In[1]:=
```
Clear["Global`*"];
```

In[2]:=
```
$Version
```

Out[2]=
```
10.4.1 for Linux x86 (64-bit) (April 11, 2016)
```

Demo file for ManeParse Package Version 2.2

Version 2.2
20 May 2016

Comments and questions to:

Eric Godat egodat@smu.edu
Ben Clark dbclark@smu.edu
Fred Olness olness@physics.smu.edu

Set Directory

This example notebook is written with relative directories and is intended to be run within the folder extracted from the tarball. Uncomment and modify the code below to set a different directory for the LHA files.

In[3]:=
```
(* This just drops the leading path
 info to make the list of files easier to read *)
dropPath = Take[(FileNameSplit /@ #) // Transpose, -1][[1]] &;
```

In[4]:=
```
Directory[];
here = Directory[]
```

Out[5]=
```
/users/olness/clark/trunk/ManeParse/Demo
```

In[6]:=
```
(* If there is a problem with the Mathematica working directory,
you can enter it manually here *)
SetDirectory[here]
```

Out[6]=
```
/users/olness/clark/trunk/ManeParse/Demo
```

In[7]:=
```
(* This shows what files should be in this main directory *)
FileNames["*.*", here] // dropPath
```
Out[7]=
```
{buil-demo.py, Demo.nb, Demo.pdf, manual_v1.nb, manual_v1.pdf, noe2.perl}
```

Setup Other Directories

In[8]:=
```
dirPackages = here <> "/MP_packages";
dirFilesLHA = here <> "/PDF_Sets/LHA";
dirFilesPDS = here <> "/PDF_Sets/PDS";

dirCT10 = dirFilesLHA <> "/CT10";
dirMSTW = dirFilesLHA <> "/MSTW2008lo68cl";
dirNNPDF = dirFilesLHA <> "/NNPDF30_nlo_as_0118";
```

In[14]:=
```
dirCT10pds = dirFilesPDS <> "/ct10.pds";
dirCTEQ66 = dirFilesPDS <> "/ctq66m.pds";
```

In[16]:=
```
dirPackages
```
Out[16]=
```
/users/olness/clark/trunk/ManeParse/Demo/MP_packages
```

Load the package

Loading the main package provides many useful functions

In[17]:=
```
Get[dirPackages <> "/pdfParseLHA.m"]
```

```
 - Required Package: pdfCalc --Loaded -


================================================================


 - pdfParseLHA -
Version:  1.0
Authors: E.J. Godat, D.B. Clark & F.I. Olness

Please cite: **************
http://ncteq.hepforge.org/code/pdf.html

For a list of available commands, enter: ?pdf*

================================================================
```

In[18]:=
```
Get[dirPackages <> "/pdfParseCTEQ.m"]
```

```
================================================================

 - pdfParseCTEQ -
Version:  1.0
Authors: D.B. Clark, E.J. Godat & F.I. Olness

Please cite: **************
http://ncteq.hepforge.org/code/pdf.html

For a list of available commands, enter: ?pdf*

================================================================
```

In[19]:=
```
Get[dirPackages <> "/pdfErrors.m"]
```

```
================================================================

 - pdfErrors -
Version:  1.0
Authors: D.B. Clark, E.J. Godat & F.I. Olness

Please cite: **************
http://ncteq.hepforge.org/code/pdf.html

For a list of available commands, enter: ?pdf*

================================================================
```

All functions begin with ' pdf'. To obtain a list of available functions, type the command '?pdf*'.

In[20]:=
```
? pdf*
```

▼ pdfCalc`

pdfAlphaS	pdfGetQlist	pdfNumQpartition	pdfSetListDisplay
pdfFlavor	pdfGetXlist	pdfReset	pdfSetXpower
pdfFunction	pdfLowFunction	pdfSetInterpolator	pdfXmin
pdfGetInfo	pdfLuminosity	pdfSetList	

▼ pdfErrors`

pdfFamilyFunction	pdfHessianError	pdfMCCentralInterval	pdfMCError
pdfHessianCorrelation	pdfMCCentral	pdfMCCorrelation	

▼ pdfParseCTEQ`

pdfFamilyParseCTEQ	pdfParseCTEQ

▼ pdfParseLHA`

pdfFamilyParseLHA	pdfParseLHA

Individual file manipulation

Individual files in either LHA or PDS format can be parsed using the functions loaded from the packages. Here we demonstrate the LHA parsing function

In[21]:=
```
? pdfParseLHA
```

pdfParseLHA[fileNameInfo, fileNameData, [verbose]]: This function reads an individual .info file and .data file specified by *fileNameInfo* and *fileNameData*, respectively, into memory.

The function returns a set number that corresponds to the listing of the .dat file in *pdfSetList*.

Additionally, the function checks that the number and the order of the flavors are the same in both files.

The optional input allows the user to supress the output of this function by choosing *verbose* to be *False*.

In[22]:=
```
datfiles = FileNames["*.dat", dirCT10];(* This is a set of LHA PDFs *)
infofile = FileNames["*.info", dirCT10];(* This is the associated info file *)
```

In[24]:=
```
sample = pdfParseLHA[infofile[[1]], datfiles[[1]]]
```

```
Successfully read /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10.info.

Successfully read
 /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0000.dat.
```

Out[24]= 1

In[25]:=
```
sample2 = pdfParseLHA[infofile[[1]], datfiles[[2]]]
```

```
Successfully read /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10.info.

Successfully read
 /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0001.dat.
```

Out[25]= 2

Calling the pdfSetList variable will give a key to the data files in memory. The information is displayed as:
{SetNumber,FileName, maxFlavor, numberValence}

In[26]:=
```
pdfSetList // TableForm
```

Out[26]//TableForm=

1	/users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0000.dat	5
2	/users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0001.dat	5

Files can be added to memory without a name. All files can be called by their set numbers.

In[27]:=
```
pdfParseLHA[infofile[[1]], datfiles[[3]]]
```

```
Successfully read /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10.info.

Successfully read
 /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0002.dat.
```

Out[27]= 3

In[28]:=
```
pdfSetList // TableForm
```

Out[28]//TableForm=

1	/users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0000.dat	5
2	/users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0001.dat	5
3	/users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10_0002.dat	5

Open and parse single files in any order. You may assign names to each pdf set. Each PDF set is identified by a SetNumber.

Batch file manipulation

Resetting memory can be accomplished with the pdfReset command.

In[29]:=
```
pdfReset[]
```

```
Default Mathematica interpolator will be used.

All internal variables have been reset.
```

The set list is now empty.

In[30]:=
```
pdfSetList // TableForm
```

Out[30]//TableForm=
```
{}
```

The pdfFamilyParseLHA command can be used to store a family of LHA info and dat files in memory. The function returns a list of values that can be associated with the family.

In[31]:=
```
? pdfFamilyParseLHA
```

pdfFamilyParseLHA[path, [fileType]]: This function reads all
the files of type *fileType* in the directory *path* and stores them in memory.

The function returns a list of set numbers that can be used to define
a list. These set numbers correspond to the listing of the .dat files in *pdfSetList*.

The optional input *fileType* has a default value of "*.dat".

Example:
pdfFamilyParseLHA["MyGrids","ct10*.dat"] reads all
.dat files in the subdirectory "MyGrids" beginning with "ct10" into memory.

First we import the ct10 dat files. The family will include the info file, the central value(set #1) and 52 eigenvector error sets. The family name can be defined at this point.

In[32]:=
```
ct10 = pdfFamilyParseLHA[dirCT10, "*.dat"]
```

```
Successfully read /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10.info.
Included 53 files in the PDF family.
```

Out[32]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53}
```

Test PDFs

The function "pdf" is left to be defined by the user. Access to the PDF of the set is given by pdfFunction. The function has the canonical form: **pdfFunction[setNumber, flavorNumber, x, Q]**. If the function is not defined, pdfFunction returns NULL.

In[33]:=
```
? pdfFunction
```

pdfFunction[setNumber, flavor, x, Q]: This function returns the interpolated value of the PDF for the .pds/.dat file specified by *setNumber*, for the given flavor and value of Bjorken *x* and scale *Q*.

Warning: The results of this function are only reliable between the maximum and minimum values of x and Q in the .pds/.dat file.

In[34]:=
```
pdfFunction[1, 1, .1, 10]
```

Out[34]=
```
3.96968
```

In[35]:=
```
Clear[pdf]
pdf[iset_?IntegerQ, ipart_?IntegerQ, x_?NumericQ, q_?NumericQ] :=
 pdfFunction[iset, ipart, x, q]
```

In[37]:=
```
pdf[1, 1, .1, 10]
pdf[2, 1, 0.1, 10]

centralvalue = 1;

