SnowFlake ver.2.00

Alexey A. Vladimirov
June 3, 2025

User manual for `SnowFlake` package, that performed evolution for twist-three PDFs.

Manual is updating.

The theory and the description of the algorithm are given in the publication. This manual contains only information about commands, structure and how to use the package.

I. GENERAL DESCRIPTION

The package contains three modules

- `HexGrid`: The module specifies the grid, and contains various auxiliary functions
- `EvolutionKernels`: The module contains routines for computation of evolution kernels, and implementation of Runge-Kuta
- `SnowFlake`: The module contains the user interface, and perform transformation between various inputs

The default way of operating is calling `SnowFlake` and its public routines. Also, it is simple to modify the grid specification, by changing corresponding variables in the top of `HexGrid.f90` file.

II. HEXGRID

The grid is a composition of 6 grids (one for each sector of the hexagon). Each sub-grid is 2D, and is parametrized by 2 integers:

$$n = 0, ..., N_R, \qquad k = 0, ..., N_\phi.$$

Here, N_ϕ is number of points on the segment, and N_R is the number of points long the radios i.e. the total number of points on perimeter is $N_p = 6N_\phi - 1$. For simplification of programming, the composition of 2D grid is transformed to 1D with $N = 0, ..., (6N_\phi)(N_R + 1)$.

Furthermore, the grid along r is stretched by function (default)

$$\rho(r) = 1 - \frac{\ln r}{\ln x_{\min}}, \qquad \Leftrightarrow \qquad r(\rho) = x_{\min}^{1-\rho},$$

where $x_{\min}$ is the minimal allowed r.

The public interface consists of the following functions

Function	output	Description

`RP_fromX12(x1,x2,r,phi)`	sub.	Transforms $x_{1,2}$ to (r,ϕ)
`X123_fromRP(r,phi,x1,x2,x3)`	sub.	Transforms (r,ϕ) to $x_{1,2,3}$
`get_X123_from_1Dindex(s,x1,x2,x3)`	sub.	Returns $x_{1,2,3}$ corresponding to the 1D index s.
`GETgrid(F)`	(real)(0:N)	Create (1D array)$F1$ from the real function $F(x1,x2)$
`GETinterpolation(x1,x2,grid)`	real	Find interpolation of grid into the point $x_{1,2}$
`GETinterpolatorB(n,x1,x2,c)`	real	Returns the interpolation function corresponding to the interpolation n at point $x_{1,2}$. Integer parameter c is to select option. c=0 normal, c=1 derivative with respect to ρ, c=2 derivative with respect to ϕ

III. EVOLUTIONKERNELS

The module compute and store the expression for kernels. This operation must be done in the begining of operation and consumes some time (depending on the setup). Also this module provide elementary evolution (by RG method) with a given kernel.

The public interface consists of the following functions

Function	output	Description
`EvolutionKernels_Initialize()`	sub.	Computes all elementary kernels, and store them as internal variables.
`EvNonSinglet(F,alpha,t0,t1)`	sub.	Evolve function F (1D array) from μ_0 to μ_1 with the kernel $\mathbb{H}_{NS}$. Input variables are $t_0=\ln(\mu_0^2)$ and $t_1=\ln(\mu_1^2)$. The coupling constant in alpha(t). The result is updated value of F
`EvSingletPLUS(F,Fg,alpha,t0,t1,nf)`	sub.	Evolve function F and F_g (1D arrays) from μ_0 to μ_1 with the singlet matrix $\mathbb{H}^+$. Input variables are $t_0=\ln(\mu_0^2)$ and $t_1=\ln(\mu_1^2)$. The coupling constant in alpha(t). Evolution is done at constant N_f =`nf` (int). The result is updated values of F and F_g.
`EvSingletMINUS(F,Fg,alpha,t0,t1,nf)`	sub.	Evolve function F and F_g (1D arrays) from μ_0 to μ_1 with the singlet matrix $\mathbb{H}^-$. Input variables are $t_0=\ln(\mu_0^2)$ and $t_1=\ln(\mu_1^2)$. The coupling constant in alpha(t). Evolution is done at constant N_f =`nf` (int). The result is updated values of F and F_g.
`EvSingletMINUS(F,Fg,alpha,t0,t1,nf)`	sub.	Evolve function F and F_g (1D arrays) from μ_0 to μ_1 with the singlet matrix $\mathbb{H}^-$. Input variables are $t_0=\ln(\mu_0^2)$ and $t_1=\ln(\mu_1^2)$. The coupling constant in alpha(t). Evolution is done at constant N_f =`nf` (int). The result is updated values of F and F_g.
`EvChirlaOdd(F,alpha,t0,t1)`	sub.	Evolve function F (1D array) from μ_0 to μ_1 with the kernel $\mathbb{H}_{CO}$. Input variables are $t_0=\ln(\mu_0^2)$ and $t_1=\ln(\mu_1^2)$. The coupling constant in alpha(t). The result is updated value of F
`SaveKernels(path)`	sub.	Saves the computed kernels into text-files that are possible to load later. The `path` points to the directory where kernels will be stored.
`ReadKernels(path)`	sub.	Read the computed kernels from text-files stored in `path`.

