Date: 2001/09/25 14:26:50

Instructions for use of the disresum package

Mrinal Dasgupta
Gavin P. Salam

NB: there have been major changes to the package between the 1.0.x and 1.1.x series. Please look at the file `CHANGELOG` for a summary of these changes

This documentation applies to version 1.1.0.
Things that are new in 1.1.x are signalled by (➥ **v1.1.x**).

---

1 Introduction

The `disresum` package calculates resummed, matched and power-corrected distributions for DIS event shapes, based on calculations in [1, 2].

The dependence of DIS event shapes on structure functions makes the technical implementation of the resummation and matching considerably more complex than in e^+e^-, especially for the variables of [1]. In particular the package needs to be able to calculate the evolution of structure functions, the first and second order expansion of structure-function evolution and the convolution of the results with a variety of coefficient functions.

The package is written in Fortran 90. This allows the introduction of new operators such as a 'convolution' operator `.conv.`, such that in the program one can write lines like

```
dpdf_dlnQ2 = as2pi * (PLO .conv. pdf)
```

where `PLO` is an object of type `type(PMat)` containing a matrix of splitting functions and `pdf` is an array with the parton distributions (tabulated in x) for a given value of Q. The ability to structure the problem in this manner allows one to separate the various aspects of the problem (convolution,[1] definition of splitting functions, evolution in Q,...) and test them individually.

For the end-user of the package, the details of the language and internal data structures used (bearing some limited similarity to classes from a C++ point of view) are

[1] The convolution uses an arbitrary order (in the numerical sense) algorithm in x-space which bears many similarities to that outlined in [3, 4] though developed independently. The structure-function evolution part of the package has been tested against the MRS and CTEQ tabulated distributions. The evolution always agrees with at least one of MRS or CTEQ. Problems have been found in the MRS99 evolution for $x < 10^{-4}$ where at higher Q the MRS distributions are up to 10% too high. It has also been found that CTEQ5 uses the MSbar NLO splitting functions in the DIS scheme, rather than DIS-scheme splitting functions.

not particularly important and can be safely ignored: instead interfaces are provided suitable for use from fortran 77 or C programs.

The `disresum` package is not able to calculate exact fixed order distributions! These need to be supplied by external programs. Depending on the form in which these are supplied there are two interfaces to the package: the `withdsm` interface is to be used when the fixed-order distributions have been calculated with the `dis-shapes-mc` package. That package provides a common interface to DISENT and DISASTER++ and using just two runs gives distributions at an arbitrary number of x, Q points, allowing time savings of an order of magnitude compared to normal DISASTER++ runs. It is described in [5].

A second interface `userfo` is to be used if the user wishes to supply their own fixed-order calculations.

There are points common to both interfaces. These are outlined in the next section. The subsequent sections go into the details specific to each interface.

2 Common considerations

All references to a scale are in terms of objects with dimensions GeV, e.g. Q or μ, not Q^2 or μ^2.

The programs do not return differential distributions, but rather *integrated distributions*:

$$R(v) = \frac{1}{\sigma} \int^{v} dv' \frac{d\sigma}{dv} \tag{1}$$

where σ is the next-to-leading order cross section for events with $E > E_{\rm lim}$ in the current hemisphere.[2]

To describe subroutine interfaces we shall use fortran 90 notation, sepcifying the "`intent`" of each argument. For compactness we write "`real(dp)`" instead "`double precision`", though they are equivalent. Within the user's programs "`real(dp)`" will only be defined if user `use`s the `types` module.

2.1 Option parameters

Communication with the package about the event shape to be studied, matching options, etc. is done using integers. Integer constants have been defined to describe the different kinds of matching

Event-shapes are designated by an `EvshpId`. The complete list is given in table 1. Matching schemes are designated by a `MatchType`, with list of options given in table 2;

[2]Note that this differs from the convention in [1, 2], where for calculational purposes it is much simpler to normalise to the Born, $\mathcal{O}\left(\alpha_s^0\right)$, cross section. In `disresum` all internal calculations are carried out normalised to the Born cross section, and then renormalised to the NLO cross section just before being given to the user.

