DISPred - a program to calculate deep inelastic scattering cross sections v1.0

J. Ferrando

University of Oxford

July 30, 2010

Abstract

A new package, DISPred, is described. The package can be used to calculate $e^{\pm}p$ deep inelastic scattering cross sections at Born level in Electroweak theory and at both leading and next-to-leading order in QCD.

1 Introduction

The package DISPred arose as a result of the need to produce predictions of deep inelastic scattering (DIS) electron-proton cross sections at next-to-leading order (NLO) in QCD for comparison to data in ZEUS publications. In the current version (1.0) predictions for the following are available at both leading order (LO) and NLO in QCD:

- the reduced/ double-diff cross sections for neutral current (NC) and charged current (CC) DIS
- the total cross section σ_{tot} for NC and CC DIS
- differential cross sections $\frac{d\sigma}{dQ^2}$, $\frac{d\sigma}{dx}$, $\frac{d\sigma}{dy}$ for NC and CC DIS

for collisions of unpolarized beams of electrons ($e^{\pm}$) and protons.

DISPred has been tested with LHAPDF 5.8.2 [1] and can produce predictions for the ZEUS-JETS [2] or HERA0.1 parton distribution functions (PDFs) in LHAPDF in the `.LHpdf` format, or any other PDF within LHAPDF in the `.LHgrid` format.

It produces output in ascii text format and can also produce histograms and graphs in ROOT-based [3] formats.

2 Leading Order Calculation

The LO QCD, Born-level electroweak, cross section is calculated according to the formulation given by Devenish and Cooper-Sarkar[4].

2.1 Reduced and Double Differential Cross Sections

2.1.1 NC DIS

The double differential cross section in NC scattering is:

$$\frac{\mathrm{d}^2\sigma_{\mathrm{NC}}^{e^{\pm}p}}{\mathrm{d}x\mathrm{d}Q^2} = \frac{2\pi\alpha^2 Y_+}{xQ^4}[F_2^{\mathrm{NC}}(x,Q^2) - \frac{y^2}{Y_+}F_L^{\mathrm{NC}}(x,Q^2) \mp \frac{Y_-}{Y_+}xF_3^{\mathrm{NC}}(x,Q^2)]. \tag{1}$$

Where, as is conventional, Q^2 is the virtuality of the exchanged boson, x is the momentum fraction of the struck parton in the infinite proton-momentum, F_i^{NC} are structure functions defined later, $Y_{\pm} = 1 \pm (1-y)^2$ and y is the inelasticity of the electron.

For the leading order calculation the structure functions are defined as follows:

$$F_2^{\mathrm{NC}} = \sum_i A_i^0(Q^2)(xq_i(x,Q^2) + x\bar{q}_i(x,Q^2)); \tag{2}$$

$$F_L^{\mathrm{NC}} = 0; \tag{3}$$

$$xF_3^{\mathrm{NC}} = \sum_i B_i^0(Q^2)(xq_i(x,Q^2) - x\bar{q}_i(x,Q^2));; \tag{4}$$

where A_i and B_i can be expressed in terms of the NC vector and axial-vector electroweak couplings to the quarks (electron) v_i (v_e) and a_i (a_e) and quark charge e_i as

$$A_i^0 = e_i^2 - 2e_i v_i v_e P_Z(Q^2) + (v_e^2 + a_e^2)(v_i^2 + a_i^2)P_Z^2(Q^2) \tag{5}$$

and

$$B_i^0 = -2e_i a_i a_e P_Z(Q^2) + 4a_i v_i v_e a_e P_z^2(Q^2) \tag{6}$$

and

$$P_Z(Q^2) = \frac{Q^2}{Q^2 + M_Z^2}\left(\frac{1}{\sin^2 2\theta_W}\right) \tag{7}$$