pdf[centralvalue, 1, 0.1, 10]
```

Out[37]= 3.96968

Out[38]= 3.95809

Out[40]= 3.96968

Check Timing :

In[41]:=
```
Table[pdf[iset0, 0, RandomReal[], 10.], {i, 1, 1000}] // Timing // First
```

Out[41]= 0.000747

In[42]:=
```
Table[pdf[iset0, 0, 1 / i, 10.], {i, 1., 1000}] // Timing // First
```

Out[42]= 0.001035

Check sum rule:

In[43]:=
```
Off[NIntegrate::izero]
Off[NIntegrate::ncvb]
q0 = 2.0;
iset0 = 1;
```

In[47]:=
```
(* This can take a while *)
tab = Table[ NIntegrate[ x pdf[iset0, ipart, x, q0], {x, 0, 1}], {ipart, -5, 5, 1}]; //
 Timing
Plus @@ tab
```

Out[47]= {2.65612, Null}

Out[48]= 0.999837

In[49]:=
```
flavorlist = {};
```

In[50]:=
```
For[i = -5, i ≤ 5, i++, AppendTo[flavorlist, pdfFlavor[i]];]
```

In[51]:=
```
flavorlist
```

Out[51]= {bbar, cbar, sbar, ubar, dbar, gluon, down, up, strange, charm, bottom}

In[52]:=
```
{Range[-5, 5], flavorlist, Round[100 tab]} // Transpose // Grid[#, Frame → All] &
```

Out[52]=

-5	bbar	0
-4	cbar	0
-3	sbar	2
-2	ubar	3
-1	dbar	4
0	gluon	42
1	down	15
2	up	32
3	strange	2
4	charm	0
5	bottom	0

Example: Plotting Single Functions

First we find the minimum value of x for our pdf family.

In[53]:=
```
? pdfXmin
```

pdfXmin[setNumber]: This function returns the minimum x value in the PDF set *setNumber*.

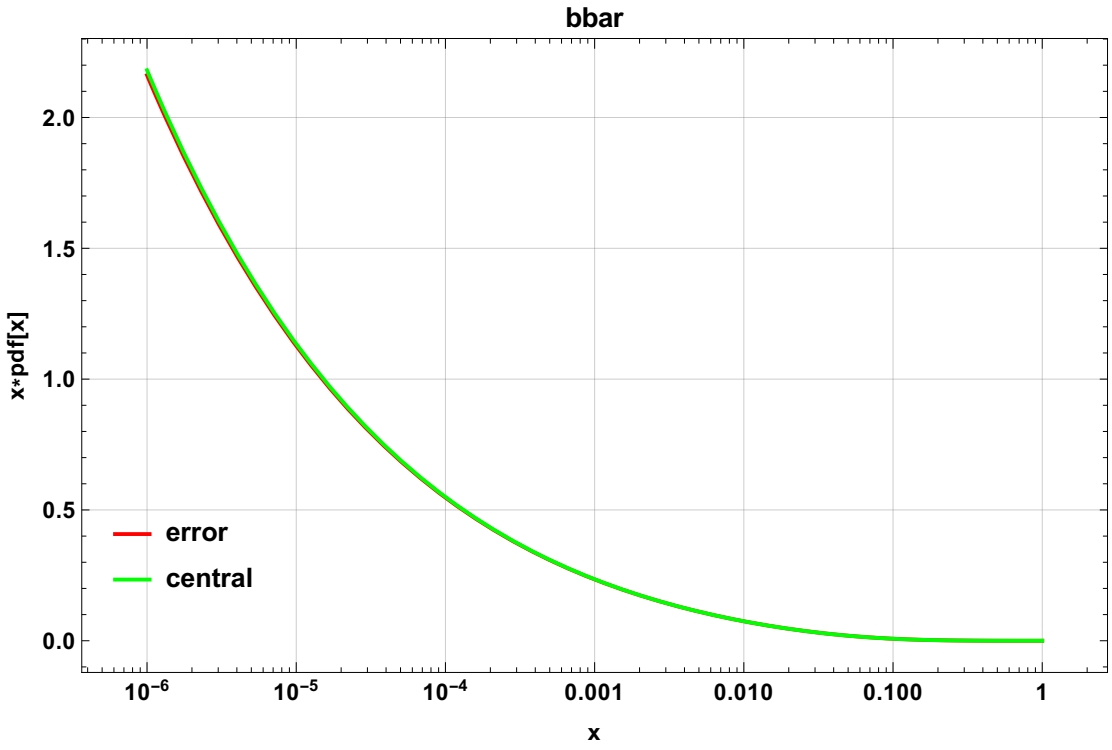
In[54]:=
```
xMin = pdfXmin[1]
```

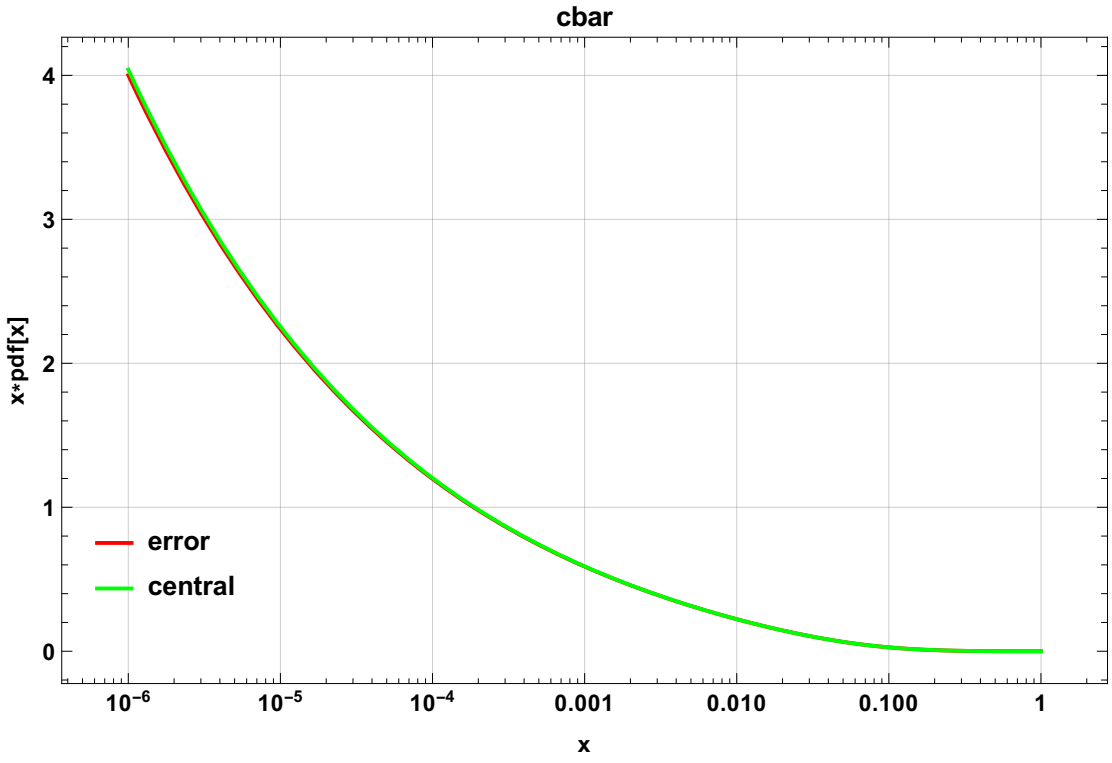
Out[54]= $1. \times 10^{-8}$

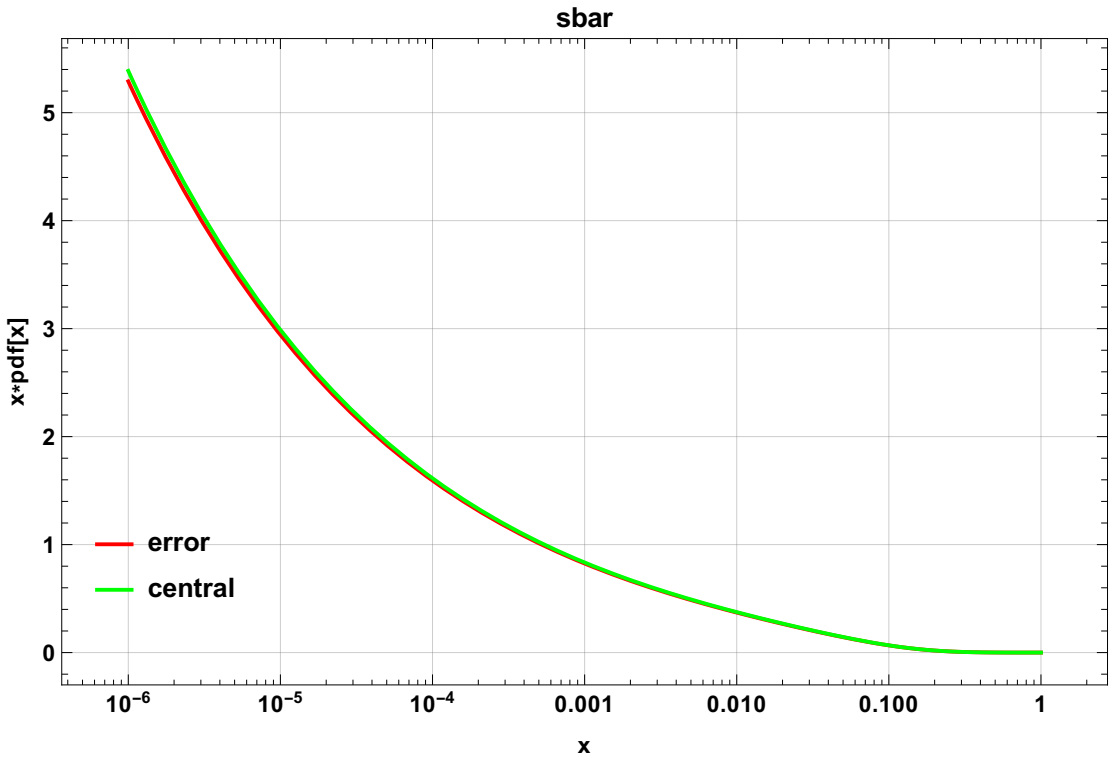
We will produce plots of x*pdf(x,Q) for all flavors with the central value in red and the first error set in green. The flavor can be called with the command pdfFlavor[flavor].

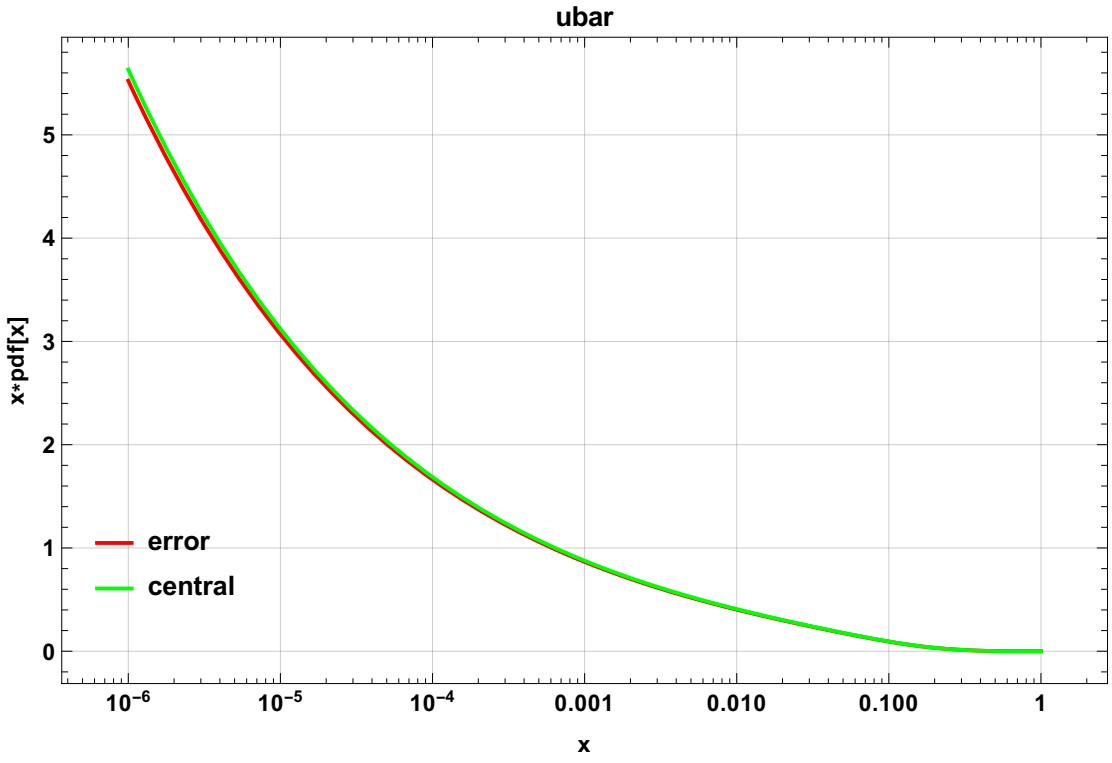
In[55]:=
```
q0 = 10;
centralvalue = 1;
errorvalue = 2;
```

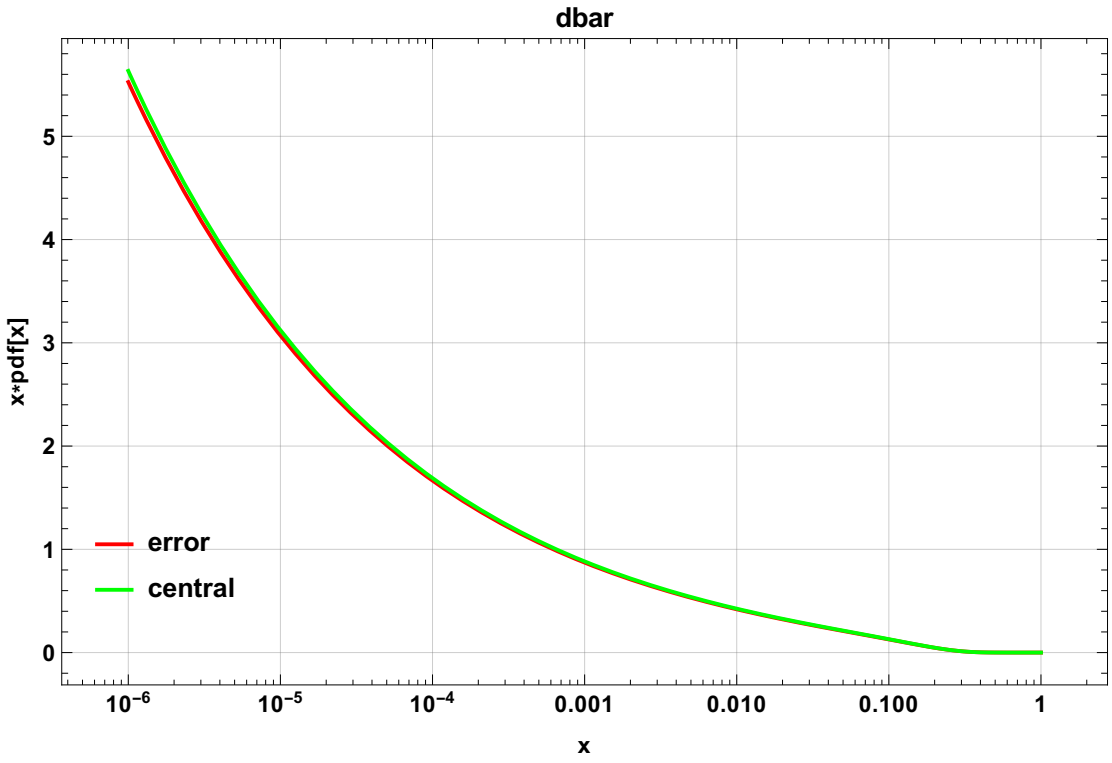
In[58]:=
```
For[i = -5, i ≤ 5, i++,
 LogLinearPlot[x {pdf[errorvalue, i, x, q0], pdf[centralvalue, i, x, q0]} // Evaluate ,
   {x, xMin * 100, 1},
   PlotStyle → {Directive[Red, Thick], Directive[Green, Thick]},
   PlotLabel → pdfFlavor[i] ,
   FrameLabel → {"x", "x*pdf[x]"},
   ImageSize → Large,
   PlotRange → All,
   Frame → True,
   BaseStyle → {FontWeight → "Bold", FontSize → 12},
   GridLines → Automatic,
   PlotLegends → Placed[{"error", "central"}, {0.1, 0.18}]] // Print
]
```

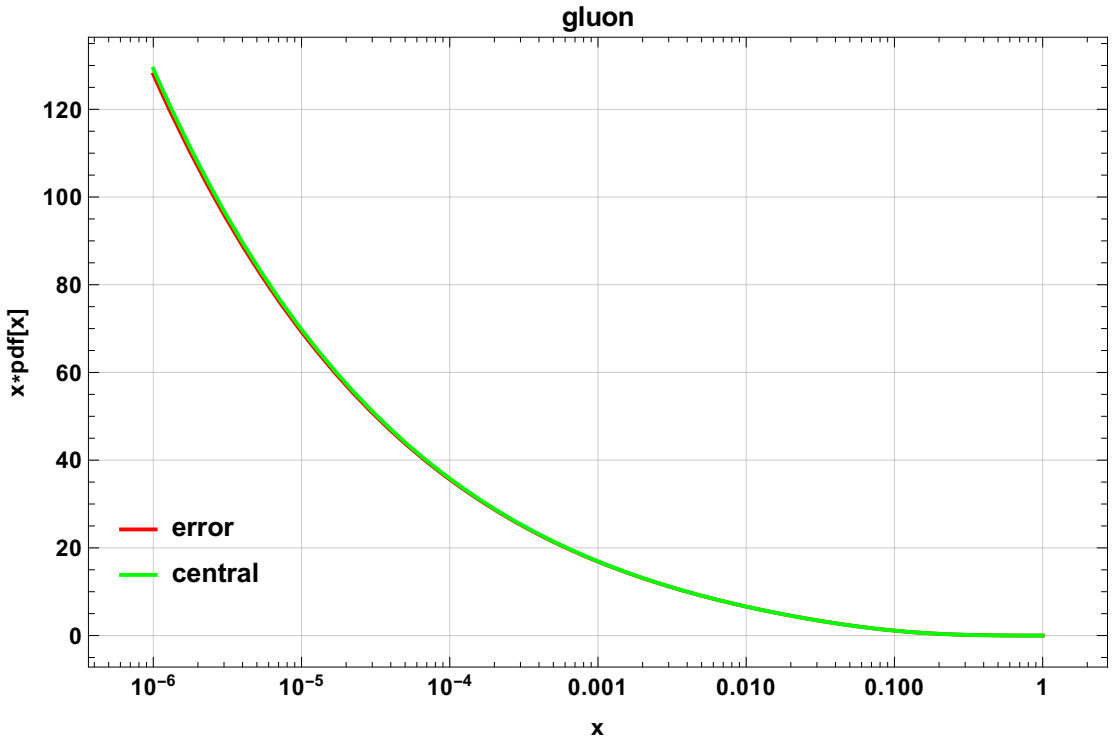


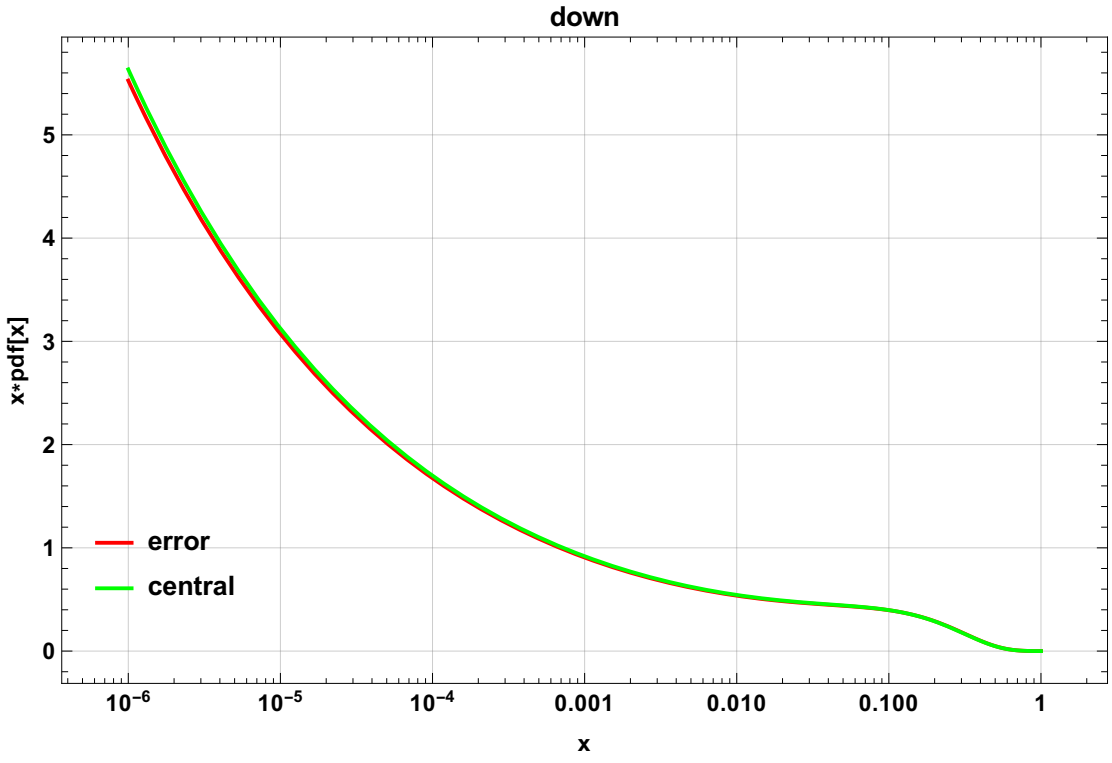

cbar
x*pdf[x]
x
error
central
0
1
2
3
4
10^-6
10^-5
10^-4
0.001
0.010
0.100
1

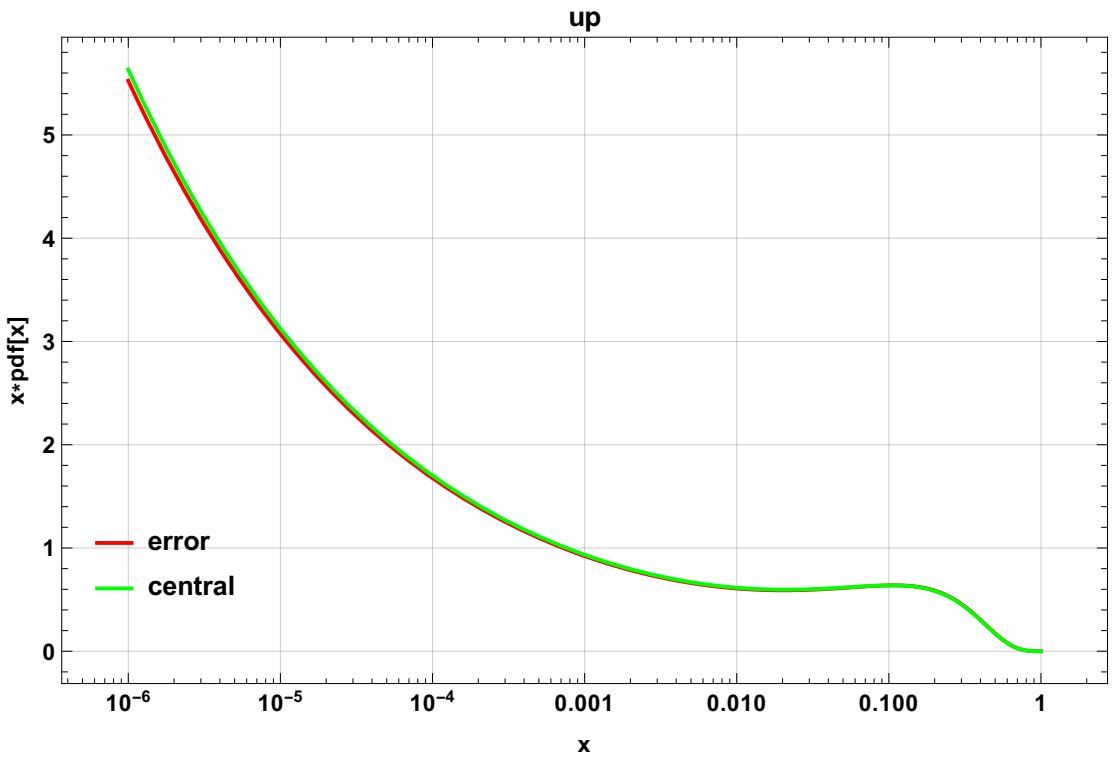

sbar
x*pdf[x]
x
error
central
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1

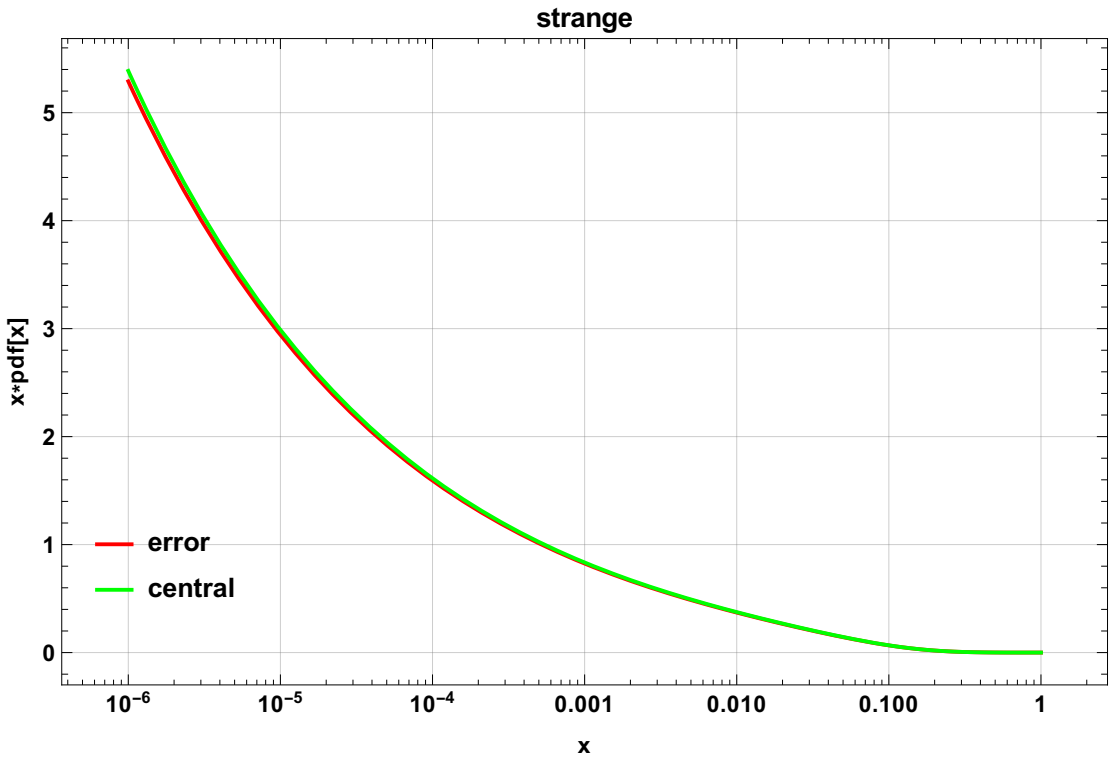

ubar
x*pdf[x]
0
1
2
3
4
5
error
central
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x

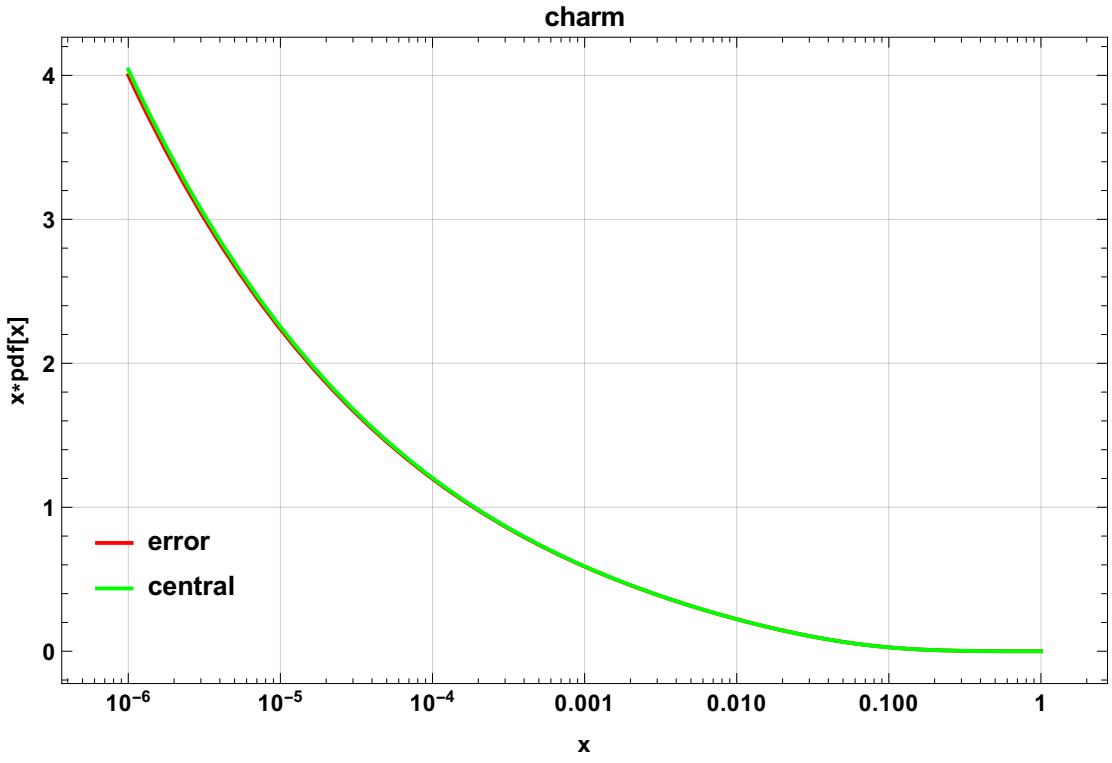

dbar
x*pdf[x]
0
1
2
3
4
5
error
central
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x

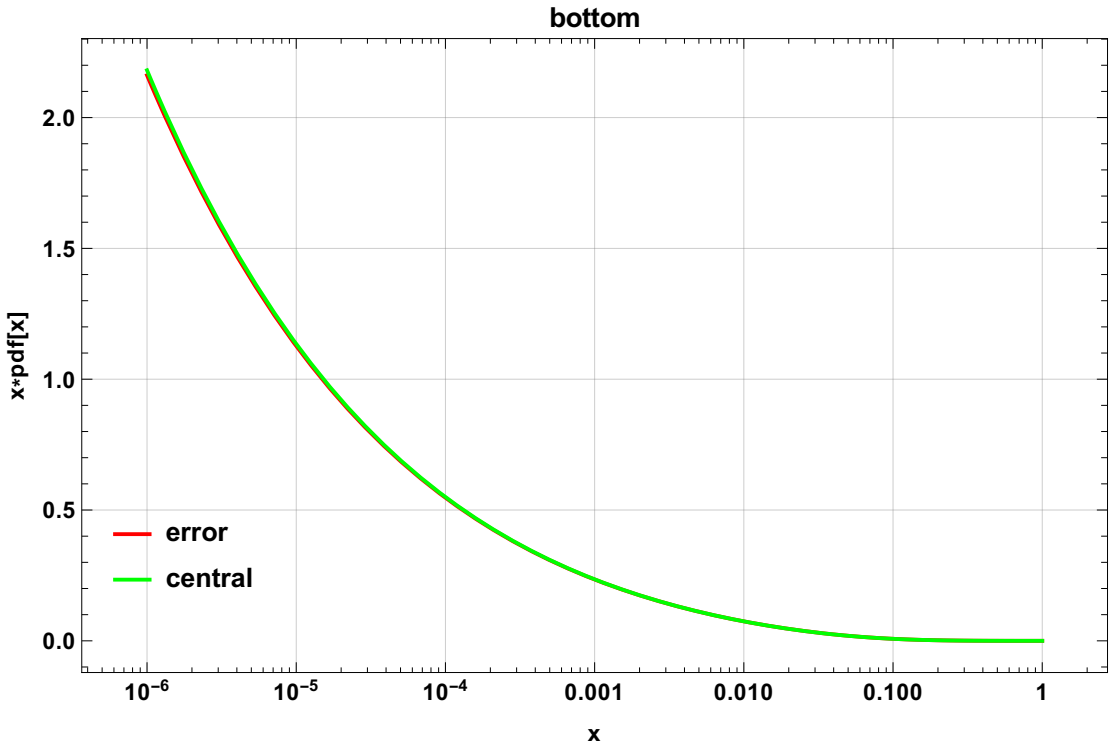

gluon
x*pdf[x]
120
100
80
60
40
20
0
error
central
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x


down
x*pdf[x]
5
4
3
2
1
0
error
central
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x


up
x*pdf[x]
x
error
central
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1


strange
x*pdf[x]
x
error
central
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1


charm
x*pdf[x]
4
3
2
1
0
error
central
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x


bottom
x*pdf[x]
2.0
1.5
1.0
0.5
0.0
error
central
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x

Example: Plotting Band Plots

Band plots can be created to compare any group of PDF sets.

In[59]:=
```
ct10
```

Out[59]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53}
```

