Heavy Quark Structure Function Package

HQSTF version 2021-08-04

M. Botje[*]
Nikhef, Science Park 105, 1098XG Amsterdam, the Netherlands

October 3, 2021

Abstract

The HQSTF package is a QCDNUM add-on that computes heavy-quark contributions to the F_2 and F_L structure functions up to NLO from pdfs evolved in the fixed-flavour number scheme with n_f light quarks.

[*]email m.botje@nikhef.nl

Contents

1 Introduction

The HQSTF package is a QCDNUM add-on with routines that calculates up to NLO the heavy flavour contributions to the F_2 or F_L structure functions from pdfs evolved in the FFNS scheme with n_f light flavours.

In this computation a proton consists of $n_\mathrm{f} = (3, 4, 5)$ intrinsic light quarks while the heavy quark $(\mathrm{c}, \mathrm{b}, \mathrm{t})$ is produced in the hard scattering process. For the formalism underlying the computation of these structure functions and their scale variations we refer to [1] and to the QCDNUM write-up.

The HQSTF package is written in FORTRAN77 but interfaces are provided so that all FORTRAN routines can be called from a C++ program.

C++ The C++ wrappers reside in the namespace `QCDNUM` and the routine names are written in lower case, as is the QCDNUM convention. We refer to the QCDNUM manual for more on C++ interfaces.

Floating-point arguments are in double precision and input numbers must, in FORTRAN, be given in double precision format like `2.5D0` instead of `2.5`. In C++ the input format is free since the data-type is specified in the function prototype and the conversion is done automatically, if necessary.

Weight tables are stored in the HQSTF memory (an internal array). The memory size is specified by the parameter `nhqstor` in the file `hqstf.inc`; if you run out of space (error message) then you must set `nhqstor` as needed, and recompile HQSTF.

The HQSTF code is based on the QCDNUM toolbox and error messages are, in most cases, issued by the toolbox routines and not by HQSTF itself. However, the calling HQSTF routine is mentioned in the error message so that you know where it came from.

The call `ivers = IHQVERS()` gives you the current HQSTF version number.

2 Subroutine calls

In the descriptions below, output arguments are pre-fixed by an ampersand (`&`). The C++ wrappers have the same name (in lower case) and arguments as in FORTRAN:

```
call SUB(arguments)     →     QCDNUM::sub(arguments);
```

The C++ prototypes are listed in Appendix A.

```
call HQWORDS ( &ntotal, &nused )
```

`ntotal`	Number of words available in the HQSTF workspace (`nhqstor` in `hqstf.inc`).
`nused`	Number of words used (set to `0` before the call to `hqfillw` or `hqreadw`).

[1] E. Laenen et al., Nucl. Phys. **B392**, 162 (1993);
S. Riemersma et al., Phys. Lett. **B347**, 143 (1995).

```
call HQFILLW ( istf, qmass, aq, bq, &nused )
```

Fill the weight tables. To be called before anything else.

`istf`	Select structure function: 1 = F_L, 2 = F_2 and 3 = both.
`qmass(3)`	Input array with the c, b, t quark masses in GeV. If a quark mass is set to $m_h < 1$ GeV, no tables will be generated for that quark.
`aq, bq`	Defines the relation $Q^2 = a\mu_F^2 + b$.
`nused`	Gives, on exit, the number of words used in the workspace.

You will get a fatal error if the workspace is not large enough to hold all tables. In that case you can increase the value of `nhqstor` in the include file `hqstf.inc` and recompile HQSTF. The values of the mass and scale parameters can be retrieved by a call to `hqparms(qmass,aq,bq)`. The relation between μ_F^2 and Q^2 is defined by this routine. To convert between the scales use:

```
Q2   = hqqfrmu(qmu2)          qmu2 = hqmufrq(Q2)
```

The Q^2 scale can only be varied when the renormalisation and factorisation scales are set equal in QCDNUM. So you can vary the Q^2 scale or the μ_R^2 scale but not both.

```
call HQDUMPW ( lun, 'filename' )
```

Dump the weights in memory via logical unit number `lun` to a disk file. The dump is un-formatted so that the weight file cannot be exchanged across machines.

```
call HQREADW ( lun, 'filename', &nused, &ierr )
```

Read weights from a disk file via logical unit number `lun`. On exit, `nused` contains the number of words read into the workspace (fatal error if not enough space, see above) and the flag `ierr` is set as follows.

0	Weights are successfully read in.
1	Read error or input file does not exist.
2	Incompatible QCDNUM version.
3	Incompatible HQSTF version.
4	Incompatible x-μ^2 grid definition.