The reduced cross section for NC scattering is:

$$\tilde{\sigma}_{\mathrm{NC}}^{e^{\pm}p} = [F_2^{\mathrm{NC}}(x,Q^2) - \frac{y^2}{Y_+}F_L^{\mathrm{NC}}(x,Q^2) \mp \frac{Y_-}{Y_+}xF_3^{\mathrm{NC}}(x,Q^2)]. \tag{8}$$

2.1.2 CC DIS

The double differential cross section in CC scattering is:

$$\frac{\mathrm{d}^2\sigma_{\mathrm{CC}}^{e^{\pm}p}}{\mathrm{d}x\mathrm{d}Q^2} = \frac{G_F^2 M_W^2}{4\pi x(Q^2 + M_W^2)^2}[Y_+F_2^{\mathrm{CC}}(x,Q^2) - y^2F_L^{\mathrm{CC}}(x,Q^2) \mp Y_-xF_3^{\mathrm{CC}}(x,Q^2)]. \tag{9}$$

Where M_W is the mass of the W boson, G_F the Fermi coupling constant and the F_i^{CC} are defined at LO in QCD as[1]:

$$F_{2,e^+}^{CC} = x(d+s+\bar{u}+\bar{c}) \quad (10)$$
$$xF_{3,e^+}^{CC} = x(d+s-\bar{u}-\bar{c}) \quad (11)$$
$$F_{2,e^-}^{CC} = x(u+c+\bar{d}+\bar{s}) \quad (12)$$
$$xF_{3,e^-}^{CC} = x(u+c-\bar{d}-\bar{s}) \quad (13)$$

which leads to the following expressions for the reduced cross sections:

$$\tilde{\sigma}_{e^+}^{CC} = x[(1-y)^2(d+s)+\bar{u}+\bar{c}] \quad (14)$$
$$\tilde{\sigma}_{e^-}^{CC} = x[(1-y)^2(\bar{d}+\bar{s})+u+c] \quad (15)$$

2.2 Single Differential and Total Cross Sections

In order to calculate the single differential and total cross sections, the expression for the double differential cross sections is integrated over the allowed regions of Q^2, x and y using the VEGAS [5] algorithm as implemented in the GNU Scientific Library[6]. The number of calls used in VEGAS may be specified via the control cards. Differential cross sections may also be calculated at a "point"; in this case the width of the bin which contains the point is multiplied by a predetermined factor (which can be chosen in the control cards) to provide an approximate calculation. If no input points are specifed via cards, the option "AUTO" may be chosen which makes the program `DISPrediction` calculate the differential cross sections at the centroid of the bin.

3 Next-to-Leading Order Calculation

3.1 Reduced and Double Differential Cross Sections

The implementation of QCDNUM 16.13 [7] included in LHAPDF is used in DISPred to evaluate structure functions F_2, F_L and F_3 at NLO in QCD. The prescription used by the ZEUS collaboration for the ZEUS JETS fit [8, 2] has been adopted. As such `DISPred` can perform the QCD evolution for the ZEUS-JETS and ZEUS-S fits and use the `.LHpdf` format files from LHAPDF for this. In the case of other PDFs `DISPred` can fill a Q^2, x grid for QCDNUM using the values from the tt .LHGrid file. The structure functions are then generated from this grid.All other aspects of the reduced cross section cross section are the same as for the leading order case.

Predicted NLO reduced cross sections in CC DIS made using DISPred for e^+p collisons with proton energy 920 GeV and positron energy 27.56 GeV are shown in figure 1. Predictions are shown for the PDF sets ZEUS-JETS [2], MSTW08 [9], CTEQ66 [10] and HERAPDF1.0 [11]. In addition the uncertainties for the ZEUS-JETS predictions are shown as a yellow band.