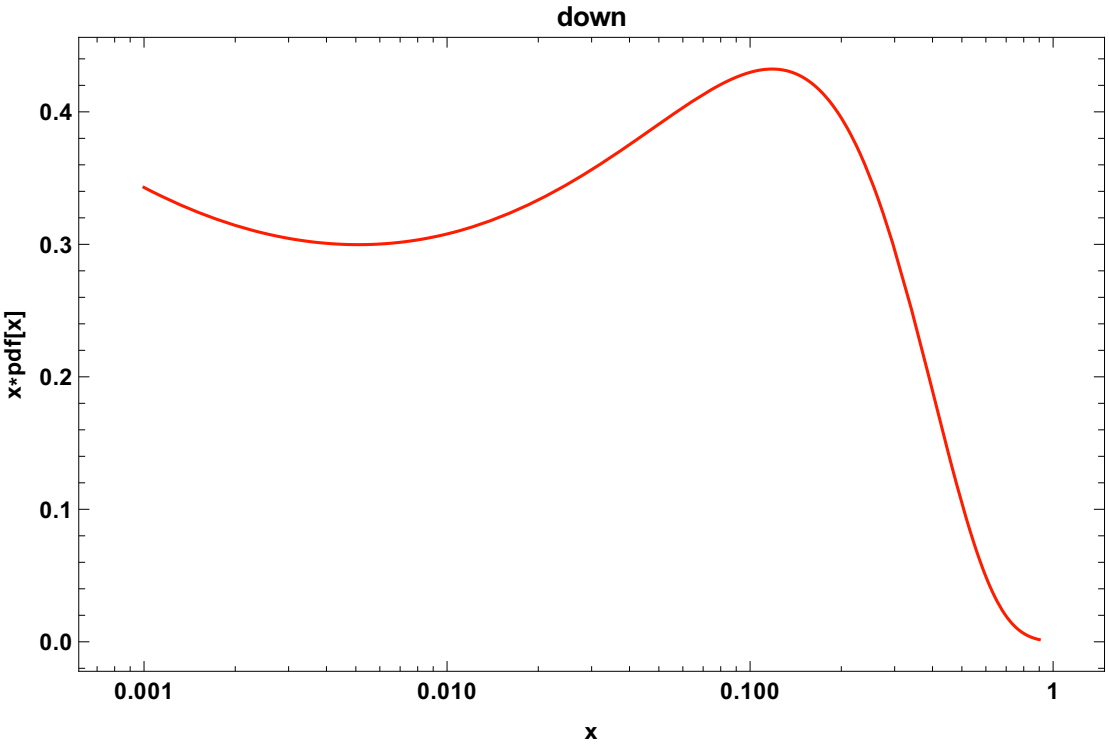
In[60]:=
```
length = ct10 // Length
```