EvshpId_tau_zQ	τ_{zQ}
EvshpId_tau_zE	τ_{zE}
EvshpId_tau_tE	τ_{tE}
EvshpId_rho_E	ρ_E
EvshpId_Cpr_E	C_E
EvshpId_brd_zE	B_{zE}
EvshpId_brd_tE	B_{tE}

Table 1: Correspondence between EvshpId and event shape.

MatchType_none	no matching
MatchType_logR	usual log-R matching
MatchType_logRmod	the same, but modified to give correct upper limits
MatchType_M	Multiplicative matching (previously called basicR)
MatchType_Mmod	'modified' version
MatchType_M2	Like R-matching at $\mathcal{O}(\alpha_s)$ and M matching at $\mathcal{O}(\alpha_s^2)$
MatchType_M2mod	'modified' version
MatchType_NoResum	no resummation, just fixed order

Table 2: The different kinds of matching (➥ **v1.1.x**)

the different kinds of matching are documented in [2].

There exist various options also for the power corrections; these are given in table 3. The procedure for modifying $v_{\max}$ is described below.

Finally event-shape variables measured with respect to the z (i.e. photon) axis generally require structure-function evolution as part of the resummation. If one is for example fitting α_s one typically wants that structure function evolution to be carried out with the same value of α_s used elsewhere in the resummation and fixed-order calculation. Therefore one should not rely on the evolution of the parameterised distributions.[3] The different evolution options are outlined in table 4. LO evolution uses a LO α_s, while NLO evolution use a NLO α_s. Currently the PDF evolution (and the α_s evolution associated with it) is done with a fixed number of flavours (the number used in the fixed-order calculation) — this is consistent with the fact that the rest of the resummed calculation uses a fixed number of flavours. It should be noted that both LO and NLO evolutions are consistent with the use of NLO parton distributions because the difference between them is a NNL logarithm, and at order α_s^2 this is accounted for by the matching with the fixed-order distribution.

Some of the names given in the tables listed above may change in the future, especially those related to the matching.

[3]Strictly speaking of course one should use a whole new parton distribution set which has been fitted at that new α_s. But the error from using the wrong distribution at Q is in some sense less serious than the error from using the wrong α_s for the evolution between Q and τQ, since the distribution is directly sensitive to the latter.

`PowerType_none`	no power correction
`PowerType_shift`	simple shift of the distribution
`PowerType_modshift`	shift by an amount proportional to $(1 - v/v_{\max})^p$

Table 3: The different kinds of power correction (➥ **v1.1.x**)

`EvlnType_fromPRM`	Use the parton distribution parameterisation
`EvlnType_LO`	Evolve down from each Q using leading-order evolution
`EvlnType_NLO`	Evolve down from each Q using NLO evolution

Table 4: The different options for pdf evolution; relevant only for those variables defined with respect to the z axis.

The constants are all defined in `disresum_constants.h` for C++, and for fortran 77 in `disresum_constants.inc`. For fortran 90 they are defined in the `module` appropriate to the fixed-order method.

2.2 Parton distribution sets

Symbolic names have only been assigned to certain of the PDF sets. Whereas in other cases the number associated with the symbol had little importance, for pdf sets it matters because the number must be the same as that used in the `dis-shapes-mc` package.

The pdf interpolation routines included in this package have been slightly modified compared to the default ones, in that they expect to find the data files (containing the tabulated distributions) in directories `grv/`, `cteq/` and `mrs/` for the different parameterisations. It is up to the user to provide these sets. The interpolation order used for CTEQ has been increased from 2 to 4. Certain error messages in MRS may also have been changed.

2.3 Resummation parameters and properties (➥ v1.1.x)

In event shape resummations there are a variety of ambiguities related to (uncontrolled) higher-order terms, non-asymptotic terms and other such issues. Several options have been built into the program to enable to user to modify subleading behaviour so as to gauge its effect on predictions.