[1] In the expressions shown above the small, Cabbibo-suppressed, contribution from the b-quark is neglected, it is however included in the calculation made by DISPred. In v1.0 DISPred is only suitable for use for the HERA energy regime and so top quark contributions are not included

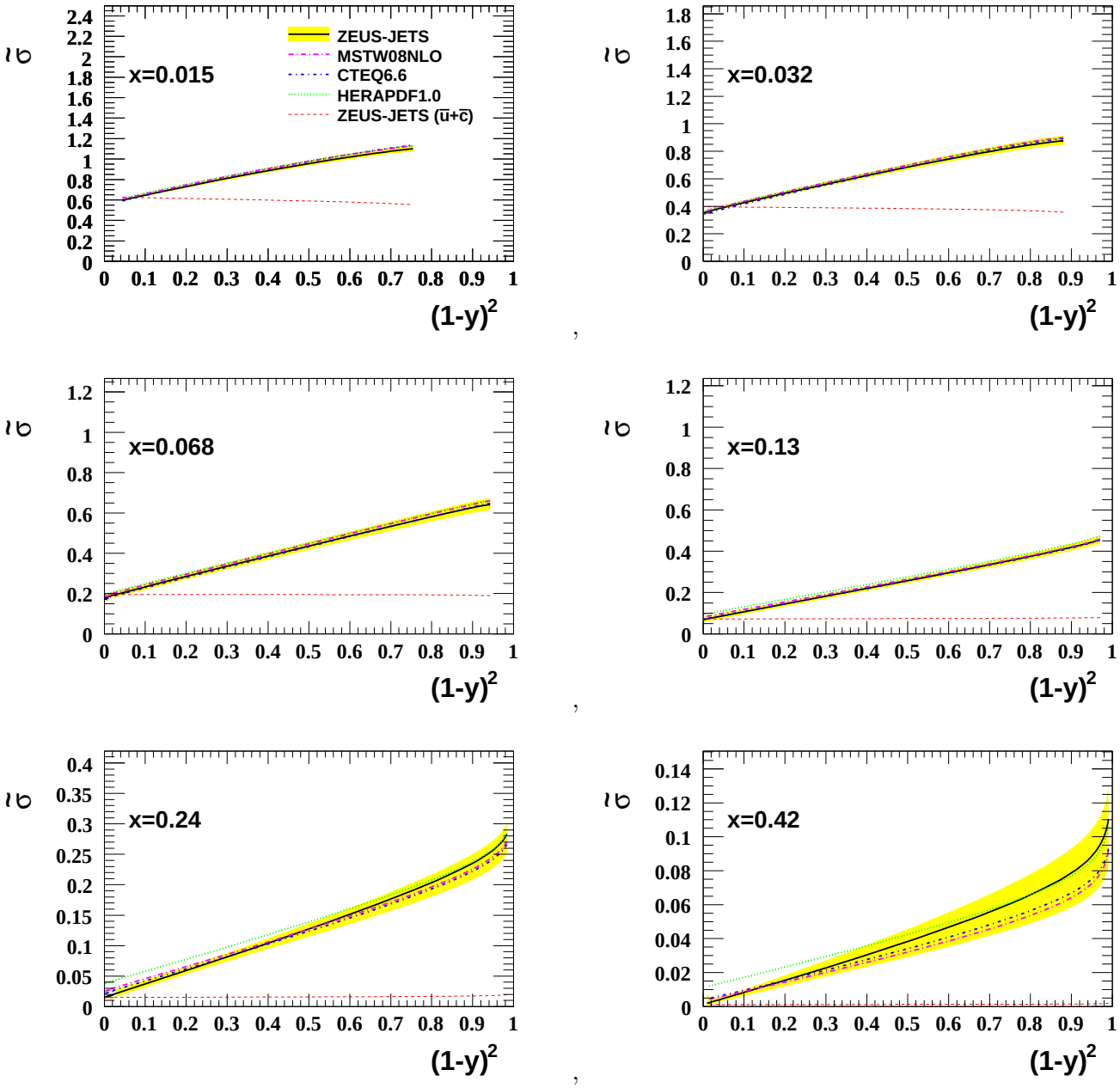

Figure 1: Predictions of $\tilde{\sigma}$ at NLO in QCD for CC DIS in e^+p collisions with proton energy 920 GeV and positron energy 27.56 GeV.

