`: the name convention of subroutines `CalculateXsection_...`, changed to `CalculateXsec_...`, to shorten the name length.

– `TMDX`: added functions `CalculateXsec_PTint_Qint_YintComplete(X,qtMin,qtMax,QMin,QMax)` and `CalculateXsec_Qint_YintComplete(X,qtMin,qtMax,QMin,QMax)`.

– `TMDX`&`TMDs`&`uTMDPDF`: the independent variation constant c_4 is added (in the ver.1 variation of c_3 and c_4 was simultaneous). The corresponding routines are updated.

Ver.1 Release: `uTMDPDF`, `TMDR`, `TMDs` and `TMDX` modules. Only Drell-Yan-like cross-sections.

Appendix C: Version of input-file

31. Initial version (corresponds to clean ver.3.00)
32. +`eeTMDFF`-section (section 17).
33. +parameters for large-x resummation for wgt-function (section 13.B.p4 - p5).
34. +`wglTMDPDF`-section (section 16).
35. +`hPDF`-section (section 1.F).
36. +parameters of grids for Sivers function-section (section 12.E).
37. +`CollinsTMDFF`-section (section 18).
38. +options regarding the q_T-integration in TMDX_DY section (section 9.B.p6-p8)

For input-file before version 3.00

1. Initial version (corresponds to clean ver.2.00)
2. +TMD evolution type 4.
3. +linearly polarized gluon TMDPDF (sections 11, and 1.D)
4. +mass of top-quark (sec.1.A.p3)
5. +parameters of Higgs boson (mass,width,vev) (sec.2.D.p1-2.D.p3)
6. +option for accounting of π^2 corrections in coef.function for DY-like processes (sec.9.A.p3)
7. +options for specification of z-bin integrations in SIDIS module (sec.10.B.p3 & 10.B.p4)
8. +options for specification of x-bin integrations in SIDIS module (sec.10.B.p5 & 10.B.p6)
9. +options accounting q_T-correction in x_1 and z_1 in SIDIS module (sec.10.A.p5)
10. +initialization arrays are added to TMDR, uTMDPDF, uTMDFF, lpTMDPDF (secs.3.B.p2; 4.B.p2; 5.B.p2; 11.B.p2)
11. +added primary section of parameters for aTMDe-control (0.C) and the check of trigger of initialization by NP arrays (secs.0.C.p1)
12. +composite TMD option for each TMD module (secs. 4.A.p2, 5.A.p2, 11.A.p2)
13. +masses of leptons e, μ, τ (secs. 2.E.p1, 2.E.p2, 2.E.p3)
14. +value of α_{QED} at mass of tau-leptons (secs. 2.A.p4)
15. +Sivers function (sections 12)
16. +global mass scale value used in TMDF (sections 7.p2)
17. +option for accounting of exact values of factorization scales in DY (sections 9.A.p4)
18. + helicity PDF in QCDinput (section 1.E)
19. + worm gear T function (section 13)
20. +option for accounting of exact values of factorization scales in SIDIS (sections 10.A.p6)
21. +maximum size of the Q-bin (sections 9.B.p3)

Appendix D: Theory

This section is under construction.

In this section, the details of theoretical input coded in `artemide` is given. This section does not pretend to be a comprehensive review of TMD factorization. Only superficial details and references to particular realizations are given. For detailed and accurate description of the theory see specialized literature.

1. Definition of TMD distributions

TO BE WRITTEN. In this section, there will be operator definition of TMD distributions, and details on way the perturbative matching and NP-medeling is realized in `artemide`.

$$F_f(x,b) = \int_x^1 \frac{dz}{z} C_{f\leftarrow f'}(z,b^*,c_4\mu_{\rm OPE}) f_{f'}(\frac{z}{x},c_4\mu_{\rm OPE}) f^f_{NP}(x,z,b,\{\lambda\}), \tag{D1}$$

where $f_f(x,\mu)$ is PDF of flavor f, C is the coefficient function in ζ-prescription, f_{NP} is the non-perturbative function. The variable c_4 is used to test the scale variation sensitivity of the TMD PDF. The NNLO coefficient functions used in the module were evaluated in [12] (please, cite it if use).