Out[60]=
```
53
```

In[61]:=
```
(*The following function has been designed to
 create a LogLinear Plot and modified for appearance sake*)
LHAplot[iset_?IntegerQ, ipart_?IntegerQ, q_] :=
  LogLinearPlot[x (pdf[iset, ipart, x, q]), {x, 10^-3, 0.9},
   PlotStyle → Hue[iset / length],
   ImageSize → Large,
   FrameLabel → {"x", "x*pdf[x]"},
   Frame → True,
   BaseStyle → {FontWeight → "Bold", FontSize → 12},
   PlotLabel → pdfFlavor[ipart],
   PlotRange → All];
```

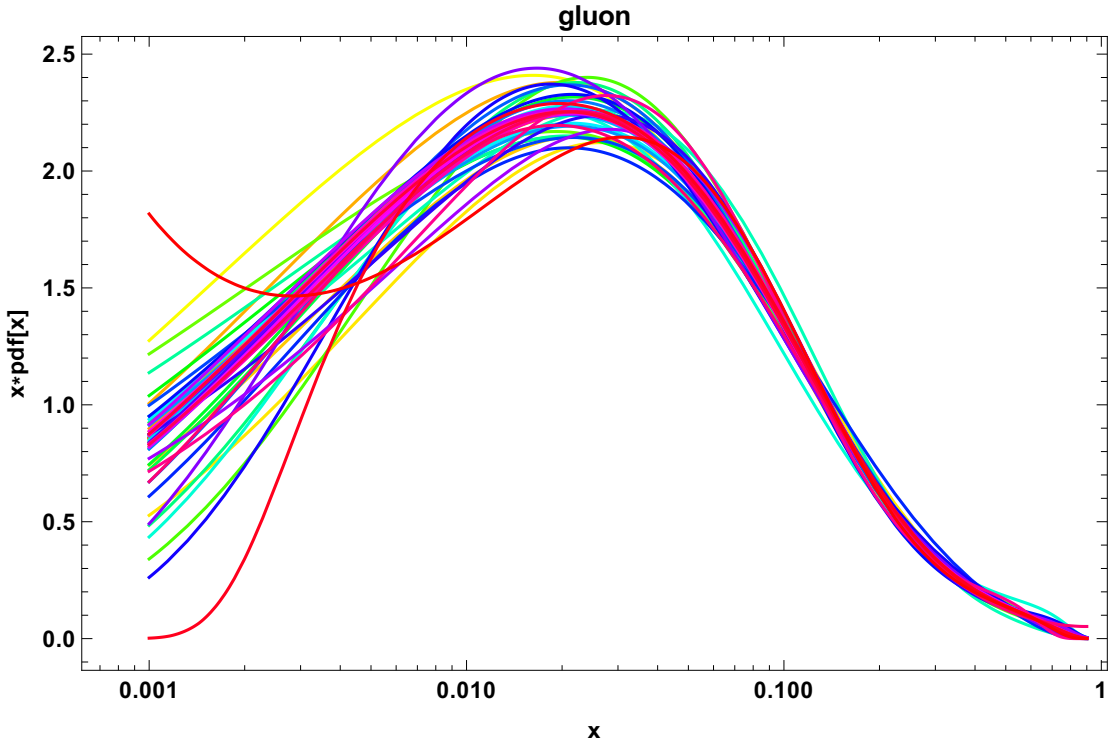
In[62]:=
```
LHAplot[1, 1, 1.3]
```

Out[62]=



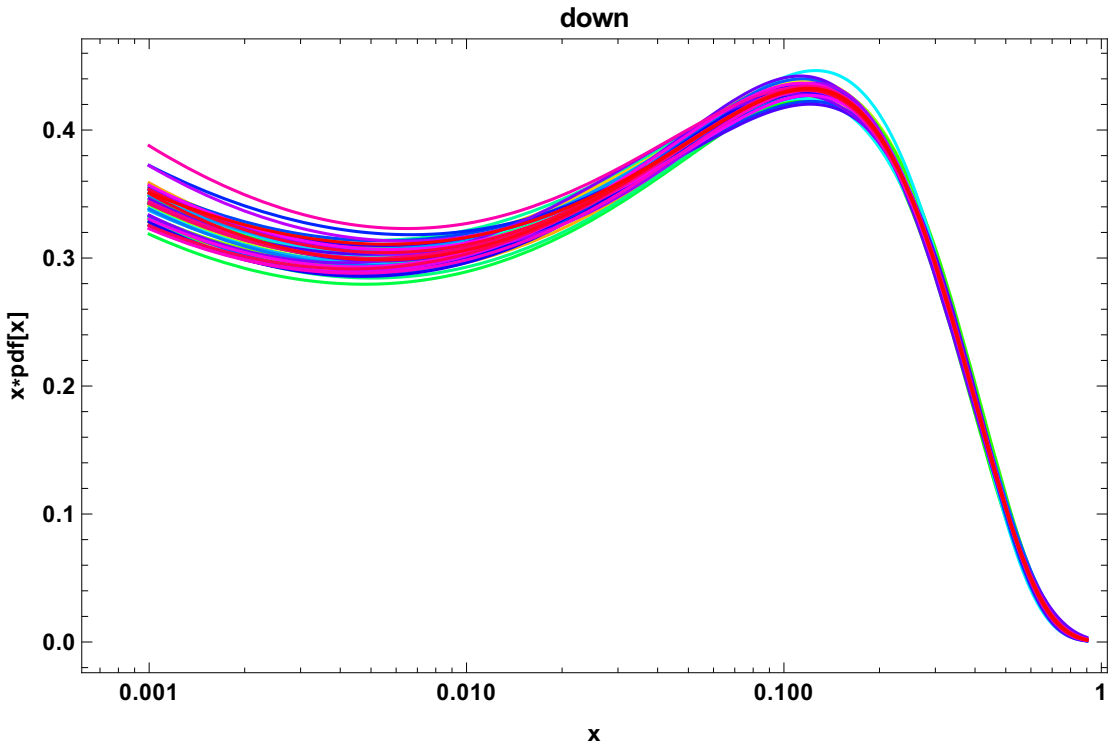
In[63]:=
```
ipart = 0; (* gluon *)
q0 = 1.3;
bandplot = Table[LHAplot[i, ipart, q0] , {i, ct10[[1]], length}];
Show[bandplot, PlotRange → All]
```

Out[66]=



In[67]:=
```
ipart = 1; (* down *)
q0 = 1.3;
bandplot = Table[LHAplot[i, ipart, q0] , {i, ct10[[1]], length}];
Show[bandplot, PlotRange → All]
```

Out[70]=



Example : Ratio Plots

This compares a value for the same initial variables across all the PDFs in a family

In[71]:=
```
q0 = 10.;
```

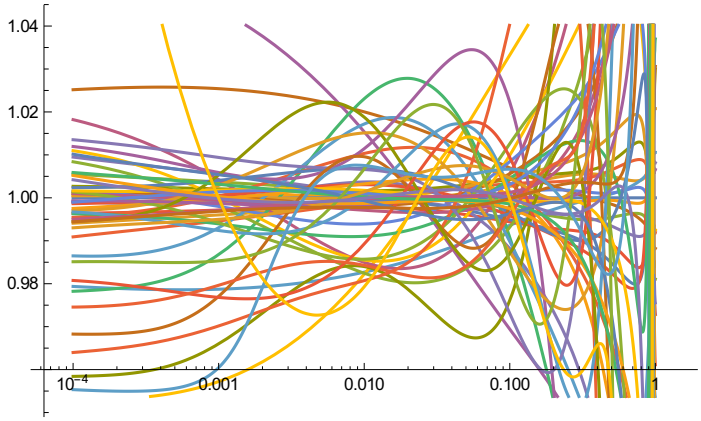
In[72]:=
```
pdfFunction[#, 21, 0.1, q0] & /@ ct10
```

Out[72]=
```
{11.2111, 11.2411, 11.1835, 11.2479, 11.1739, 11.2892, 11.1395, 11.584, 10.8682,
 11.2591, 11.1483, 11.2247, 11.2034, 11.2734, 11.1084, 11.1538, 11.2646, 11.1343,
 11.25, 11.0791, 11.2739, 11.0974, 11.3223, 11.4609, 10.9436, 11.1641, 11.3055,
 11.2178, 11.2101, 11.2612, 11.1734, 11.0699, 11.3524, 11.6641, 10.9092, 11.1899,
 11.2257, 11.2024, 11.2147, 11.0661, 11.3169, 11.2463, 11.1828, 11.2255,
 11.1947, 11.1695, 11.2627, 11.042, 11.1654, 11.2094, 11.209, 11.2859, 11.2088}
```

Here all the PDFs in the family are compared to the central value PDF

In[73]:=
```
ratio1 = LogLinearPlot[
 Table[pdfFunction[iset, 21, x, q0] / pdfFunction[1, 21, x, q0],
     {iset, 1, Length[ct10], 1}] //
 Evaluate, {x, 10.^-4, 1}]
```

Out[73]=



Using pdfGetXlist and pdfGetQlist

The x and Q grids can be directly read from the stored PDFs.

In[74]:=
```
iset = 1;
```

In[75]:=
```
pdfGetQlist[iset]
```

Out[75]=
```
{{1.3, 1.50159, 1.75516, 2.07811, 2.49494, 3.04086, 3.76712, 4.74999,
  6.23105, 8.37433, 11.5549, 16.4074, 24.0385, 36.4364, 57.3141, 93.8684,
  160.657, 288.438, 545.574, 1092.35, 2326.49, 5300.33, 12995., 34515.}}
```

In[76]:=
```
pdfGetXlist[iset]
```

Out[76]=
```
{{1. × 10^-8, 1.2414 × 10^-8, 1.54112 × 10^-8, 1.9132 × 10^-8, 2.37512 × 10^-8, 2.94855 × 10^-8,
  3.66043 × 10^-8, 4.54417 × 10^-8, 5.64127 × 10^-8, 7.00323 × 10^-8, 8.69398 × 10^-8,
  1.07929 × 10^-7, 1.33985 × 10^-7, 1.66332 × 10^-7, 2.06486 × 10^-7, 2.56334 × 10^-7,
  3.18214 × 10^-7, 3.95031 × 10^-7, 4.90389 × 10^-7, 6.09206 × 10^-7, 7.56249 × 10^-7,
  9.38778 × 10^-7, 1.16535 × 10^-6, 1.44661 × 10^-6, 1.79581 × 10^-6, 2.22938 × 10^-6,
  2.76764 × 10^-6, 3.43585 × 10^-6, 4.26537 × 10^-6, 5.29516 × 10^-6, 6.57355 × 10^-6,
  8.16054 × 10^-6, 0.0000101306, 0.0000125762, 0.0000156121, 0.0000193806,
  0.0000240586, 0.0000298652, 0.0000370727, 0.0000460185, 0.0000571214,
  0.0000709007, 0.0000880003, 0.000109218, 0.000135543, 0.0001682, 0.000208704,
  0.000258931, 0.000321196, 0.000398359, 0.000493945, 0.000612291, 0.000758725,
  0.000939771, 0.00116349, 0.00143942, 0.00177935, 0.00219741, 0.00271045,
  0.00333846, 0.00410484, 0.00503669, 0.00616486, 0.00752468, 0.00915177,
  0.0110878, 0.0133759, 0.0160562, 0.019169, 0.0227509, 0.0268334, 0.0314404,
  0.0365886, 0.0422866, 0.0485349, 0.0553271, 0.0626598, 0.0704985, 0.0788306,
  0.087621, 0.0968651, 0.106525, 0.116574, 0.126988, 0.137743, 0.148809,
  0.160178, 0.171817, 0.183716, 0.19585, 0.208205, 0.220764, 0.233517, 0.246449,
  0.259568, 0.272827, 0.286233, 0.299777, 0.313451, 0.327248, 0.341159, 0.355179,
  0.369295, 0.383515, 0.397826, 0.412222, 0.4267, 0.441256, 0.455884, 0.470576,
  0.485354, 0.50018, 0.515067, 0.530013, 0.545014, 0.560068, 0.575171, 0.590323,
  0.605521, 0.620762, 0.636045, 0.651369, 0.66673, 0.682126, 0.697558, 0.713035,
  0.728531, 0.744058, 0.759616, 0.775202, 0.790813, 0.806447, 0.822102, 0.837776,
  0.853464, 0.869163, 0.884866, 0.900563, 0.916236, 0.931854, 0.947357, 0.962579,
  0.977049, 0.989087, 0.995616, 0.997992, 0.998976, 0.999487, 0.999795, 1.}}
```

The pdfXmin function gives the minimum value of x for the set. pdfFunction can only reliably interpolate down to this value.