3.2 Single Differential and Total Cross Sections

The single and total cross sections at NLO are calculated in precisely the same manner as for the LO calculations except that the NLO expressions for the structure functions are used.

4 Installation and Usage

4.1 Requirements

The code has been tested on GNU/Linux systems and as such the following packages are required for installation:

- GNU autoconf automake
- The GNU scientific library
- LHAPDF
- ROOT

4.2 Installation

Tarballs of the package may be downloaded from `http://www.hepforge.org/downloads/dispred`. After downloadig the tarball, the package may be installed with:

```
tar -zxvf DISPred-1.0.tgz
cd DISPred
./configure --prefix=<installation directory>
make
make install
```

This assumes that `root-config` and `lhapdf-config` are already in your path. Do not choose `<installation directory>` to be the same as the location of the expanded tar file. It is now possible to run the example program or to include the DISPred libraries in your own code.

4.3 Example Program: DISPrediction

Within `<installation directory>/bin` there is a program `DISPrediction` which can be used to produce predictions for *ep* DIS cross sections. This program takes as input a cards file e.g.:

```
DISPrediction example.cards
```

An example set for testing is available in the `example` subdirectory of the tarball. The available options for the cards file are summarised in table 1.

4.4 DISPred Library and Classes

The DISPred packagedprovides a library as well as the `DISPrediction` executable. This library makes it easy to construct programmes that calculate DIS cross sections. An example of a simple programme is in fact `DISPrediction` itself, which is very short:

Type	Name	Default Value	Meaning
int	ElectronCharge	1	choose e^+ or e^-
int	VegasCalls"	50000	Calls to Vegas for integration
string	PDFSetFileName	"ZEUS2005_ZJ.LHpdf"	Name of PDF file to use
int	PDFSubSet	0	Subset of PDF to use
string	QCDCalculationLevel	"LO"	Can be "LO" or "NLO"
string	DISProcess	"NC"	Choose CC or NC
double	ZBosonMass	91.1876	M_Z (GeV)
double	WBosonMass	80.398	M_Z (GeV)
double	AlphaEM	7.297352570$\times 10^-3$	$\alpha_{\rm EM}$
double	GFermi	1.1663710^{-5}	G_F
double	TopMass	171.2	M_t
double	BottomMass	4.20	M_b
double	Vub	41.2e-3	CKM V_{ub}
double	Vcb	3.93e-3	CKM V_{cb}
double	Sin2ThetaW	0.22308	$\sin^2\theta_W$
double	Sin2ThetaC	0.05	$\sin^2\theta_C$
double	CouplingVu	0.203	v_u SM= $0.5 - 4 * \sin^2\theta_W$
double	CouplingVd	-0.351	v_d SM= $-0.5 + 2 * \sin^2\theta_W$
double	CouplingVe	-0.00538	v_e SM=$-0.5 + 2 * \sin^2\theta_W$
double	CouplingAu	0.5	a_u
double	CouplingAd	-0.5	a_d
double	CouplingAe	-0.5	a_e
string	ReducedCrossSection	"OFF"	can be OFF or ON
string	ReducedCrossSectionBins	"q2xpoints.dat"	file containing points for $\tilde{\sigma}$
double	DiffBinPointScale	1e-6	Fraction of bin width for $\frac{d\sigma}{dQ^2}$ etc.
string	DSigmaDQ2	"OFF"	can be OFF or ON
string	DSigmaDQ2Bins	"q2bins.dat"	file containing bins for
string	DSigmaDQ2Points	"AUTO"	file with points for $\frac{d\sigma}{dQ^2}$
string	DSigmaDX	"OFF"	can be OFF or ON
string	DSigmaDXBins	"xbins.dat"	file containing bins for $\frac{d\sigma}{dx}$
string	DSigmaDXPoints	"AUTO"	file with points for
string	DSigmaDY	"OFF"	can be OFF or ON
string	DSigmaDYBins	"ybins.dat"	file containing points for $\frac{d\sigma}{dy}$
string	DSigmaDYPoints	"AUTO"	file with points for
double	Q2Min	0.0	minimum Q^2
double	Q2Max	100000.0	maximum Q^2
double	XMin	0.0	minimum x
double	XMax	1.0	maximum x
double	YMin	0.0	minimum y
double	YMax	1.0	maximum y
double	ELepton	27.5	Electron beam energy
double	EProton	920.0	Proton beam energy
string	ROOTOutputFile	"DISPredOut.root"	Output file for ROOT