IV. SNOWFLAKE

This module take as input the boundary conditions and deshifrate them, then evolve this configuration to a give scale and store grids of distributions as internal variable. Upon request it provide an interpolation to a given point (Q,x_1,x_2), in a requested format.

The main procedure is

```
ComputeEvolution(mu0,mu1,alpha,G1,U1,D1,S1,C1,B1,G2,U2,D2,S2,C2,B2,inputQ,inputG)
```

where obligatory arguments are

- `mu0` is the value μ_0 at which the boundary condition is specified
- `mu1` is the value μ_1 the maximum value of scale (up to which the evolution is prepared)
- `alpha` is an external function of single real variable. $\alpha_s(\mu) = g^2/(4\pi)$.

For values $\mu_0 < \mu_c$ the c-quark is set to zero. For $\mu_0 < \mu_b$ the b-quark is set to zero. **This procedure prepare the grids of solution for evolution equation, from μ_0 to μ_1.**

All the rest arguments are optional and thus must be provided as with "`G1=...`" syntax. They are

- `G1,U1,D1,S1,C1,B1,G2,U2,D2,S2,C2,B2` external real-valued functions of (x1,x2) [real,real], corresponding to {gluon,u,d,s,c,b} flavors, and type 1,2 (depending on `inputQ,inputG`). If not provided, these functions replaced by zeros.
- `inputQ` can be `'T'`, `'S'`, `'C'`(default). For the cases

  `'T'` Input quark functions `Q1` and `Q2` are interpreted as functions $T(x_1, x_2, x_3)$ and $\Delta T(x_1, x_2, x_3)$

  `'S'` Input quark functions `Q1` and `Q2` are interpreted as functions $S^+(x_1, x_2, x_3)$ and $S^-(x_1, x_2, x_3)$

  `'C'` Input quark functions `Q1` and `Q2` are interpreted as functions $\mathfrak{S}^+(x_1, x_2, x_3)$ and $\mathfrak{S}^-(x_1, x_2, x_3)$
- `inputG` can be `'T'`, `'C'`(default). For the cases

  `'T'` Input quark functions `G1` and `G2` are interpreted as functions $T^+_{3F}(x_1, x_2, x_3)$ and $T^-_{3F}(x_1, x_2, x_3)$

  `'C'` Input quark functions `G1` and `G2` are interpreted as functions $\mathfrak{F}^+(x_1, x_2, x_3)$ and $\mathfrak{F}^-(x_1, x_2, x_3)$

The grid in Q is made using the variable $t = 2\ln(Q/\mu_0)$, with the step Δt defined in the INI-file. The interpolation is linear (**UPDATE! to polynomial**). Beyond the limits of Q the extrapolation is done.

After call of this subroutine the module stores the result in the internal format. It can be accessed by the function

```
GetPDF(x1,x2,Q,f,outputT)
```

where obligatory arguments are

- `x1, x2` the values of (x_1, x_2) at which the function is interpolated.
- `Q` the scale of distribution.
- `f` (int) specifies the flavor and the type of function. $f = 0, 1, 2, 3, 4, 5$ which corresponds to g, d, u, s, c, b (a la LHAPDF numeration).

The argument `outputT` is optional it can be

`'T'` Returns T^+_{3F} for $f = 0$ or $f = 10$, T^-_{3F} for $f = -10$, and T for $f = 1..5$ and ΔT for $f = -1..-5$

`'S'` Returns T^+_{3F} for $f = 0$ or $f = 10$, T^-_{3F} for $f = -10$, and S^+ for $f = 1..5$ and S^- for $f = -1..-5$

`'C'` (default) Returns $\mathfrak{F}^+$ for $f = 0$ or $f = 10$, $\mathfrak{F}^-$ for $f = -10$, and $\mathfrak{S}^+$ for $f = 1..5$ and $\mathfrak{S}^-$ for $f = -1..-5$

IMPORTANT:
The boundary conditions must satisfy the physical symmetry. Otherwise the result is not reliable.