In[77]:=
```
pdfXmin[iset]
```

Out[77]=
```
1. × 10^-8
```

In[78]:=
```
pdfGetXlist[iset] // Min
```

Out[78]=
```
1. × 10^-8
```

Additional user functions

pdfGetInfo function

can be used to show what content from the info file has been read into memory

In[79]:=
```
? pdfGetInfo
```

pdfGetInfo[setNumber]: This function returns the information corresponding to set
setNumber read from the .info file or generated from the header of a
.pds file.

pdfGetInfo[setNumber, value]: This function accepts a string and returns the info corresponding to set
setNumber read from the .info file or generated from the header of a
.pds file for a specific *value*.

Example:
pdfGetInfo[setNumber, "Flavors"] will return the
quark flavor scheme for the info file if that information is available.

Note: If the user is unaware of what is present in the info file,
pdfGetInfo[*setNumber*] may still be used and displays the all values in the
info file.

In[80]:=
```
pdfGetInfo[1] // TableForm
```

Out[80]//TableForm=

SetDesc → 'PDF fits using the standard CTEQ PDF evolution but using the HOPPIT alph
Authors → H.-L.Lai, M.Guzzi, J. Huston, Z.Li, P.M.Nadolsky, J.Pumplin and C.-P.Yuan
Reference → arXiv
Format → lhagrid1
DataVersion → 2
NumMembers → 53
Flavors → {-5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 21}
ForcePositive → 1
OrderQCD → 1
FlavorScheme → fixed
NumFlavors → 5
ErrorType → hessian90
XMin → $\frac{1}{100\,000\,000}$
XMax → 1
QMin → 1.3
QMax → 100 000
MZ → 91.1876
MUp → 0
MDown → 0
MStrange → 0
MCharm → 1.3
MBottom → 4.75
MTop → 172
AlphaS_MZ → 0.117982
AlphaS_OrderQCD → 1
AlphaS_Type → ipol
AlphaS_Qs → {1.3, 1.50159, 1.75516, 2.07811, 2.49494, 3.04086, 3.76712, 4.74999, 6.23105
AlphaS_Vals → {0.396535, 0.359977, 0.328291, 0.300505, 0.275891, 0.253897, 0.234103, 0.
AlphaS_Lambda4 → 0.326
AlphaS_Lambda5 → 0.226

In[81]:=
```
pdfGetInfo[1, "Flavors"]
```

Out[81]=
```
{-5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 21}
```

AlphaS functions

In[82]:=
```
alpha = pdfGetInfo[1, "AlphaS_Vals"]
```

Out[82]=
```
{0.396535, 0.359977, 0.328291, 0.300505, 0.275891, 0.253897,
 0.234103, 0.216597, 0.200727, 0.18602, 0.172381, 0.159721,
 0.147963, 0.137042, 0.126895, 0.117468, 0.108708, 0.100573,
 0.0930171, 0.0860018, 0.0794917, 0.0734518, 0.0678503, 0.0626567}
```

In[83]:=
```
? pdfAlphaS
```

pdfAlphaS[setNumber, Q]:This function returns the value of α_S at hard scattering energy Q when this information is available in the .pds or .info file.

Warning: This function will print a text message and return a Null value if the α_S information is not available.

In[84]:=
```
qlist = pdfGetQlist[1](*retrieve qlist from the .dat file*)
```

Out[84]=
```
{{1.3, 1.50159, 1.75516, 2.07811, 2.49494, 3.04086, 3.76712, 4.74999,
  6.23105, 8.37433, 11.5549, 16.4074, 24.0385, 36.4364, 57.3141, 93.8684,
  160.657, 288.438, 545.574, 1092.35, 2326.49, 5300.33, 12995., 34515.}}
```

In[85]:=
```
pdfAlphaS[1, Flatten[qlist][[1]]]
```

```
Created pdfAlphaS for iSet = 1

1 has  1 sub-grid
```

Out[85]=
```
0.396535
```

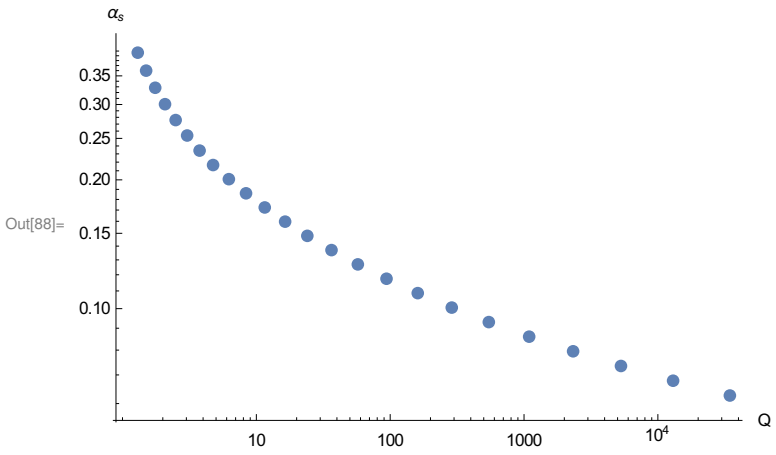
In[86]:=
```
(*For the Q values provided in the .dat file, this checks that the
 the AlphaS values at those values match those in the .info file*)
(*If they don't match, the alpha value in the info file
 is dropped. This is done for demonstration purposes*)
For[i = 1, i ≤ Length[Flatten[qlist]], i++,
 If[
  pdfAlphaS[1, Flatten[qlist][[i]]] ≠ alpha[[i]], alpha = Drop[alpha, {i}]
 ]
]
```

In[87]:=
```
tab1 = Transpose[{Flatten[qlist], alpha}]
```

Out[87]=
```
{{1.3, 0.396535}, {1.50159, 0.359977}, {1.75516, 0.328291}, {2.07811, 0.300505},
 {2.49494, 0.275891}, {3.04086, 0.253897}, {3.76712, 0.234103}, {4.74999, 0.216597},
 {6.23105, 0.200727}, {8.37433, 0.18602}, {11.5549, 0.172381}, {16.4074, 0.159721},
 {24.0385, 0.147963}, {36.4364, 0.137042}, {57.3141, 0.126895},
 {93.8684, 0.117468}, {160.657, 0.108708}, {288.438, 0.100573},
 {545.574, 0.0930171}, {1092.35, 0.0860018}, {2326.49, 0.0794917},
 {5300.33, 0.0734518}, {12995., 0.0678503}, {34515., 0.0626567}}
```

In[88]:=
```
p1 = ListLogLogPlot[tab1, AxesLabel → {"Q", "αs"}, PlotStyle → {PointSize[.02]}]
```

Out[88]=



pdfAlphaS function plotting

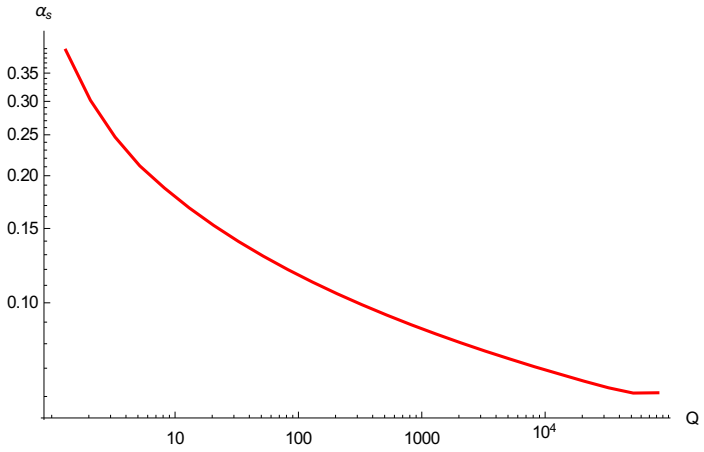
In[89]:=
```
(*This produces a data set of interpolated AlphaS values*)
tab2 = {};
e0 = Log[10, 1.3];
For[ee = e0, ee ≤ 5, ee += 1 / 5.,
 i = 10.^ee;
 tab2 = AppendTo[tab2, {i, pdfAlphaS[1, i]}]
]
```

InterpolatingFunction::dmval :
Input value {51753.9} lies outside the range of data in the interpolating function. Extrapolation will be used. ≫

InterpolatingFunction::dmval :
Input value {82024.5} lies outside the range of data in the interpolating function. Extrapolation will be used. ≫

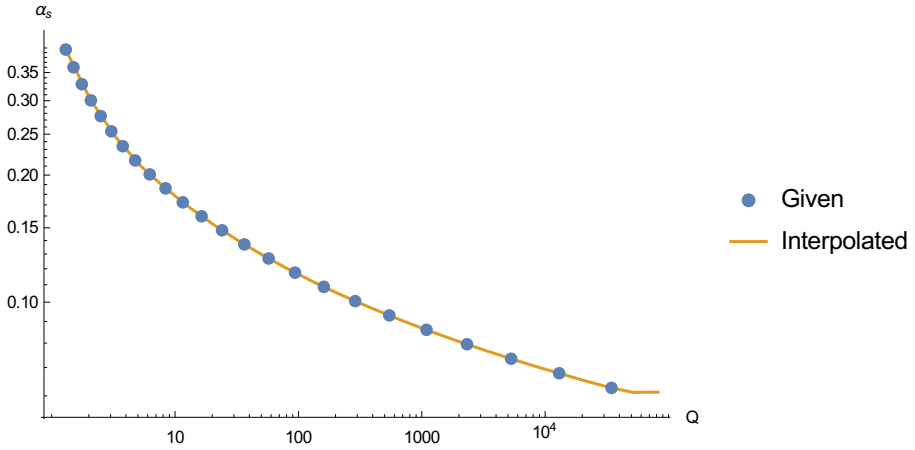
In[92]:=
```
p2 = ListLogLogPlot[tab2, PlotRange → Full,
  AxesLabel → {"Q", "αs"}, PlotStyle → {Red}, Joined → True]
```

Out[92]=



In[93]:=
```
ListLogLogPlot[{tab1, tab2}, PlotRange → Full,
 AxesLabel → {"Q", "αs"}, PlotLegends → {"Given", "Interpolated"},
 PlotStyle → {PointSize[.02], PointSize[.005]}, Joined → {False, True}]
```

Out[93]=



The noticeable kink in the 1/AlphaS plot when the b quark turns on is apparent in the plots below

In[94]:=
```
tab = {};
For[i = 1.3, i ≤ 50, i += .01,
 tab = AppendTo[tab, {i, 1/pdfAlphaS[1, i]}]
]
```

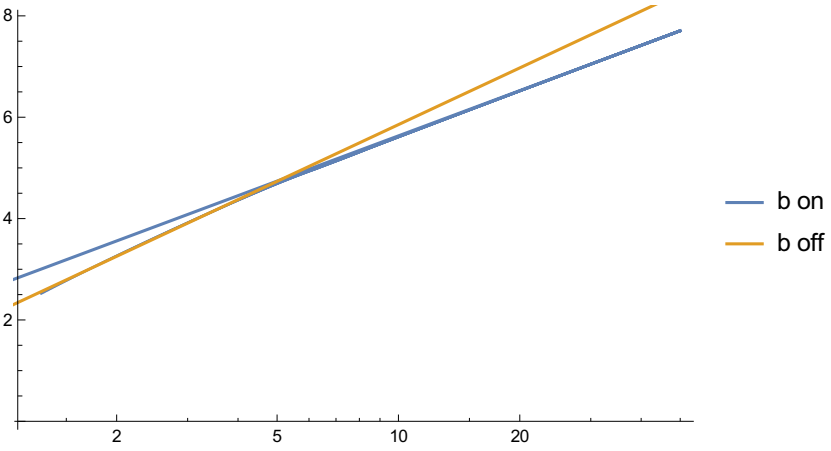
In[96]:=
```
p3 = ListLogLinearPlot[tab];
```

In[97]:=
```
f1 = Fit[Take[tab, -500], {1, Log[x]}, x];
f2 = Fit[Take[tab, 300], {1, Log[x]}, x];
```

In[99]:=
```
p4 = LogLinearPlot[{f1, f2}, {x, 1, 50}, PlotLegends → {"b on", "b off"}];
```

In[100]:=
```
Show[p3, p4]
```

Out[100]=



Example: Change to new PDF family

This demonstrates how to switch from one PDF family to another

In[101]:=
```
pdfReset[]
```

```
Default Mathematica interpolator will be used.

All internal variables have been reset.
```

In[102]:=
```
pdfSetXpower[1]
```

```
ManeParse cubic interpolation will be used.

The x-power of the interpolation is set to 1
```