Table 1: Available control cards for DISPrediction.

```
#include <iostream>
#include "DISPredictor.h"

using namespace DISPred;

int main (int argc, char **argv){
std::cout << "DISPrediction v1.0 - 31 Mar 2010" << std::endl;
// Create instance of DISPredictor
DISPred::DISPredictor *DISPred= DISPred::DISPredictor::Instance(); // initialise
from control cards provided via command line
DISPred->Initialise(argc,argv);  DISPred->CalculateCrossSections();
DISPred->PrintResults();
std::cout << "DISPrediction v1.0 - Run finished Succesfully" << std::endl;
DISPred->WriteOutput();
return 0;
}
```

All classes are part of the name space `DISPred`.

4.4.1 The `ControlCards` Class

The control cards class is used to handle configuration options that can be read in from a text file. It is implemented as a singleton class. Available methods for the class are detailed below.

`ControlCards* Instance()`: Returns a pointer to the instance of control cards.

`void AddCardDouble(const std::string key, const double defval)`: Defines a card with name `key` and with a default double precision value `defval`.

`void AddCardInt(const std::string key, const int defval)`: Defines a card with name `key` and with a default value `defval` which is an integer.

`void AddCardString(const std::string key, std::string defval)` : Defines a card with name `key` and with a default value `defval` which is a string.

`void AddCardVector(const std::string key, const std::vector<double> defval)`: Defines a card with name `key` and with a default value `defval` which is a vector of double precision values.

`int readKeys(const char* fileName)`: Reads in card values from the file with name `fileName`.

`double fetchValueDouble(const std::string& key)`: fetch the value of card `key`.

`int fetchValueInt(const std::string& key)`: fetch the values of card `key`.

`std::string fetchValueString(const std::string& key)`: fetch the value of card `key`.

`std::vector<double> fetchValueVector(const std::string& key)`: fetch the values of card `key`.

`void printCards()`: Print current card values to stdout.

4.4.2 The `DISPredictor` class

The `DISPredictor` class is a singleton class that is the workhorse of DISPred. It has many public methods.

`static DISPredictor* Instance()`: returns the instance of `DISPredictor`.

`void Initialise(int my_argc, char **my_argv)`: Initialise DISPredictor based on a cards file name which can come directly from `stdin`.

`void CalculateCrossSections()`: Calculate cross sections as configured in the cards.

`void InitPDF(int subset)`: Initialise the chosen PDF set.

`void PrintResults()`: Print results to `stdout`.

`void WriteOutput()`: Write the output rootfile.

`double CalculateReducedCrossSection(double x, double q2)`: Calculate a NC DIS reduced cross section.

`double CalculateCCReducedCrossSection(double x, double q2)`: Calculate a CC DIS reduced cross section.

`double CalculatePropagator(double q2, double x)`: Calculate the NC propagator.

`double CalculateCCPropagator(double q2, double x)`: Calculate the CC propagator.

`double CalculateDSigmaDQ2(double q2min,double q2max)`: Calculate $\frac{d\sigma}{dQ^2}$.