PLAN TO ADD

- Definition of TMD distributions, operators, Lorenz structures, coordinate and momentum space.
- Perturbative matching, and ζ-prescription
- NP-modeling

2. Evolution of TMD distributions

The detailed theory is given in the article [6]. The NLO rapidity anomalous dimension has been evaluated in [13]. The NNLO rapidity anomalous dimension has been evaluated in [14, 15].

The evolution of TMD distribution of any kind is given by the following pair of equations

$$\mu^2\frac{d}{d\mu^2}F_{f\leftarrow h}(x,b;\mu,\zeta) = \frac{\gamma_F^f(\mu,\zeta)}{2}F_{f\leftarrow h}(x,b;\mu,\zeta), \tag{D2}$$

$$\zeta\frac{d}{d\zeta}F_{f\leftarrow h}(x,b;\mu,\zeta) = -\mathcal{D}^f(\mu,b)F_{f\leftarrow h}(x,b;\mu,\zeta), \tag{D3}$$

where $F_{f\leftarrow h}$ is the TMD distribution (TMDPDF or TMDFF) of the parton f in hadron h. The function $\gamma_F(\mu,\zeta)$ is called the TMD anomalous dimension and contains both single and double logarithms. The function $\mathcal{D}(\mu,b)$ is called the rapidity anomalous dimension. TMD and rapidity anomalous dimensions have not unified notation in the literature, for comparison of notation see table I in ref.[6]. The only important quantum number for TMDs is the color representation the initiating parton, which is tied to the parton flavor, namely, quark (fundamental representation) or gluon (adjoint representation). However, as the TMD evolution does not mix the flavors and for simplicity of notation, we omit the flavor index f in this section, unless it is important.

The uniqueness of solution for the coupled system (D2)-(D3) is guaranteed by *the integrability condition*

$$\zeta\frac{d}{d\zeta}\gamma_F(\mu,\zeta) = -\mu\frac{d}{d\mu}\mathcal{D}(\mu,b), \tag{D4}$$

The mutual dependence can be worked out explicitlydue to the fact that the ultraviolet divergences of the TMD operator partially overlap with the rapidity divergences, (see e.g.[15, 16]),

$$\zeta\frac{d}{d\zeta}\gamma_F(\mu,\zeta) = -\Gamma(\mu), \tag{D5}$$

$$\mu\frac{d}{d\mu}\mathcal{D}(\mu,b) = \Gamma(\mu), \tag{D6}$$

where Γ is the (light-like) cusp anomalous dimension. The equation (D5) entirely fixes the logarithm dependence of the TMD anomalous dimension, which reads

$$\gamma_F(\mu,\zeta) = \Gamma(\mu)\ln\left(\frac{\mu^2}{\zeta}\right) - \gamma_V(\mu). \tag{D7}$$

The anomalous dimension γ_V refers to the finite part of the renormalization of the vector form factor. In contrast, the equation (D6) cannot fix the logarithmic part of $\mathcal{D}$ entirely, but only order by order in perturbation theory, because the parameter μ is also responsible for the running of the coupling constant.

The solution of eq. (D2)-(D3) can be written as

$$F(x,b;\mu_f,\zeta_f) = R[b;(\mu_f,\zeta_f)\to(\mu_i,\zeta_i)]F(x,b;\mu_i,\zeta_i), \tag{D8}$$

where R is the TMD evolution factor. The general form of the evolution factor is

$$R[b;(\mu_f,\zeta_f)\to(\mu_i,\zeta_i)] = \exp\left[\int_P\left(\gamma_F(\mu,\zeta)\frac{d\mu}{\mu} - \mathcal{D}(\mu,b)\frac{d\zeta}{\zeta}\right)\right], \tag{D9}$$

where (μ_f,ζ_f) and (μ_i,ζ_i) refer respectively to a final and initial set of scales. Here, the $\int_P$ denotes the line integral along the path P in the (μ,ζ)-plane from the point (μ_f,ζ_f) to the point (μ_i,ζ_i). The integration can be done on an arbitrary path P, and the solution is independent on it, thanks to the integrability condition eq. (D4).