In[103]:=
```
MSTW = pdfFamilyParseLHA[dirMSTW]
```

```
Successfully read
 /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/MSTW2008lo68cl/MSTW2008lo68cl.
   info.

Included 41 files in the PDF family.
```

Out[103]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41}
```

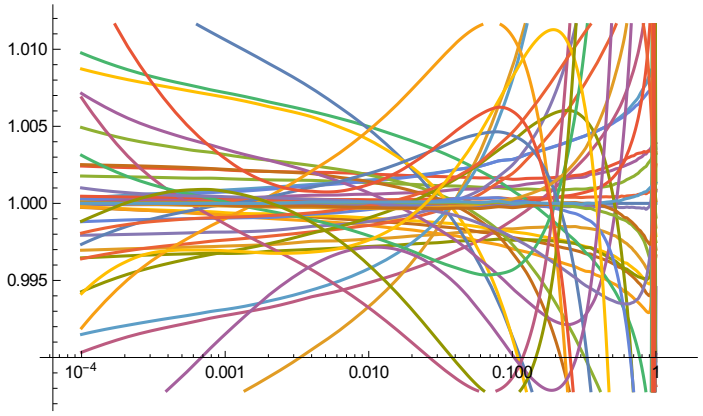
In[104]:=
```
pdfFunction[#, 0, 0.1, 10.] & /@ MSTW
```

Out[104]=
```
{10.873, 10.855, 10.883, 10.876, 10.871, 10.849, 10.904, 10.853, 10.885, 10.847, 10.89,
 10.904, 10.841, 10.861, 10.882, 10.779, 10.969, 10.844, 10.894, 10.9, 10.838,
 10.96, 10.778, 10.813, 10.917, 10.873, 10.875, 10.991, 10.75, 10.826, 10.921,
 10.876, 10.873, 10.923, 10.849, 10.87, 10.873, 10.962, 10.766, 10.736, 10.938}
```

In[105]:=
```
q0 = 10.;
length = MSTW // Length;
```

In[107]:=
```
ratio2 = LogLinearPlot[
 Table[pdfFunction[iset, 0, x, q0] / pdfFunction[1, 0, x, q0],
     {iset, 1, length, 1}] //
 Evaluate, {x, 10.^-4, 1}]
```

Out[107]=



In[108]:=
```
pdfGetInfo[1] // TableForm
```

Out[108]//TableForm=

SetDesc → "MSTW 2008 LO (68% C.L.). This set has 41 member PDFs. mem=0 => central
SetIndex → 21 000
Authors → A.D. Martin, W.J. Stirling, R.S. Thorne and G. Watt
Reference → arXiv
Format → lhagrid1
DataVersion → 2
NumMembers → 41
Particle → 2212
Flavors → {−5, −4, −3, −2, −1, 1, 2, 3, 4, 5, 21}
OrderQCD → 0
ForcePositive → 1
FlavorScheme → variable
NumFlavors → 5
ErrorType → hessian
XMin → $\frac{1}{1\,000\,000}$
XMax → 1
QMin → 1
QMax → 31 622.8
MZ → 91.1876
MUp → 0
MDown → 0
MStrange → 0
MCharm → 1.4
MBottom → 4.75
MTop → 1e+10
AlphaS_MZ → 0.139387
AlphaS_OrderQCD → 0
AlphaS_Type → ipol
AlphaS_Qs → {1., 1.11803, 1.22475, 1.4, 1.4, 1.58114, 1.78885, 2., 2.23607, 2.52982, 2.82
AlphaS_Vals → {0.68183, 0.614834, 0.569141, 0.513188, 0.513188, 0.473939, 0.439816, 0.4

In[109]:=
```
xlist = pdfGetXlist[1]
```

Out[109]=
```
{{1.×10^-6, 2.×10^-6, 4.×10^-6, 6.×10^-6, 8.×10^-6, 0.00001, 0.00002, 0.00004,
  0.00006, 0.00008, 0.0001, 0.0002, 0.0004, 0.0006, 0.0008, 0.001, 0.002,
  0.004, 0.006, 0.008, 0.01, 0.014, 0.02, 0.03, 0.04, 0.06, 0.08, 0.1,
  0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4,
  0.425, 0.45, 0.475, 0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65, 0.675, 0.7,
  0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95, 0.975, 1.},
 {1.×10^-6, 2.×10^-6, 4.×10^-6, 6.×10^-6, 8.×10^-6, 0.00001, 0.00002, 0.00004,
  0.00006, 0.00008, 0.0001, 0.0002, 0.0004, 0.0006, 0.0008, 0.001, 0.002,
  0.004, 0.006, 0.008, 0.01, 0.014, 0.02, 0.03, 0.04, 0.06, 0.08, 0.1,
  0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4,
  0.425, 0.45, 0.475, 0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65, 0.675, 0.7,
  0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95, 0.975, 1.},
 {1.×10^-6, 2.×10^-6, 4.×10^-6, 6.×10^-6, 8.×10^-6, 0.00001, 0.00002, 0.00004,
  0.00006, 0.00008, 0.0001, 0.0002, 0.0004, 0.0006, 0.0008, 0.001, 0.002,
  0.004, 0.006, 0.008, 0.01, 0.014, 0.02, 0.03, 0.04, 0.06, 0.08, 0.1,
  0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4,
  0.425, 0.45, 0.475, 0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65, 0.675, 0.7,
  0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95, 0.975, 1.}}
```

Note for MSTW grids, the .dat file is divided into multiple sections based on

Q value, thus extra inputs maybe required.

In[110]:=
```
? pdfNumQpartition
```

pdfNumQpartition[setNumber]: This function returns the number of Q grids in the PDF set *setNumber*.

In[111]:=
```
pdfNumQpartition[1]
```

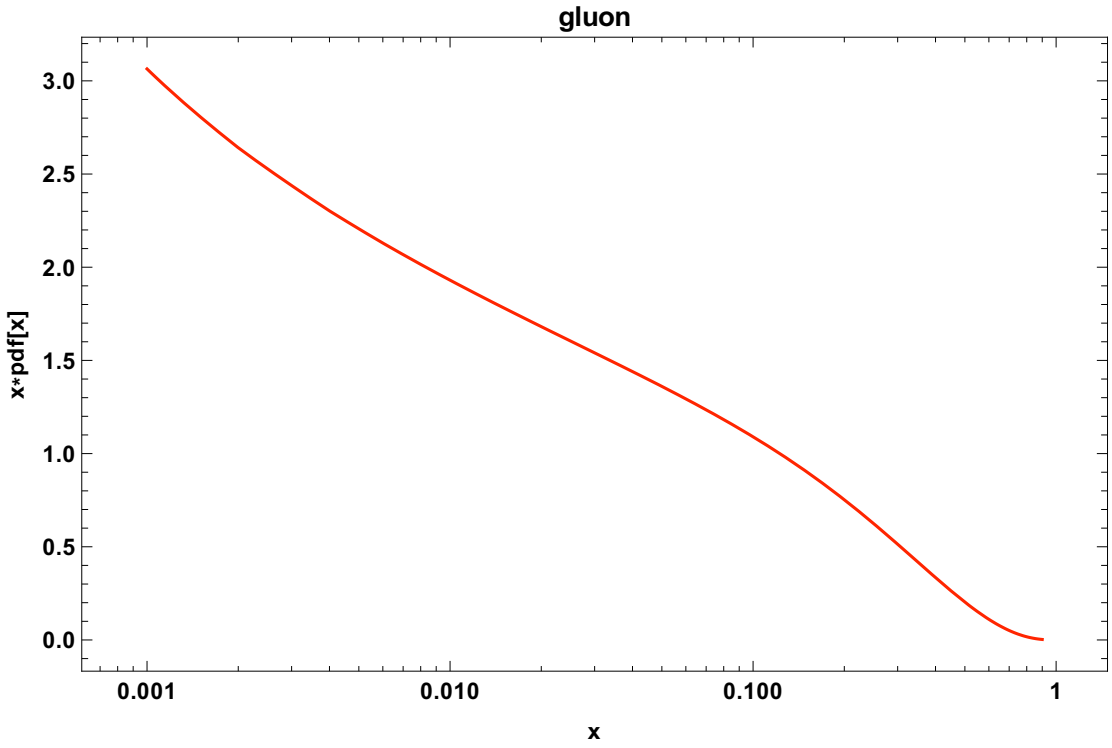
Out[111]= 3

In[112]:=
```
qlist = pdfGetQlist[1, 3]
```

Out[112]=
```
pdfGetQlist[1, 3]
```

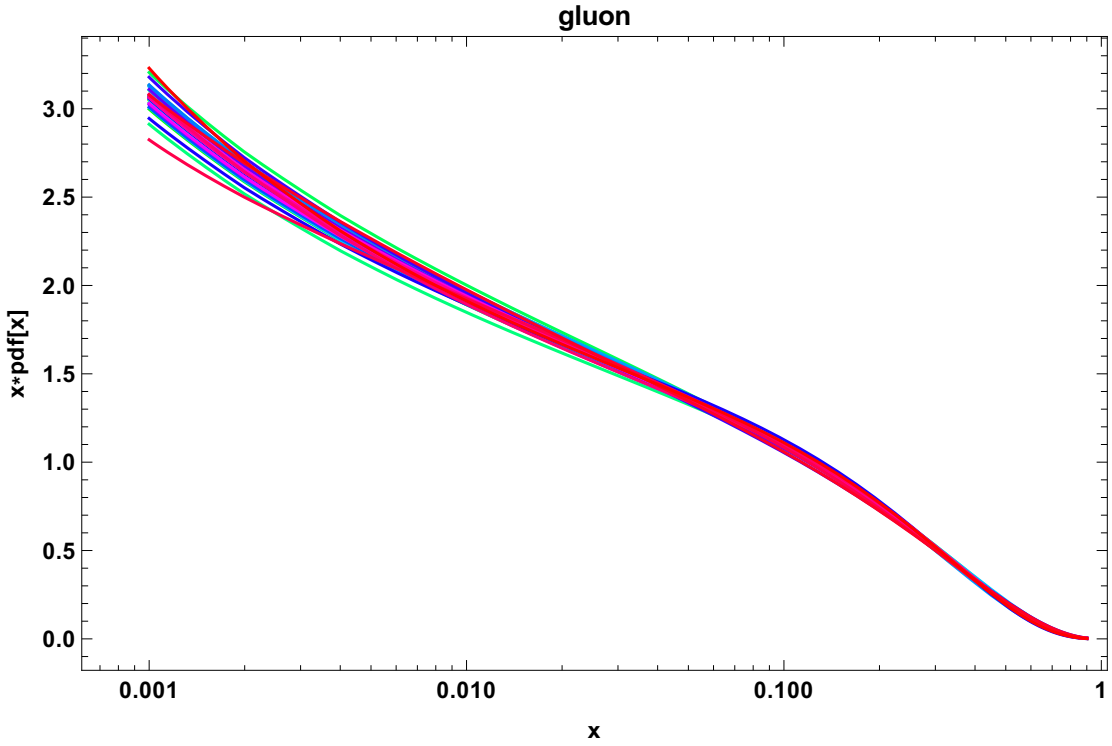
In[113]:=
```
LHAplot[1, 0, 1.3]
```

Out[113]=



In[114]:=
```
ipart = 0; (* GLUON *)
q0 = 1.3;
bandplot = Table[LHAplot[i, ipart, q0] , {i, MSTW[[1]], length}];
Show[bandplot, PlotRange → All]
```

Out[117]=



Working with multiple families at once.

In[118]:=
```
pdfReset[]
```

```
Default Mathematica interpolator will be used.
All internal variables have been reset.
```

In[119]:=
```
MSTW = pdfFamilyParseLHA[dirMSTW]
ct10 = pdfFamilyParseLHA[dirCT10]
nnpdf = pdfFamilyParseLHA[dirNNPDF]
```

```
Successfully read
 /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/MSTW2008lo68cl/MSTW2008lo68cl.
   info.
```

```
Included 41 files in the PDF family.
```

Out[119]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41}
```

```
Successfully read /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10.info.
```

```
Included 53 files in the PDF family.
```

Out[120]=
```
{42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94}
```

```
Successfully read
 /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/NNPDF30_nlo_as_0118/NNPDF30_nlo
   _as_0118.info.
```

```
Included 101 files in the PDF family.
```

Out[121]=
```
{95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147,
 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,
 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179,
 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195}
```

In[122]:=
```
total = pdfSetList // Length
```

Out[122]=
```
195
```

In[123]:=
```
Join[MSTW, ct10, nnpdf] // Length
```

Out[123]=
```
195
```

Check sum rule:

In[124]:=
```
Off[NIntegrate::izero]
Off[NIntegrate::ncvb]
q0 = 2.0;
iset0 = 1;(*MSTW*)
```

In[128]:=
```
tab = Table[ NIntegrate[ x * pdf[iset0, ipart, x, q0], {x, 0, 1}], {ipart, -5, 5, 1}];
Plus @@ tab
```

Out[129]=
```
0.998711
```

In[130]:=
```
flavorlist = {};
```

In[131]:=
```
For[i = -5, i ≤ 5, i++, AppendTo[flavorlist, pdfFlavor[i]];]
```

In[132]:=
```
flavorlist
```

Out[132]=
```
{bbar, cbar, sbar, ubar, dbar, gluon, down, up, strange, charm, bottom}
```

In[133]:=
```
{Range[-5, 5], flavorlist, Round[100 tab]} // Transpose // Grid[#, Frame → All] &
```

Out[133]=

-5	bbar	0
-4	cbar	0
-3	sbar	1
-2	ubar	3
-1	dbar	4
0	gluon	43
1	down	15
2	up	31
3	strange	1
4	charm	0
5	bottom	0

Check sum rule:

In[134]:=
```
Off[NIntegrate::izero]
Off[NIntegrate::ncvb]
q0 = 2.0;
iset0 = total - (Length[nnpdf]) + 1;(*nnpdf*)
```

In[138]:=
```
tab = Table[ NIntegrate[ x * pdf[iset0, ipart, x, q0], {x, 0, 1}], {ipart, -5, 5, 1}];
Plus @@ tab
```