`double CalculateDSigmaDX(double xmin,double xmax)`: Calculate $\frac{d\sigma}{dx}$.

`double CalculateQ2DSigmaDQ2(double q2min,double q2max)`: Calculate $Q^2\frac{d\sigma}{dQ^2}$.

`double CalculateXDSigmaDX(double xmin,double xmax)`:Calculate $x\frac{d\sigma}{dx^2}$.

`double CalculateDSigmaDY(double ymin,double ymax)`: Calculate $\frac{d\sigma}{dy}$.

`double CalculateYDSigmaDY(double ymin,double ymax)`: Calculate $y\frac{d\sigma}{dy}$.

`double S()`: Return the centre-of-mass energy squared.

4.4.3 The `RedSigmaPoint` class

The `RedSigmaPoint` class is a simple class for storing information about double-differential cross sections points. For each point the Q^2(`_q2`), x(`_x`), $\tilde{\sigma}$(`_redsigma`) and $\frac{d^2\sigma}{dQ^2dx}$ (`_d2sdq2dx`).

`RedSigmaPoint(double q2, double x)`: constructor that creates a point with `_q2=q2` and `_x=x` and other values 0.

`RedSigmaPoint(double q2, double x,double redsigma)`: constructor that creates a point with `_q2=q2` and `_x=x`, `_redsigma=redsigma` and `_d2sdq2dx=0`.

`RedSigmaPoint(double q2, double x,double redsigma, double d2sdq2dx )`: constructor that creates a point with `_q2=q2` and `_x=x`, `_redsigma=redsigma` and `_d2sdq2dx=d2sdq2dx`.

`RedSigmaPoint()`: Constructor with all vlaues set to 0;

`double Q2()`: returns `_q2`.

`double X()`: returns `_x`.

`double RedSigma()`: returns `_redsigma`.

`double D2sDQ2Dx()`: returns `_d2sdq2dx`.

`void SetRedSigma(double reduced)`: Set `_redsigma`.

`void SetD2sDQ2Dx(double reduced)`: Set `_d2sdq2dx`.

`void Print()`: Print out information.

`void PrintShort()`: Briefly print out information.

4.4.4 The `RedSigmaGrid` class

The `RedSigmaGrid` class inherits from a `std::vector<RedSigmaPoint>`. With the following extra methods:

`void Print()`: Print out information.

`void PrintShort()`: Briefly print out information.

4.4.5 The `DiffSigmaPoint` class

The `DiffSigmaPoint` class is a simple class for storing information about single-differential cross sections at a point. A point in the variable of choice called `_var` and the differential cross section `_diffsigma` are stored. The following public methods are available.

`DiffSigmaPoint()`: Default constructor, sets `_var` to 1.5 and `_diffsigma` to 0.

`DiffSigmaPoint(double var)`: Constructor that creates a point at `_var= var` with `_diffsigma`=0.

`DiffSigmaPoint(double var, double diffsigma)`: Constructor that creates a point at `_var= var` with `_diffsigma=diffsigma`.

`void Print()`: Print out information.

`void PrintShort()`: Briefly print out information.

`double Var()`: returns `_var`.

`double DiffSigma()`: returns `_diffsigma`.

`void SetDiffSigma(double diffsigma)`: sets `_diffsigma` to `diffsigma`.

5 Root Output

When DISPred produces an output root file, then a `TTree` and several histograms and graphs are produced.

5.1 Root `TTree`

A `TTree` called `ReducedCrossSections` is produced. The variables in this tree are listed in table 2.

Type	Variable Name	Description
int	point	An integer giving the ID of the point
double	Q2	The Q^2 of the point
double	x	The x of the point
double	ddiffsigma	The double-differential cross section
double	redsigma	The reduced cross section
double	d	The d PDF at this point
double	dbar	The $\bar{d}$ PDF at this point
double	u	The u PDF at this point
double	ubar	The $\bar{u}$ PDF at this point
double	s	The s PDF at this point
double	sbar	The $\bar{s}$ PDF at this point
double	c	The c PDF at this point
double	cbar	The $\bar{c}$ PDF at this point
double	b	The b PDF at this point
double	bbar	The $\bar{b}$ PDF at this point

Table 2: Tree variables in the root output file.