The TMD evolution factor R obeys the transitivity relation

$$R[b;(\mu_1,\zeta_1)\to(\mu_2,\zeta_2)] = R[b;(\mu_1,\zeta_1)\to(\mu_3,\zeta_3)]R[b;(\mu_3,\zeta_3)\to(\mu_2,\zeta_2)], \tag{D10}$$

where (μ_3,ζ_3) is arbitrary point in (μ,ζ)-plane and the point inversion property

$$R[b;(\mu_1,\zeta_1)\to(\mu_2,\zeta_2)] = R^{-1}[b;(\mu_2,\zeta_2)\to(\mu_1,\zeta_1)]. \tag{D11}$$

These equations are the cornerstones of the evolution mechanism, since they allow an universal definition of the non-perturbative distributions and the comparison of different experiments.

Different realizations of TMD evolution (if they are compatible with the factorization theorem) can be casted into selected of different paths of integration ADD PICTURE. All choices are equivalent, up to next-to-given order perturbative corrections, treatment of NP parts, and implementation of matching for TMD distributions. In `artemide` sevaral paths are presented, however, the matching of TMD distribution to collinear distributions is done in ζ-prescription.

a. Equipotential lines & ζ-prescription

The ζ-prescription is based on consideration of equipotential lines for evolution equations (D2)-(D3) (or lines of null-evolution, for description of evolution potential and its properties see [6]). An equipotential line $\ell(b) = (\mu,\zeta)$ is determined by values of anomalous dimensions γ_F and $\mathcal{D}$ at given b. The equation for line $\ell(b)$ can be written in the form $\zeta = \zeta(\mu,b)$. In this parameterization the for function $\zeta(\mu,b)$ is

$$-\Gamma(\mu)\ln\left(\frac{\zeta(\mu,b)}{\mu^2}\right) - \gamma_V(\mu) = 2\mathcal{D}(\mu,b)\frac{d\ln\zeta(\mu,b)}{d\ln\mu^2}. \tag{D12}$$

Among set of equipotential lines there is a single special equipotential line, which passes thorough the saddle point of the evolution field. The saddle point $(\mu_{\text{saddle}},\zeta_{\text{saddle}})$ is defined by the equations

$$\mathcal{D}(\mu_{\text{saddle}},b) = 0, \qquad \gamma_V(\mu_{\text{saddle}},\zeta_{\text{saddle}}) = 0. \tag{D13}$$

Note, that the value of μ_{saddle} depends on value of b, and on NP model for $\mathcal{D}$. In general it could be found only numerically. In typical models μ_{saddle} is decreasing monotonous function of b. At certain large values of b it passes though the Landau pole, and its value could not be determined. This line we denote as

$$\textbf{special equipotential line} = \zeta_\mu(b), \qquad (\mu_{\text{saddle}},\zeta_{\text{saddle}}) \in \zeta_\mu(b). \tag{D14}$$

By definition a TMD distribution is the same for all points of equipotential line. In particular it means that there is no dependence on variable μ, since a variation of μ is compensated by appropriate variation in $\zeta(\mu)$.

Important note: The presence of NP part in $\mathcal{D}$ makes determination of special line involved. We distinguish three "realizations" of the special line. The exact which is determined by equation (D12) and boundary conditions (D14) (its values can be calculated by function `zetaSL`). The perturbative realization where NP part is dropped (its values can be calculated by functions `zetaMUperp` and `zetaMUresum`). And user-defined which is given by user in `TMDR_model.f90` file.

b. Exact solution for evolution to special line

There is a possibility to define the evolution to the optimal equipotential line in perturbation theory for any NP input, at any b. Instead of problematic variable b one uses the function $\mathcal{D}(\mu, b)$ as a variable. With the ansatz $\zeta_\mu = \mu^2 \exp(-g(\mu, b))$ we rewrite equation (D12) as

$$2\mathcal{D}(\mu, b)\left(1 - \frac{dg(\mu, b)}{d\ln\mu^2}\right) - \Gamma(\mu)g(\mu, b) + \gamma_V(\mu) = 0. \tag{D15}$$

or

$$2\mathcal{D}\left(1 - \frac{\partial g(\mu, \mathcal{D})}{\partial \ln\mu^2} - \frac{\Gamma(\mu)}{2}g'(\mu, \mathcal{D})\right) - \Gamma(\mu)g(\mu, \mathcal{D}) + \gamma_V(\mu) = 0, \tag{D16}$$

where $g' = dg/d\mathcal{D}$. The boundary condition (D13) is transformed to the condition that g is regular at $\mathcal{D} \to 0$. Note, that this condition does not depends on b and thus determines the function unambiguously at any b, even if saddle point is behind the Landau-pole values.