Out[139]=
```
1.00187
```

In[140]:=
```
flavorlist = {};
```

In[141]:=
```
For[i = -5, i ≤ 5, i++, AppendTo[flavorlist, pdfFlavor[i]];]
```

In[142]:=
```
flavorlist
```

Out[142]=
```
{bbar, cbar, sbar, ubar, dbar, gluon, down, up, strange, charm, bottom}
```

In[143]:=
```
{Range[-5, 5], flavorlist, Round[100 tab]} // Transpose // Grid[#, Frame → All] &
```

Out[143]=

-5	bbar	0
-4	cbar	0
-3	sbar	1
-2	ubar	3
-1	dbar	4
0	gluon	43
1	down	15
2	up	31
3	strange	2
4	charm	0
5	bottom	0

PDS Files

Here we demonstrate the ability to handle PDS files in addition to LHA files

In[144]:=
```
pdfReset[]
```

```
Default Mathematica interpolator will be used.

All internal variables have been reset.
```

In[145]:=
```
ct10pds = pdfFamilyParseCTEQ[dirCT10pds]
```

```
Included 53 files in the PDF family.
```

Out[145]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53}
```

In[146]:=
```
CTEQ66 = pdfFamilyParseCTEQ[dirCTEQ66]
```

```
Included 45 files in the PDF family.
```

Out[146]=
```
{54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98}
```

In[147]:=
```
pdfFunction[1, 1, .1, 10]
```

Out[147]=
```
3.96968
```

In[148]:=
```
Clear[pdf]
pdf[iset_?IntegerQ, ipart_?IntegerQ, x_?NumericQ, q_?NumericQ] :=
 pdfFunction[iset, ipart, x, q]
```

In[150]:=
```
pdf[1, 1, .1, 10]
pdf[2, 1, 0.1, 10]

centralvalue = 1;

pdf[centralvalue, 1, 0.1, 10]
```

Out[150]=
```
3.96968
```

Out[151]=
```
3.95809
```

Out[153]=
```
3.96968
```

Check sum rule:

In[154]:=
```
Off[NIntegrate::izero]
Off[NIntegrate::ncvb]
q0 = 2.0;
iset0 = 1;
```

In[158]:=
```
tab = Table[ NIntegrate[ x pdf[iset0, ipart, x, q0], {x, 0, 1}], {ipart, -5, 5, 1}];
Plus @@ tab
```

Out[159]= 0.999837

In[160]:=
```
flavorlist = {};
```

In[161]:=
```
For[i = -5, i ≤ 5, i++, AppendTo[flavorlist, pdfFlavor[i]];]
```

In[162]:=
```
flavorlist
```

Out[162]= {bbar, cbar, sbar, ubar, dbar, gluon, down, up, strange, charm, bottom}

In[163]:=
```
{Range[-5, 5], flavorlist, Round[100 tab]} // Transpose // Grid[#, Frame → All] &
```

Out[163]=

-5	bbar	0
-4	cbar	0
-3	sbar	2
-2	ubar	3
-1	dbar	4
0	gluon	42
1	down	15
2	up	32
3	strange	2
4	charm	0
5	bottom	0

Check sum rule:

In[164]:=
```
Off[NIntegrate::izero]
Off[NIntegrate::ncvb]
q0 = 2.0;
iset0 = Length[pdfSetList] - Length[CTEQ66] + 1; (*CTEQ66*)
```

In[168]:=
```
tab = Table[ NIntegrate[ x pdf[iset0, ipart, x, q0], {x, 0, 1}], {ipart, -5, 5, 1}];
Plus @@ tab
```

Out[169]= 0.99984

In[170]:=
```
flavorlist = {};
```

In[171]:=
```
For[i = -5, i ≤ 5, i++, AppendTo[flavorlist, pdfFlavor[i]];]
```

In[172]:=
```
flavorlist
```

Out[172]= {bbar, cbar, sbar, ubar, dbar, gluon, down, up, strange, charm, bottom}

In[173]:=
```
{Range[-5, 5], flavorlist, Round[100 tab]} // Transpose // Grid[#, Frame → All] &
```

Out[173]=

-5	bbar	0
-4	cbar	0
-3	sbar	2
-2	ubar	3
-1	dbar	4
0	gluon	42
1	down	15
2	up	31
3	strange	2
4	charm	0
5	bottom	0

Compare PDS and LHA files

Here we compare the CT10 PDF family in both the LHA and PDS formats. As expected, they yield the same results.

In[174]:=
```
ct10LHA = pdfFamilyParseLHA[dirCT10]
```

```
Successfully read /users/olness/clark/trunk/ManeParse/Demo/PDF_Sets/LHA/CT10/CT10.info.