Quantities which are represented as a `double precision` number are referred to as *parameters*, while quantities represented as an `integer` are referred to as *properties*. Each quantity is identified by a *handle*, and may be set independently for each event shape through a call to

```
subroutine dg_UpdateDynamicParam(handle, evshp_id, val)
```

pdf identifier	value
pdf_CTEQ5	100
pdf_GRV	700
pdf_MRS99LO	1000
pdf_MRS99	1100
pdf_MRS99DIS	1200

Table 5: The different pdf sets available and the values associated with the identifier. The parameters indicate a base value for the set; to access a particular distribution it is necessary to add an integer, `iset` for CTEQ, `mode` for MRS, where the values of `iset` and `mode` are described in the PDF set documentation [6, 7, 8]. So for example CTEQ5M1 is given by `pdf_CTEQ5` $+ 8 = 108$.

factorisation scheme id	value
factscheme_MSbar	1
factscheme_DIS	2

Table 6: The different factorisation schemes. Note that the `scheme` produced by `dis-resum-mc` does not correspond to the numerical codes given here.

```
    integer,  intent(in)  :: handle, evshp_id
    real(dp), intent(in)  :: val
```

for parameters, and

```
  subroutine dg_UpdateDynamicProp(handle, evshp_id, val)
    integer,  intent(in)  :: handle, evshp_id
    integer,  intent(in)  :: val
```

for properties. Analogously the values can be retrieved with the following routines

```
  subroutine dg_GetDynamicParam(handle, evshp_id, val)
    integer,  intent(in)  :: handle, evshp_id
    real(dp), intent(out) :: val
```

and

```
  subroutine dg_GetDynamicProp(handle, evshp_id, val)
    integer,  intent(in)  :: handle, evshp_id
    integer,  intent(out) :: val
```

2.3.1 Parameters

The underlined name is the symbol for the (integer) handle that is to be used to set a given property.

`esMax`: the maximum value of the event shape; it is used in modified matching and power corrections, as the place where the integrated cross section should go to its maximum value, the distribution to zero (in some cases) and the power correction shift also to zero. It has a default value which is more or less suited to each event shape.

`esPTModPower`: in modified matching the logarithm $\ln 1/V$ is replaced by expressions such as

$$\frac{1}{p}\ln\left(\frac{1}{V^p} - \frac{1}{V_{\max}^p} - 1\right),$$

and in some places one encounters factors such as

$$\left(1 - \frac{V^p}{V_{\max}^p}\right).$$

The value of p is set by a call to `dg_UpdateDynamicParam` with the handle equal to `esPTModPower`. Its default value is 1.

`esNPModPower`: with the `modshift` option for the power correction, the shift to the distribution is multiplied by

$$\left(1 - \frac{V^p}{V_{\max}^p}\right).$$

where the value of p is set by a call to `dg_UpdateDynamicParam` with the handle equal to `esNPModPower`. Its default value is 1.

`esRescaleV`: the resummation can be defined in terms of logarithms of $1/V$, but equally well in terms of logarithms of a rescaled variable, $1/RV$. This parameter is the value of R, and defaults to 1.

2.3.2 Properties

`esC1MuF`: this property affects resummations involving a modification of the PDF scale (V^nQ^2) and determines whether the event-shape dependent coefficient function is to be convoluted with the PDF at the modified scale (in which case the property should be given the value `esC1MuF_VQ`),

$$C_1 \otimes q(x, V^n Q^2),$$

or at the hard scale of the process (`esC1MuF_Q`)

$$C_1 \otimes q(x, V^n Q^2).$$

The former is perhaps more in line with the usual philosophy used in e^+e^- resummations, but it is the latter which remains the default for the time being. We note that currently `esC1MuF_VQ` is not supported with the `EvlnType_fromPRM` option for the evolution of the parton distributions.

esSumType: this property affects the resummation for the B_{zE} broadening. If it is given the value `esSumType_standard` then a standard resummation is carried out with just the pure LL and NLL terms (this has a spurious divergence at very small B, but for practical purposes, e.g. fits of α_s this turns out to be fairly irrelevant).

There are two improved resummations, a minimal one, `esSumType_WithLbar_m` (closer to the standard resummation), and a non-minimal one `esSumType_WithLbar`. Note that the computation of these two options is much slower than for the standard resummation.