The equation (D16) is simplified further with introduction of new function $\tilde{g}(a_s, \mathcal{D}) = \mathcal{D}g(\mu, \mathcal{D})$:

$$2\mathcal{D} + 2\beta(a_s)\frac{\partial\tilde{g}(a_s, \mathcal{D})}{\partial a_s} - \Gamma(a_s)\tilde{g}'(a_s, \mathcal{D}) + \gamma_V(a_s) = 0, \tag{D17}$$

with boundary condition $\tilde{g}(a_s, 0) = 0$. The general solution reads

$$\tilde{g}(a_s, \mathcal{D}) = -\frac{\mathcal{D}}{2}\int^{a_s}\frac{da}{\beta(a)} - \int^{a_s} da\frac{\gamma_V(a)}{2\beta(a)} - \int^{a_s} da\frac{\Gamma(a)}{2\beta(a)}\int^{a}\frac{da'}{\beta(a')} + \Phi\left(\mathcal{D} + \int^{a_s} da\frac{\Gamma(a)}{2\beta(a)}\right), \tag{D18}$$

where Φ is a solution of the following transcendental equation

$$\Phi\left(\int^{a_s} da\frac{\Gamma(a)}{2\beta(a)}\right) = \int^{a_s} da\frac{\gamma_V(a)}{2\beta(a)} + \int^{a_s} da\frac{\Gamma(a)}{2\beta(a)}\int^{a}\frac{da'}{\beta(a')}. \tag{D19}$$

Unfortunately, this equation is practically impossible to solve for higher then NNLL anomalous dimensions. It is straightforward to show that at large-$\mathcal{D}$ the solution is

$$\tilde{g}(a_s, \mathcal{D} \to \infty) = \mathcal{D}\int^{a_s}\frac{-1}{\beta(a)}da + \dots . \tag{D20}$$

Note that asymptotic is defined up to a constant.

The function g is convenient to derive as perturbative expansion

$$g(\mu, \mathcal{D}) = \frac{1}{a_s(\mu)}\sum_{n=0}^{\infty} a_s^n(\mu)g_n(\mathcal{D}). \tag{D21}$$

We have found

$$g_0 = \frac{e^{-p} + p - 1}{\beta_0 p}, \tag{D22}$$

$$g_1 = g_0\left(\frac{\beta_1}{\beta_0} - \frac{\Gamma_1}{\Gamma_0}\right) + \frac{\gamma_1}{\gamma_0} - \frac{\beta_1}{2\beta_0^2}p, \tag{D23}$$

$$g_2 = g_0\frac{\beta_2\Gamma_0 - \beta_1\Gamma_1}{\beta_0\Gamma_0} + \frac{\mathrm{ch}p - 1}{p}\frac{\beta_0\Gamma_1^2 - \beta_0\Gamma_0\Gamma_2 + \beta_1\Gamma_0\Gamma_1 - \beta_2\Gamma_0}{\beta_0^2\Gamma_0^2} + \frac{e^p - 1}{p}\frac{\Gamma_0\gamma_2 - \Gamma_1\gamma_1}{\Gamma_0^2}, \tag{D24}$$

where $p = 2\beta_0\mathcal{D}/\Gamma_0$.

There is an additional problem that appears at NNNLO, namely, the coefficient g_3 is negative. It results to a singular solution for ζ_μ at large-$\mathcal{D}$. It indicates that the series has bad convergence properties. This problems occur for very large $\mathcal{D} > 0.9 - 1.1$. Currently to by-pass this problem the 4-loop exact ζ-line is not used (3-loop expression is used instead). For small-$\mathcal{D}$ the difference is negligible.