These errors will not generate a program abort so that one should check the value of `ierr`, and take the appropriate action if it is non-zero.

```
call HSWITCH ( iset )
```

By default, the structure functions are calculated from the QCDNUM pdf-set `iset` = 1. With this routine you can switch to another set, provided that it contains unpolarised pdfs. For imported pdf-sets you have to make sure of that yourself.

It is also possible to switch pdf-sets via the `istf` argument in the structure function routine `hqstfun`, see below.

```
call HQSTFUN ( istf, icbt, def, x, Q2, *f, n, ichk )
```

Calculate the heavy quark contribution to a structure function.

`istf`	Calculate F_L (1) or F_2 (2).
`icbt`	Select contribution from charm (1), bottom (2) or top (3).
`def(-6:6)`	Coefficients of the quark linear combination for which the structure function is to be calculated (see below).
`x, Q2`	Input arrays containing a list of x and Q^2 (not μ^2) values.
`f`	Output array containing the list of structure functions.
`n`	Number of items in `x`, `Q2` and `f`.
`ichk`	If set to zero, `hqstfun` will return a `null` value when x or μ^2 are outside the grid boundaries;[1] otherwise you will get a fatal error message.

By default, the structure function is computed for pdf-set `1`, or for the set selected by `hswitch`. Alternatively you can encode the pdf-set index in `istf`:

```
istf   →   10*iset + istf
```

The routine checks that for `icbt` = (1,2,3) = (c,b,t) the pdfs were evolved in the FFNS with $n_f = (3, 4, 5)$ and issues an error message if that is not the case. To relax the check you can prefix `icbt` by a minus sign: both the FFNS and the MFNS are then allowed with any number of fixed flavours. The VFNS does not make sense and is not allowed.

A linear combination of quarks and anti-quarks is specified in the input array `def(-6:6)` in FORTRAN or `def[13]` in C++. In these arrays the quark flavours are indexed as follows.

	$\bar{t}$	$\bar{b}$	$\bar{c}$	$\bar{s}$	$\bar{u}$	$\bar{d}$	g	d	u	s	c	b	t
C++	0	1	2	3	4	5	6	7	8	9	10	11	12
FORTRAN	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6

To calculate a structure function for more than one interpolation point, it is recommended to not execute `zmstfun` in a loop but to pass the entire list of interpolation points in a single call. The loop is then internally optimised for much greater speed.

Here is a snippet of code that, in combination with ZMSTF, calculates the d,u,s contribution, the charm contribution and the total F_2 (neglecting bottom and top) in charged lepton-proton scattering (the pdfs should have been evolved with $n_f = 3$ flavours).

```
      dimension x(100),Q2(100),F2dus(100),F2c(100),F2p(100)
      dimension proton(-6:6)
      data proton /4.,1.,4.,1.,4,.1.,0.,1.,4.,1.,4.,1.,4./ !divide by 9
        ..
      call zmstfun(2,    proton, x, Q2, F2dus, 100, ichk)
      call hqstfun(2, 1, proton, x, Q2, F2c  , 100, ichk)
      do i = 1,100
        F2p(i) = F2dus(i) + F2c(i)
      enddo
```

[1]For technical reasons a cut $Q^2 > 0.5$ GeV^2 is also imposed.

A List of FORTRAN routines and C++ prototypes

Subroutine or function	Description
`IHQVERS ( )`	HQSTF version number
`HQWORDS ( &ntotal, &nused )`	Words available, used
`HQFILLW ( istf, qmass, aq, bq, &nused )`	Fill weight tables
`HQDUMPW ( lun, 'filename' )`	Dump weight tables
`HQREADW ( lun, 'filename', &nused, &ierr )`	Read weight tables
`HQPARMS ( &qmass, &aq, &bq )`	Retrieve parameters
`HQQFRMU ( qmu2 )`	Convert μ_F^2 to Q^2
`HQMUFRQ ( Q2 )`	Convert Q^2 to μ_F^2
`HSWITCH ( iset )`	Switch pdf set
`HQSTFUN ( istf, icbt, def, x, Q2, &f, n, ichk )`	Structure function

Output arguments are prefixed with an ampersand (&).

C++ prototype

```
int     ihqvers ( )
void    hqwords ( int &ntotal, int &nused )
void    hqfillw ( int istf, double *qm, double aq, double bq, int &nu )
void    hqdumpw ( int lun, string filename )
void    hqreadw ( int lun, string filename, int &nused, int &ierr )
void    hqparms ( double *qmass, double &qa, double &qb )
double  hqqfrmu ( double qmu2 )
double  hqmufrq ( double Q2 )
void    hswitch ( int iset )
void    hqstfun ( int istf, int icbt, double *def, double *x,
                  double *Q2, double *f, int n, int ichk )
```