5.2 Root Histograms

Six `TH1D` objects are produced:

`DSigmaDQ2`: Binwise $\frac{d\sigma}{dQ^2}$;

`DSigmaDX`: Binwise $\frac{d\sigma}{dx}$;

`DSigmaDY`: Binwise $\frac{d\sigma}{dy}$.

`Q2DSigmaDQ2`: Binwise $Q^2\frac{d\sigma}{dQ^2}$;

`XDSigmaDX`: Binwise $x\frac{d\sigma}{dx}$;

`YDSigmaDY`: Binwise $y\frac{d\sigma}{dy}$.

5.3 Root Graphs

Three `TGraphAsymErrors` objecs are produced:

`GraphDSigmaDQ2`: Pointwise $\frac{d\sigma}{dQ^2}$;

`GraphDSigmaDX`: Pointwise $\frac{d\sigma}{dx}$;

`GraphDSigmaDY`: Pointwise $\frac{d\sigma}{dy}$.

6 Summary

This manual for the DISPred package v1.0 has outlined the features currently implemented together with a simple example programme that will make predictions for DIS cross sections in *ep* scattering. The code and most up-to-date information are hosted by hepforge at: `http://projects.hepforge.org/dispred/`.

Acknowledgements

The author wishes to thank C. Gwenlan for help with QCDNUM and cross checks of the results from DISPred, M. Sutton for testing the code, A. Tapper for providing code that is used for the implementation of control cards and R. Ciesielski, A. Cooper-Sarkar, K. Oliver, E. Tassi and M. Turcato for feedback on the results.

References

[1] M. R. Whalley, D. Bourilkov, and R. C. Group, *The Les Houches Accord PDFs (LHAPDF) and Lhaglue*, hep-ph/0508110.

[2] **ZEUS** Collaboration, S. Chekanov *et al.*, *An NLO QCD analysis of inclusive cross-section and jet- production data from the zeus experiment*, *Eur. Phys. J.* **C42** (2005) 1–16, [hep-ph/0503274].

[3] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, *Nucl. Instrum. Meth.* **A389** (1997) 81–86.

[4] R. Devenish and A. Cooper-Sarkar, *Deep inelastic scattering*, . Oxford, UK: Univ. Pr. (2004) 403 p.

[5] G. P. Lepage, *A New Algorithm for Adaptive Multidimensional Integration*, *J. Comput. Phys.* **27** (1978) 192.

[6] M. Galassi *et al.*, *GNU Scientific Library Reference Manual*, . (3rd Ed.) ISBN 0954612078.

[7] M. Botje, *A QCD analysis of HERA and fixed target structure function data*, *Eur. Phys. J.* **C14** (2000) 285–297, [hep-ph/9912439].

[8] **ZEUS** Collaboration, S. Chekanov *et al.*, *A ZEUS next-to-leading-order QCD analysis of data on deep inelastic scattering*, *Phys. Rev.* **D67** (2003) 012007, [hep-ex/0208023].

[9] A. D. Martin, W. J. Stirling, R. S. Thorne, and G. Watt, *Parton distributions for the LHC*, arXiv:0901.0002.

[10] W. K. Tung *et al.*, *Heavy quark mass effects in deep inelastic scattering and global QCD analysis*, *JHEP* **02** (2007) 053, [hep-ph/0611254].

[11] **H1 and ZEUS** Collaboration, F. D. Aaron *et al.*, *Combined Measurement and QCD Analysis of the Inclusive ep Scattering Cross Sections at HERA*, *JHEP* **01** (2010) 109, [arXiv:0911.0884].