The solution for the evolution from $(\mu, \zeta) \to (\mu, \zeta_\mu)$ is

$$R[b; (\mu, \zeta) \to (\mu, \zeta_\mu)] = \exp\left(-\mathcal{D}(\mu)\ln\left(\frac{\zeta}{\mu^2}\right) - \mathcal{D}(\mu, b)g(\mu, \mathcal{D})\right). \tag{D25}$$

3. Expressions for evolution functions in `artemide`

Here are the expression for various evolution functions how they are written in `artemide` in `TMDR`-module.

a. Fixed order RAD:

`Dpert(mu,bT,f)`: returns $\mathcal{D}^f(b,\mu)$ [real(dp)]
`mu, bT` [real(dp)] scales in GeV, `f`[integer] flavour.
Order is controlled by `orderD` ($=n_{\max}$). At `orderD`=0, `Dpert`=0.

Solving (D6) we get

$$\mathcal{D}(b,\mu)=\sum_{n=1}^{\infty}a_s^n\sum_{k=0}^{n}\mathbf{L}_\mu^k d^{(n,k)}, \tag{D26}$$

where $d^{(n,0)}$ are obtained by calculation. The coefficients $d^{(n,k)}$ are

$$\begin{aligned}
d^{(1,1)} &= \frac{\Gamma_0}{2},\\
d^{(2,2)} &= \frac{\Gamma_0\beta_0}{4},\\
d^{(2,1)} &= \frac{\Gamma_1}{2},\\
d^{(3,3)} &= \frac{\Gamma_0\beta_0^2}{6},\\
d^{(3,2)} &= \frac{\Gamma_0\beta_1+2\Gamma_1\beta_0}{4},\\
d^{(3,1)} &= \frac{\Gamma_2+4\beta_0 d^{(2,0)}}{2},
\end{aligned}$$

where flavor on both sides is the same.

b. Resummed RAD:

`Dresum(mu,bT,f)`: returns $\mathcal{D}^f_{\text{resum}}(b,\mu)$ [real(dp)]
`mu, bT` [real(dp)] scales in GeV, `f`[integer] flavour.
Order is controlled by `orderDresum` ($=n_{\max}$). At `orderDresum`=0, `Dresum`=$-\frac{\Gamma_0}{2\beta_0}\ln(1-X)$.

One can resumm the logarithms in this expression and write it as

$$\mathcal{D}_{\text{resum}}=-\frac{\Gamma_0}{2\beta_0}\Big[\ln(1-X)+\sum_{n=1}^{\infty}\frac{a_s^n}{(1-X)^n}\sum_{k,l=0}^{n}X^k\ln^l(1-X)d_r^{(n,k,l)}\Big],\qquad X=\beta_0 a_s\mathbf{L}_\mu, \tag{D27}$$

the coefficients d_r are

$$\begin{aligned}
d_r^{(1,0,1)} &= B_1,\qquad d_r^{(1,1,0)}=B_1-G_1,\qquad d_r^{(1,0,0)}=0,\\
d_r^{(2,0,2)} &= -\frac{B_1^2}{2},\qquad d_r^{(2,0,1)}=B_1G_1,\\
d_r^{(2,2,0)} &= \frac{1}{2}\left(G_2-B_1G_1-B_2+B_1^2\right)\qquad d_r^{(2,1,0)}=B_1G_1-G_2,\\
d_r^{(2,0,0)} &= -2\frac{d^{(2,0)}\beta_0}{\Gamma_0},\qquad d_r^{(2,2,2)}=d_r^{(2,1,1)}=0,
\end{aligned}$$

$$
\begin{aligned}
d_r^{(3,0,3)} &= \frac{B_1^3}{3}, \qquad d_r^{(3,0,2)} = -\frac{B_1^3}{2} - B_1^2 G_1, \qquad d_r^{(3,0,1)} = B_1 G_2 + 4\frac{d^{(2,0)}\beta_1}{\Gamma_0}, \\
d_r^{(3,1,1)} &= B_1 B_2 - B_1^3, \qquad \text{other } d_r^{(3,k>0,l>0)} = 0, \\
d_r^{(3,3,0)} &= \frac{1}{3}\left(B_1^3 - 2B_1 B_2 + B_3 - B_1^2 G_1 + B_2 G_1 + B_1 G_2 - G_3\right), \\
d_r^{(3,2,0)} &= -\frac{1}{2}B_1^3 + B_1 B_2 - \frac{B_3}{2} + B_1^2 G_1 - B_2 G_1 - B_1 G_2 + G_3, \\
d_r^{(3,1,0)} &= B_1 G_2 - G_3, \qquad d_r^{(3,0,0)} = -2\frac{d^{(3,0)}\beta_0}{\Gamma_0}
\end{aligned}
$$