2.4 Running coupling

Limited access is also provided to an (approximate) two-loop evolution routine for α_s which implements the following formula

$$\alpha_s = \frac{1}{\beta_0 \ln Q^2/\Lambda^2}\left(1 - \frac{\beta_1}{\beta_0^2}\frac{\ln\ln Q^2/\Lambda^2}{\ln Q^2/\Lambda^2}\right)$$

and uses a variable number of flavours. One should first initialise it with the value of $\alpha_s(M_Z)$, using

```
subroutine dg_asInit(asmz)
  real(dp), intent(in) :: asmz
```

One can then access the value of α_s at a given scale through

```
real(dp) function dg_asValue(mu)
  real(dp), intent(in) :: mu
```

The α_s routines are not completely robust (and initialisation does not necessarily use the fastest method for determining Λ_{QCD} from $\alpha_s(M_Z)$) and if one tries to set $\alpha_s(M_Z)$ $\alpha_s(M_Z) > 0.152$, the program will fail with an obscure error code.

3 The withdsm interface

NOTE: certain facilities may not be fully supported; see section 5 for a description of what is and is not supported.

This interface is to be used if one has access to fixed-order distributions generated with the `dis-shapes-mc` package. A fortran 90 interface for the routines and constants of the previous section, as well as the routines described here can be obtained by placing

```
use driver_withdsm
```

in your fortran 77 routines (before any type declarations, including '`implicit none`') and compiling with fortran 90 rather than fortran 77. The routines can also be called without this explicit interface, but in that case one needs to have one of

```
include 'disresum_constants.inc'
#include "disresum_constants.h"
```

(depending on the language being used) so as to include the constants described in the previous section.

There are two steps. First one should read in the file containing all the fixed-order distributions and specify the event-shape variable of interest. This is achieved by a call to

```
      subroutine dwd_start(dsm_file, EvshpId,
     1                                FOE_handle_lo, FOE_handle_hi)
        character(len=*), intent(in)  :: dsm_file
        integer,          intent(in)  :: EvshpId
        integer,          intent(out) :: FOE_handle_lo, FOE_handle_hi
```

`dsm_file` is the name of the file produced by the `dis-shapes-mc` package and `EvshpId` is the integer parameter specifying the event shape to be studied. The variables `FOE_handle_lo` and `FOE_handle_hi` (Fixed Order Event-shape handles) are integers which indicate a set of data structures (one for each x, Q point, one for each event-shape) which contain all the information about matching and the fixed-order distribution. Any integer between these two values (inclusive) refers to a particular x, Q value and event shape. One can find out which by a call to

```
subroutine dwd_info(FOE_handle, xbj, ybj, Q, EvshpId)
  integer,  intent(in)  :: FOE_handle
  real(dp), intent(out) :: xbj, ybj, Q
  integer,  intent(out) :: EvshpId
```

which returns the information labelled by `intent(out)`. A second subroutine gives extra information:

```
subroutine dwd_extrainfo(FOE_handle, pdf_id, muF_Q, factscheme, nf)
  integer,  intent(in)  :: FOE_handle
  real(dp), intent(out) :: muF_Q
  integer,  intent(out) :: pdf_id, factscheme, nf
```

where `pdf_id` indicates the parton distribution that was used, according to the codes given in table 5. Though the factorisation scheme is implicitly specified by the PDF, this is also returned in `factscheme`, according to the codes in table 6. The ratio of the factorisation scale to Q is returned in `muF_Q` and the number of flavours is returned in `nf`.

To calculate a distribution one calls

```
      subroutine dwd_dist(FOE_handle, nbin, varvals, sigmavals,
     1                    alfas,  muR_Q, alfa0, muI,
```

```
     2                     MatchType, EvlnType, PowerType)
       integer,  intent(in)  :: FOE_handle
       integer,  intent(in)  :: nbin
       real(dp), intent(in)  :: varvals(*)
       real(dp), intent(out) :: sigmavals(*)
       real(dp), intent(in)  :: alfas, muR_Q, alfa0, muI
       integer,  intent(in)  :: MatchType, EvlnType, PowerType
```

This calculates the $R(v)$ at each of the `nbin` values of v given in `varvals` and puts the result in `sigmavals`. It is necessary to provide `alfas` at renormalisation scale μ_R, together with `muR_Q` $= \mu_R/Q$. The factorisation scale μ_F is predetermined by the fixed-order event shape distributions, as specified in the file produced by `dis-resum-mc`. The first moment of the infrared coupling $\alpha_0(\mu_I)$ is set by `alfa0` with the infrared matching scale μ_I given by `muI`. The various options concerning matching, structure function evolution and the power correction are set through the last three arguments.