Same procedure is used for the evolution of chiral-odd distributions. In this case use

```
ComputeEvolutionChiralOdd(mu0,mu1,alpha,U1,D1,S1,C1,B1)
```

and

```
GetPDFChiralOdd(x1,x2,Q,f)
```

There are no flags (since the input/output has unique form), no gluons and `f` can be only $1,2,3,4,5$. The type of the function (H or E) is defined by the symmetry of boundary condition.

The showflake also computes the functions $\bar g_2(x,Q)$ and $d_2(x,Q)$ which are defined as

$$\bar g_{2,f}(x,Q)=\int_x^1\frac{dy}{y}\left(\Delta q_T(y)+\Delta q_T(-y)\right),\qquad \Delta q(y)=\int[d\xi]\frac{\mathfrak{S}^+(\xi_{1,2,3})-\mathfrak{S}^-(\xi_{1,2,3})}{2}\frac{d}{d\xi_3}\frac{\delta(y+\xi_3)-\delta(y-\xi_1)}{\xi_1+\xi_3}. \tag{4.1}$$

and

$$d_{2,f}(Q)=\frac{1}{2}\int_{x_{\min}}^1 dx\;x^2\;\bar g_{2,f}(x,Q). \tag{4.2}$$

The corresponding functions are

```
G2(x,Q,f),    D2(Q,f).
```

In both cases there are also function

```
G2_List(x,Q,f),    D2_List(Q,f),
```

which are defined for the list of variables, and produce the list of result. There are computed in parallel, and give some gain in the evaluation velocity.

The integer f specifies the target and is defined via the following table

f	target	Description
1, 2, 3, 4, 5	d, u, s, c, b	
11	u-d	
12	u+d	
100	p	$\bar g_{2,p} = \frac{4}{9}\left(\bar g_{2,u}+\bar g_{2,c}(x,Q)\right)+\frac{1}{9}\left(\bar g_{2,d}+\bar g_{2,s}+\bar g_{2,b}\right)$ $d_{2,p}(x,Q) = \frac{4}{9}\left(d_{2,u}+d_{2,c}\right)+\frac{1}{9}\left(d_{2,d}+d_{2,s}(x,Q)+d_{2,b}\right).$
101	n	$\bar g_{2,n} = \frac{4}{9}\left(\bar g_{2,d}+\bar g_{2,c}\right)+\frac{1}{9}\left(\bar g_{2,u}+\bar g_{2,s}+\bar g_{2,b}\right)$ $d_{2,n} = \frac{4}{9}\left(d_{2,d}+d_{2,c}\right)+\frac{1}{9}\left(d_{2,u}+d_{2,s}+d_{2,b}\right).$
102	d=$\frac{p+n}{2}$	$\bar g_{2,d} = \frac{5}{18}\left(\bar g_{2,d}+\bar g_{2,c}+\bar g_{2,u}+\bar g_{2,s}+\bar g_{2,b}\right)$ $d_{2,d} = \frac{5}{18}\left(d_{2,d}+d_{2,c}+d_{2,u}+d_{2,s}+d_{2,b}\right).$

Another function defined in snowflake is g_{1T}. It is the small-b asymptotic of the worm-gear-T function (for that reason it is refereed as WGT). The definition is

$$\begin{aligned} g_{1T}(x) &= 2x\int[dy]\Bigg[\frac{\theta(y_1<-x;y_3>x)}{y_1y_3^2}S^+(y_{123}) \\ &\quad+\left(\frac{\theta(y_1<-x;y_3<x)}{y_2^2y_1}+\frac{\theta(y_3>x;y_1>-x)}{y_2^2y_3}\right)\left(S^+(y_{123})-S^+(-x,0,x)\right)\Bigg]. \end{aligned} \tag{4.3}$$

for $x>0$, and

$$\begin{aligned} g_{1T}(x) &= 2x\int[dy]\Bigg[\frac{\theta(y_1>-x;y_3<x)}{y_1y_3^2}S^+(y_{123}) \\ &\quad+\left(\frac{\theta(y_1>-x;y_3>x)}{y_2^2y_1}+\frac{\theta(y_3<x;y_1<-x)}{y_2^2y_3}\right)\left(S^+(y_{123})-S^+(-x,0,x)\right)\Bigg]. \end{aligned} \tag{4.4}$$

There two function related to this element

`WGT(x,Q,f),     WGT_fList(x,Q).`

The first is defined for $-1 < x < 1$ and $f = 1, .., 5$. The second is defined for $0 < x < 1$ and returns the list of WGT's as

$$\{-g_b(-x), -g_c(-x), -g_s(-x), -g_u(-x), -g_d(-x), 0, g_d(x), g_u(x), g_s(x), g_c(x), g_b(x)\}.$$