Here,

$$
G_i = \frac{\Gamma_i}{\Gamma_0}, \qquad B_i = \frac{\beta_i}{\beta_0}
$$

c. Fixed order ζ-line:

`zetaMUpert(mu,bT,f)`: returns $\zeta^f(b,\mu)$ [real(dp)]
`mu, bT` [real(dp)] scales in GeV, `f`[integer] flavour.
Order is controlled by `orderZETA` ($= n_{\max}$). At `orderZETA`¡0, `zetaMUpert`=1.

The solution of (D12) in PT can be written as

$$
\zeta^f(b,\mu) = \frac{C_0\mu}{b}e^{-v}, \qquad v = \sum_{n=0}^{\infty} a_s^n \sum_{k=0}^{n+1} \mathbf{L}_\mu v^{(n,k)}. \tag{D28}
$$

The *perturbative boundary condition* is that v is regular at $\mathbf{L} \to 0$. The first coefficients are

$$
\begin{aligned}
v^{(0,0)} &= g_1, \qquad v^{(0,1)} = 0, \\
v^{(1,2)} &= \frac{\beta_0}{12}, \qquad v^{(1,1)} = 0, \qquad v^{(1,0)} = \mathbf{d}^{(2,0)} - g_1 G_1 + g_2, \\
v^{(2,3)} &= \frac{\beta_0^2}{24}, \qquad v^{(2,2)} = \frac{\beta_0}{12}\left(B_1 + G_1\right), \qquad v^{(2,1)} = \frac{\beta_0}{2}\left(\frac{8}{3}\mathbf{d}^{(2,0)} - g_1 G_1 + g_2\right), \\
v^{(2,0)} &= \mathbf{d}^{(3,0)} - \mathbf{d}^{(2,0)} G_1 + g_1 G_1^2 - g_2 G_1 - g_1 G_2 + g_3,
\end{aligned}
$$

here

$$
\mathbf{d}^{(n,k)} = \frac{d^{(n,k)}}{\Gamma_0}, \qquad g_i = \frac{\gamma_V^{(i)}}{\Gamma_0}, \qquad G_i = \frac{\Gamma_i}{\Gamma_0}, \qquad B_i = \frac{\beta_i}{\beta_0}.
$$

d. Resummed ζ-line:

`zetaMUresum(mu,bT,f)`: returns $\zeta^f_{\rm resum}(b,\mu)$ [real(dp)]
`mu, bT` [real(dp)] scales in GeV, `f`[integer] flavour.
Order is controlled by `orderZETA`.
REMOVED in v.2.03. TO BE REINTRODUCED LATER.

e. Exact ζ-line:

`zetaSL(mu,bT,f)`: returns $\zeta^f_{\rm exact}(b,\mu)$ [real(dp)]
`mu, bT` [real(dp)] scales in GeV, `f`[integer] flavour.
Order is controlled by `orderZETA`. At `orderZETA`¡0, `zetaMUpert`=1.
Currently at the order =3, the result is NNLO. Due to instability of N^{3}LO at very large $\mathcal{D}$.