4 The `userfo` interface

This has not yet been written: fine details could depend on the future development of matching schemes so it is simplest to avoid writing it until one knows exactly what info is needed for matching schemes.

5 Status

The program is not 100% complete, and in particular the B_{tE} variable has yet to be included. Below we discuss what is included, what needs to be taken into account from an efficiency point of view, and we list known bugs.

5.1 What does and doesn't work (➥ v1.1.x)

Corresponds to release 1.1.0.

	τ_{tE}, ρ_E, C_E	τ_{zQ}, τ_{zE}	B_{tE}	B_{zE}
`dis-shapes-mc`	✓	✓	✓	✓
Resummation	✓	✓	✗	✓
Matching	✓	✓	✗	✓
$\mu_R \neq Q$	✓	✓	—	✓(only std)
$\mu_F \neq Q$	✓	✓	—	✓(only std)
Cached evolution	—	sometimes	—	sometimes

Cached evolution is discussed below.

5.2 Speed considerations

In carrying out fits for α_s and α_0 the user should be aware of certain issues relating to the speed of the program.

There is an initial penalty associated with setting up objects containing splitting and coefficient functions. This is one-off. More important is the speed of the evaluation of distributions.

For event-shapes measured with respect to the thrust axis the most time-consuming part of the evaluation of the distribution is the calculation of the various functions such as Gamma-functions, logarithms and exponentials. Generally this ought not to be prohibitive.

For event-shapes measured with respect to the z axis there is an additional need to calculate the parton distributions at a range of scales which depends on the variable. For example for τ_{zE} the parton distributions have to be evaluated at a scale $\sqrt{\tau_{zE}}Q$. The program calculates a table of F_2 values as a function of Q and stores it in a cache, whose size is specified by the parameter `cache_size` in `resummation_manager.f90`. If a resummed distribution is recalculated with all parameters the same except α_0 then an evolution present in the cache can be reused. In other situations it may be necessary to recalculate an entry in the cache (and this may be fast or slow). The various cases are given in the table below.

Situation	`EvlnType_fromPRM`	`EvlnType_LO`	`EvlnType_NLO`
New x, Q or `pdf_id`	New entry, fast	New entry, slow	New entry, slow
New `pdf_id` at same x, Q	New entry, fast	New entry, slow	New entry, slow
New α_s or μ_R/Q	Reuse entry	Reuse entry	New entry, slow

Accordingly if one wants to fit for α_s and α_0, and speed is an issue, one is better off using `EvlnType_LO`.

(➥ **v1.1.x**)There are important speed considerations also in the case of B_{zE}. With the standard resummation all necessary integrals have been tabulated so as not to have to recalculate them for each point. With the improved resummations the integrals are generally a function of two or more parameters and so it is more straightforward to evaluate them on demand. This makes the program considerably slower.

5.3 Known bugs (➥ v1.1.x)

The routine for the running α_s (both internal and that offered to the end user) does not yield an exact solution to the renormalisation group equation; rather it implements an approximate formula given in the PDG handbook. This should not have too large an effect, but should be investigated a bit further.

References

[1] V. Antonelli, M. Dasgupta and G. P. Salam, JHEP **0002** (2000) 001 [hep-ph/9912488].

[2] M. Dasgupta and G. P. Salam, in preparation.

[3] P. G. Ratcliffe, hep-ph/0012376.

[4] C. Pascaud and F. Zomer, hep-ph/0104013.

[5] The documentation for the `dis-shapes-mc` program is not yet available.

[6] CTEQ5

[7] MRS99

[8] GRV98