The solution for ζ can be written as

$$\zeta(b,\mu) = \mu^2 \exp\left(-g(a_s,\mathcal{D})\right). \tag{D29}$$

The function g is convinient to present as

$$g(a_s,\mathcal{D}) = \frac{\Omega(a_s,p)}{a_s\beta_0}, \qquad p = \frac{2\beta_0\mathcal{D}}{\Gamma_0}, \tag{D30}$$

where Ω is convenient to present in the following form

$$\Omega(a_s,p) = \sum_{n=0}^{\infty} a_s^n \Omega_n(p), \tag{D31}$$

with

$$\begin{aligned}
&\Omega_0 = z_1, \qquad \Omega_1 = (B_1-G_1)z_1 + g_1 - \frac{B_1}{2}p,\\
&\Omega_2 = \frac{B_2 - B_1G_1 + G_1^2 - G_2}{2} z_1 + \frac{-B_2 + B_1G_1 + G_1^2 - G_2 - 2G_1g_1 + 2g_2}{2} z_{-1} + g_2 - g_1G_1\\
&\Omega_3 = \frac{2B_3 + B_1^2G_1 - B_2G_1 - G_1^3 + 3G_1G_2 - B_1B_2 - B_1G_2 - 2G_3}{6} z_1\\
&+\frac{G_1-B_1}{2}\left(-B_2 + B_1G_1 + G_1^2 - G_2 - 2G_1g_1 + 2g_2\right) z_{-1}\\
&\left[\frac{-B_3 + B_1^2G_1 + 2B_2G_1 - 4G_1^3 + 6G_1G_2 - B_1B_2 + B_1G_2 - 2G_3}{12} + (2G_1 - B_1)\frac{G_1g_1 - g_2}{2} - \frac{G_2g_1 - g_3}{2}\right] z_{-2}\\
&+G_1^2g_1 - G_2g_1 - G_1g_2 + g_3.
\end{aligned}$$

Here

$$B_i = \frac{\beta_i}{\beta_0}, \qquad G_i = \frac{\Gamma_i}{\Gamma_0}, \qquad g_i = \gamma^{(i)}\frac{\beta_0}{\Gamma_0}, \qquad z_n = \frac{e^{-np} - 1 + np}{p}. \tag{D32}$$

Note that $\lim_{p\to 0} z_n \sim n^2 p/2$.

[1] I. Scimemi and A. Vladimirov, Eur. Phys. J. C **78**, 89 (2018), 1706.01473.
[2] I. Scimemi and A. Vladimirov, JHEP **06**, 137 (2020), 1912.06532.
[3] V. Moos, I. Scimemi, A. Vladimirov, and P. Zurita (2023), 2305.07473.
[4] A. Buckley, J. Ferrando, S. Lloyd, K. Nordström, B. Page, M. Rüfenacht, M. Schönherr, and G. Watt, Eur. Phys. J. C **75**, 132 (2015), 1412.7420.
[5] H. Ogata, Publications of The Research Institute for Mathematical Sciences **41**, 949 (2005), URL `https://api.semanticscholar.org/CorpusID:14429790`.
[6] I. Scimemi and A. Vladimirov, JHEP **08**, 003 (2018), 1803.11089.
[7] O. del Rio, A. Prokudin, I. Scimemi, and A. Vladimirov, Phys. Rev. D **110**, 016003 (2024), 2402.01836.
[8] T. Gehrmann, E. W. N. Glover, T. Huber, N. Ikizlerli, and C. Studerus, JHEP **06**, 094 (2010), 1004.3653.
[9] V. Ahrens, T. Becher, M. Neubert, and L. L. Yang, Phys. Rev. D **79**, 033013 (2009), 0808.3008.
[10] V. Ahrens, T. Becher, M. Neubert, and L. L. Yang, Eur. Phys. J. C **62**, 333 (2009), 0809.4283.
[11] A. Bacchetta, M. Diehl, K. Goeke, A. Metz, P. J. Mulders, and M. Schlegel, JHEP **02**, 093 (2007), hep-ph/0611265.
[12] M. G. Echevarria, I. Scimemi, and A. Vladimirov, JHEP **09**, 004 (2016), 1604.07869.

[13] M. G. Echevarria, I. Scimemi, and A. Vladimirov, Phys. Rev. D **93**, 054004 (2016), 1511.05590.
[14] A. A. Vladimirov, Phys. Rev. Lett. **118**, 062001 (2017), 1610.05791.
[15] A. Vladimirov, JHEP **04**, 045 (2018), 1707.07606.
[16] J. Collins, *Foundations of perturbative QCD*, vol. 32 (Cambridge University Press, 2013), ISBN 978-1-107-64525-7, 978-1-107-64525-7, 978-0-521-85533-4, 978-1-139-09782-6.