Included 53 files in the PDF family.
```

Out[174]=
```
{99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134,
 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151}
```

In[175]:=
```
q0 = 10;
centralvaluePDS = 1;
centralvalueLHA = Length[pdfSetList] - Length[ct10LHA] + 1;
```

In[178]:=
```
pdf[centralvalueLHA, 1, .1, 1.5]
```

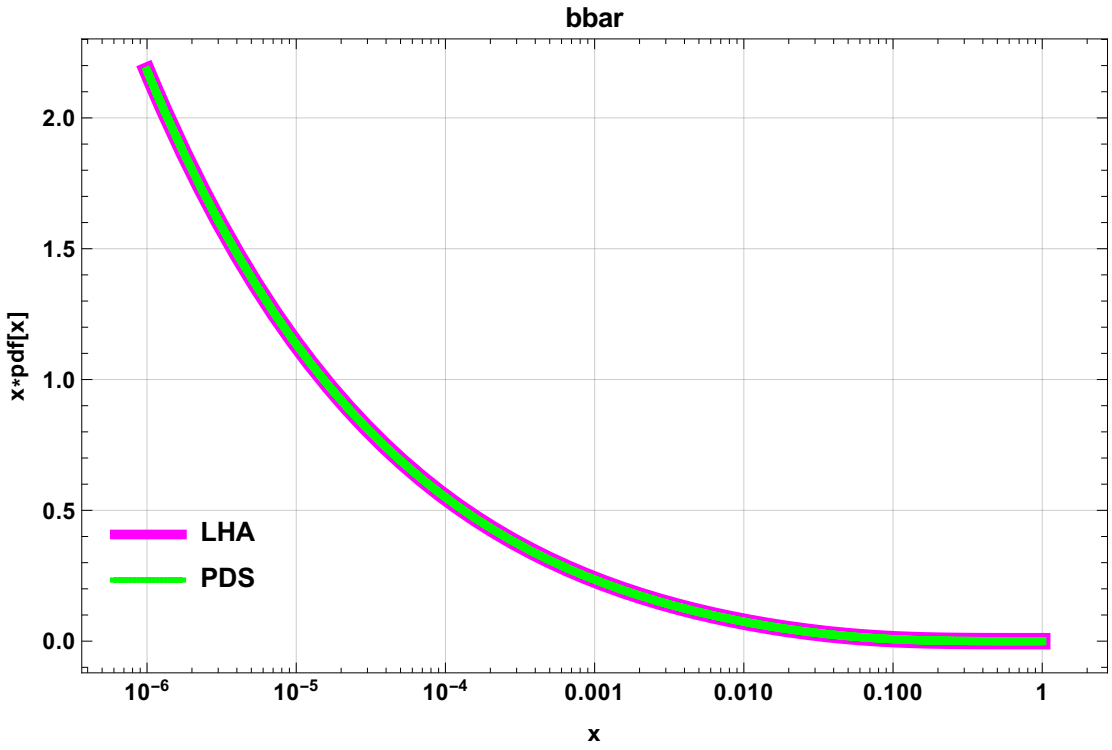
Out[178]=
```
4.28504
```

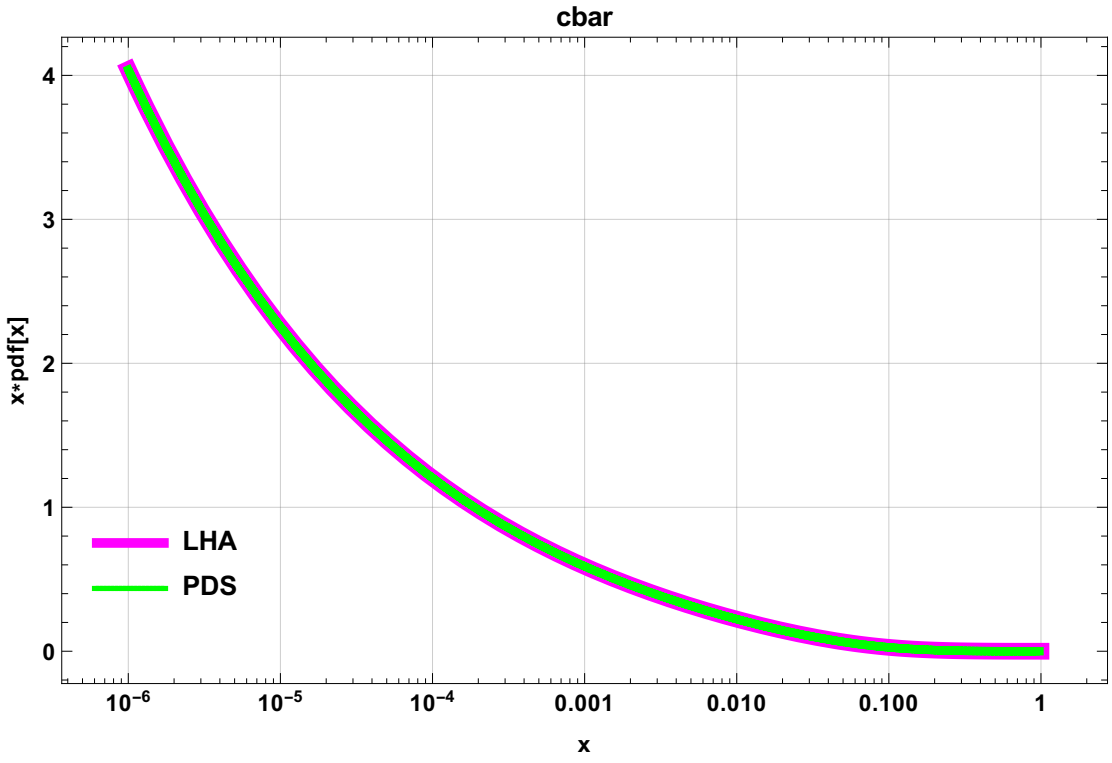
In[179]:=
```
pdf[centralvaluePDS, 1, .1, 1.5]
```

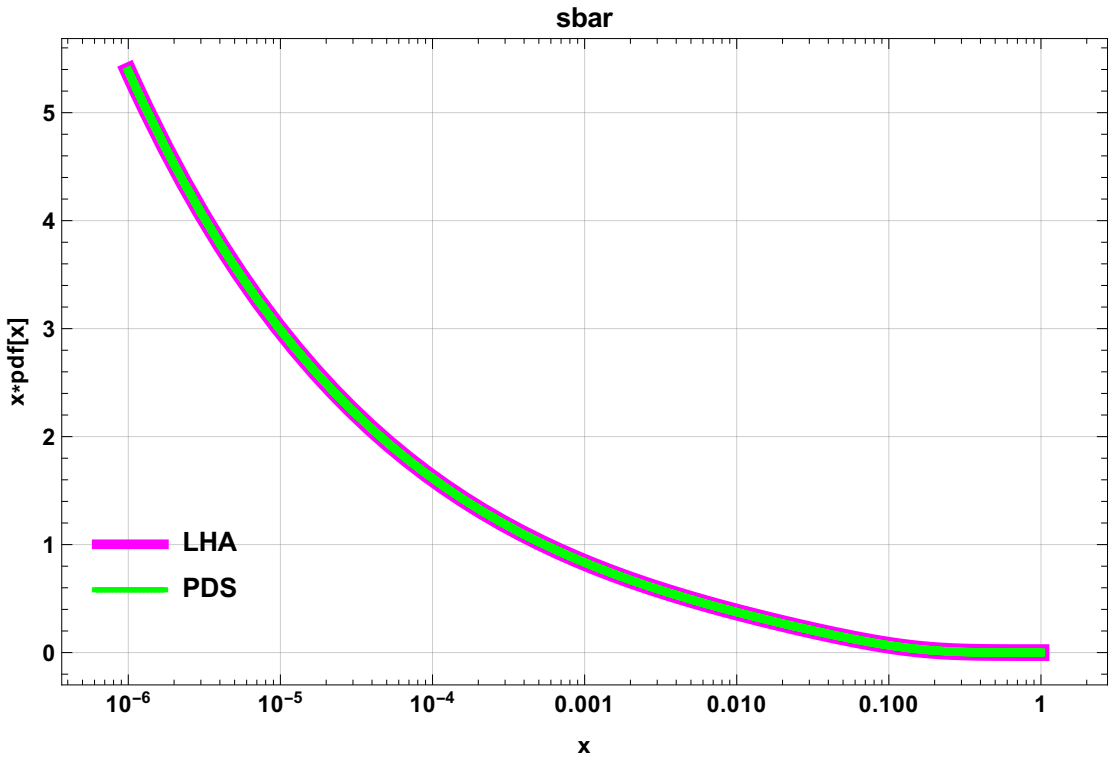
Out[179]=
```
4.28504
```

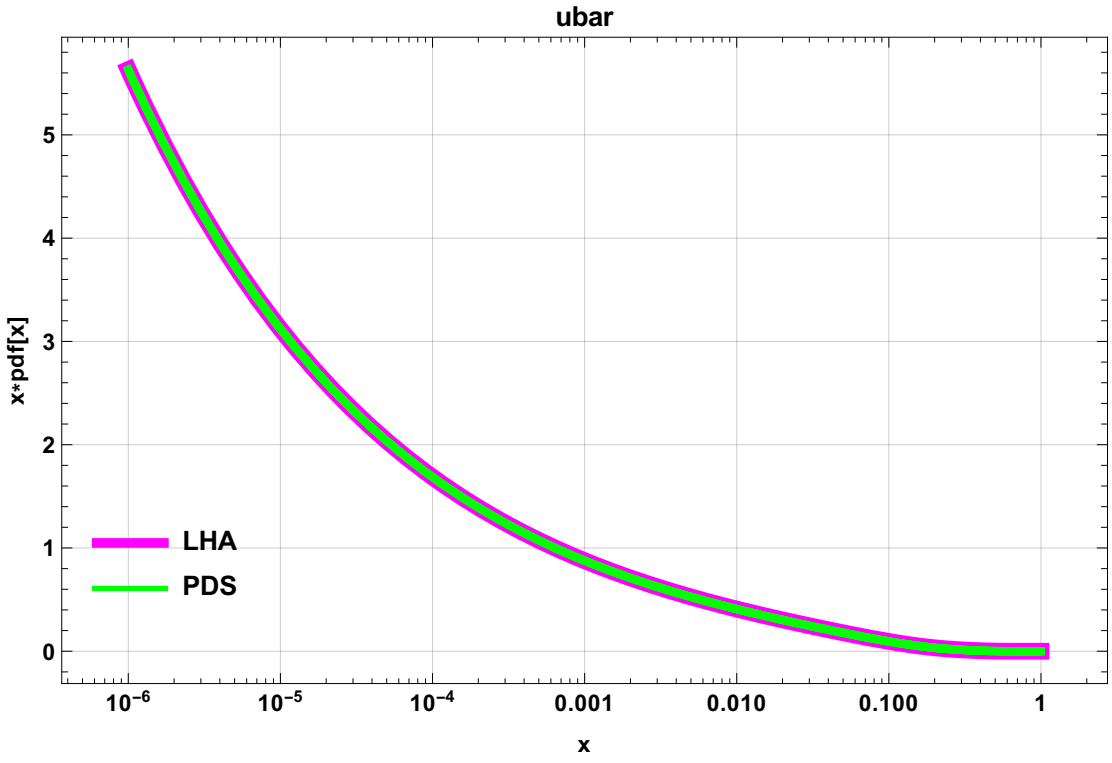
In[180]:=
```
xMin = Max[pdfXmin[centralvaluePDS], pdfXmin[centralvalueLHA]];
```

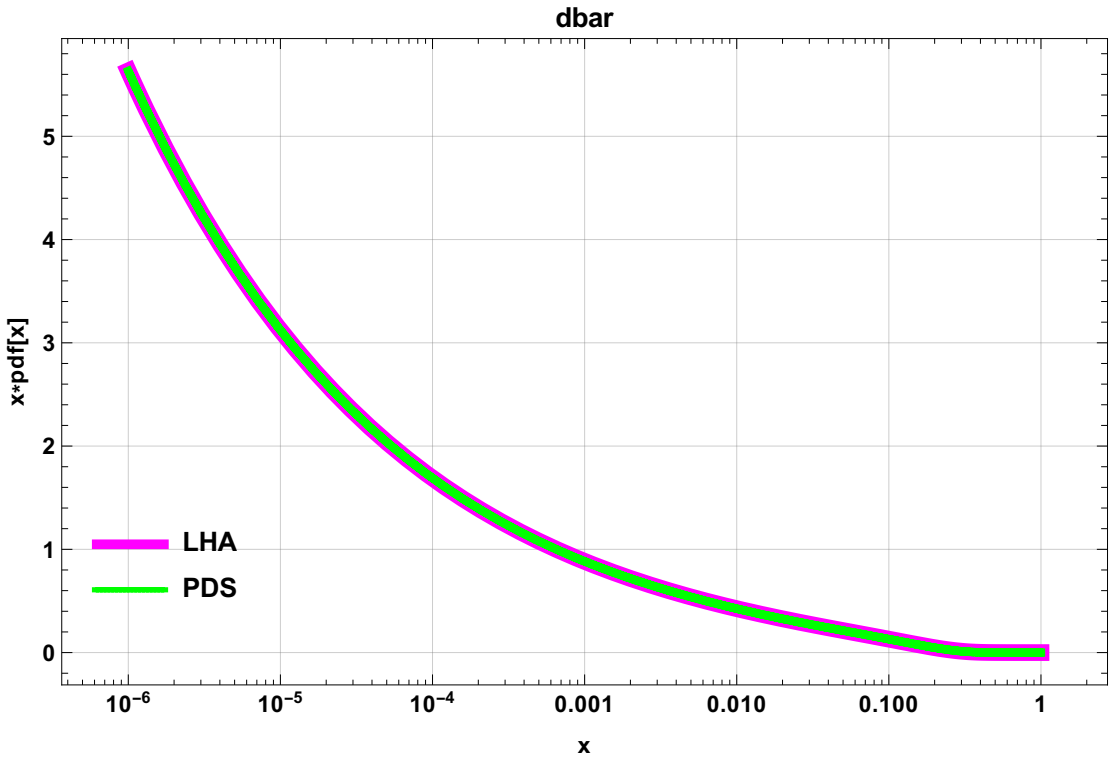
In[181]:=
```
For[i = -5, i ≤ 5, i++,
 LogLinearPlot[x {pdf[centralvaluePDS, i, x, q0], pdf[centralvalueLHA, i, x, q0]} //
    Evaluate , {x, xMin * 100, 1},
   PlotStyle → {Directive[Magenta, Thickness[0.016]],
     Directive[Green, Thickness[0.008], Dashing[.0]]},
   PlotLabel → pdfFlavor[i] ,
   FrameLabel → {"x", "x*pdf[x]"},
   ImageSize → Large,
   PlotRange → All,
   Frame → True,
   BaseStyle → {FontWeight → "Bold", FontSize → 12},
   GridLines → Automatic,
   PlotLegends → Placed[{"LHA", "PDS"}, {0.1, 0.18}]] // Print
]
```

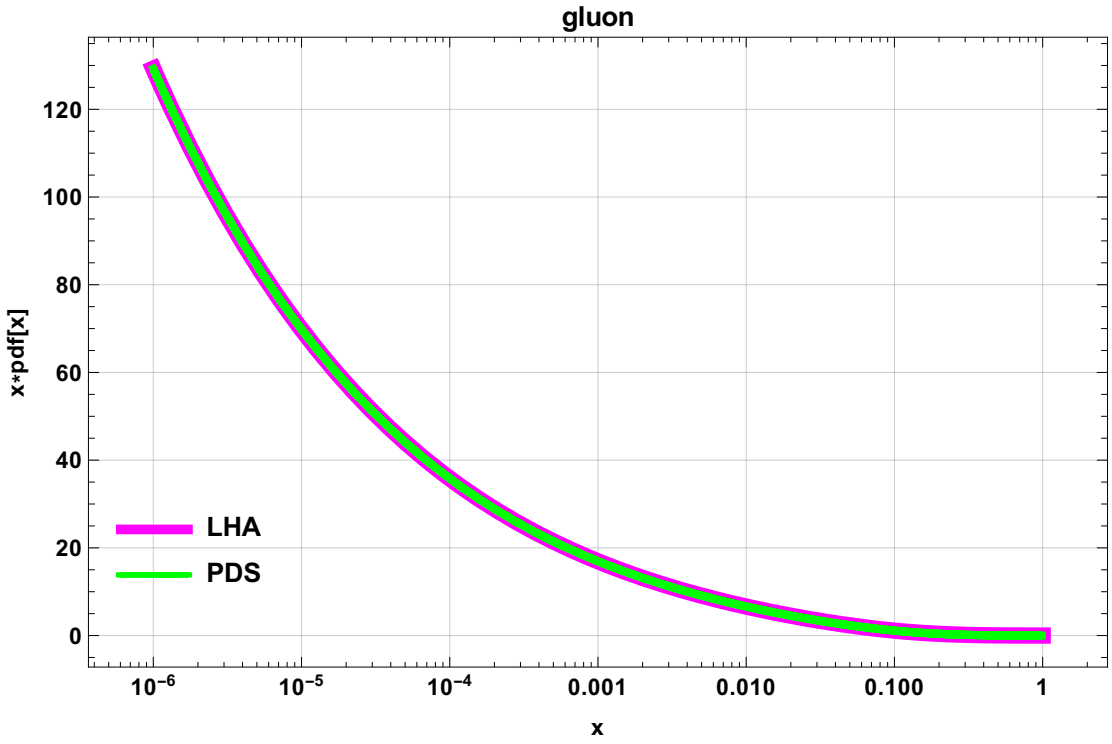


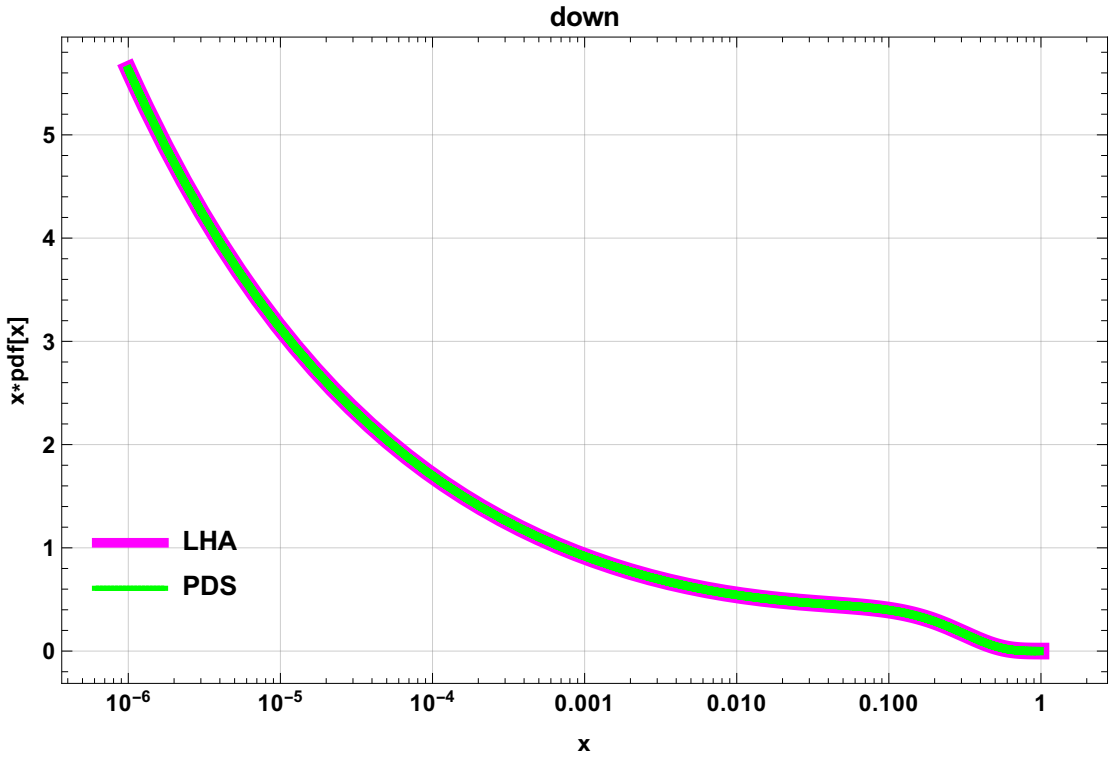

cbar
x*pdf[x]
x
LHA
PDS
0
1
2
3
4
10^-6
10^-5
10^-4
0.001
0.010
0.100
1

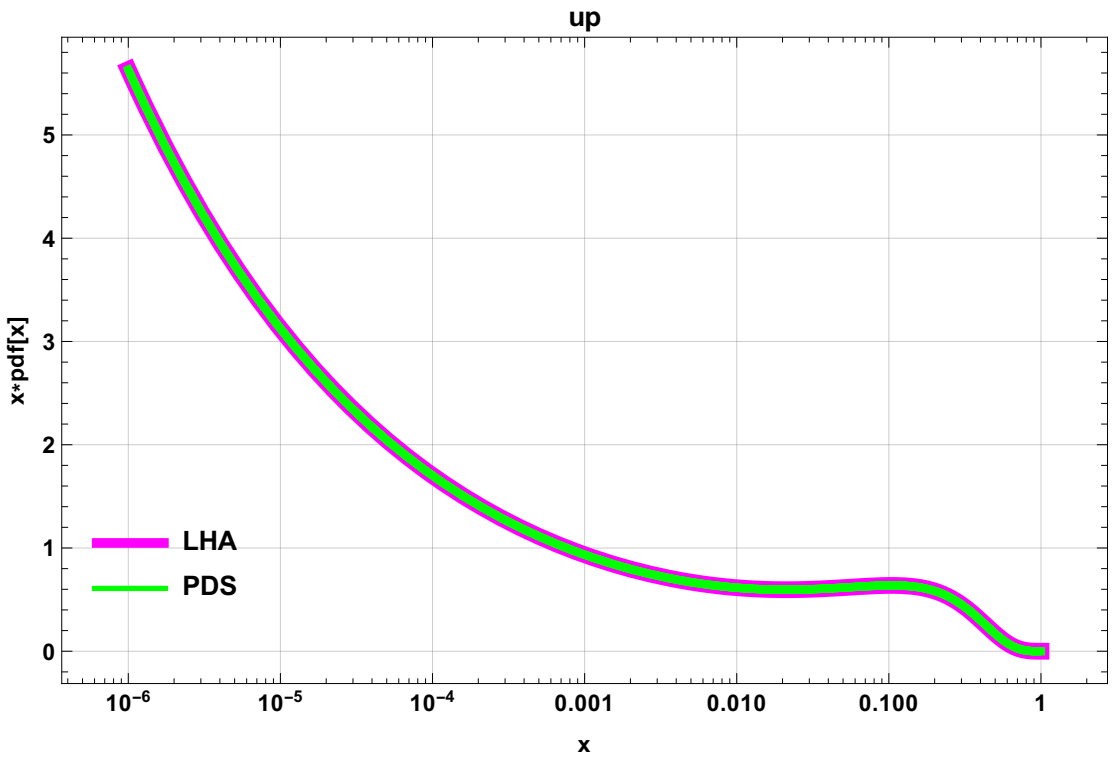

sbar
x*pdf[x]
x
LHA
PDS
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1

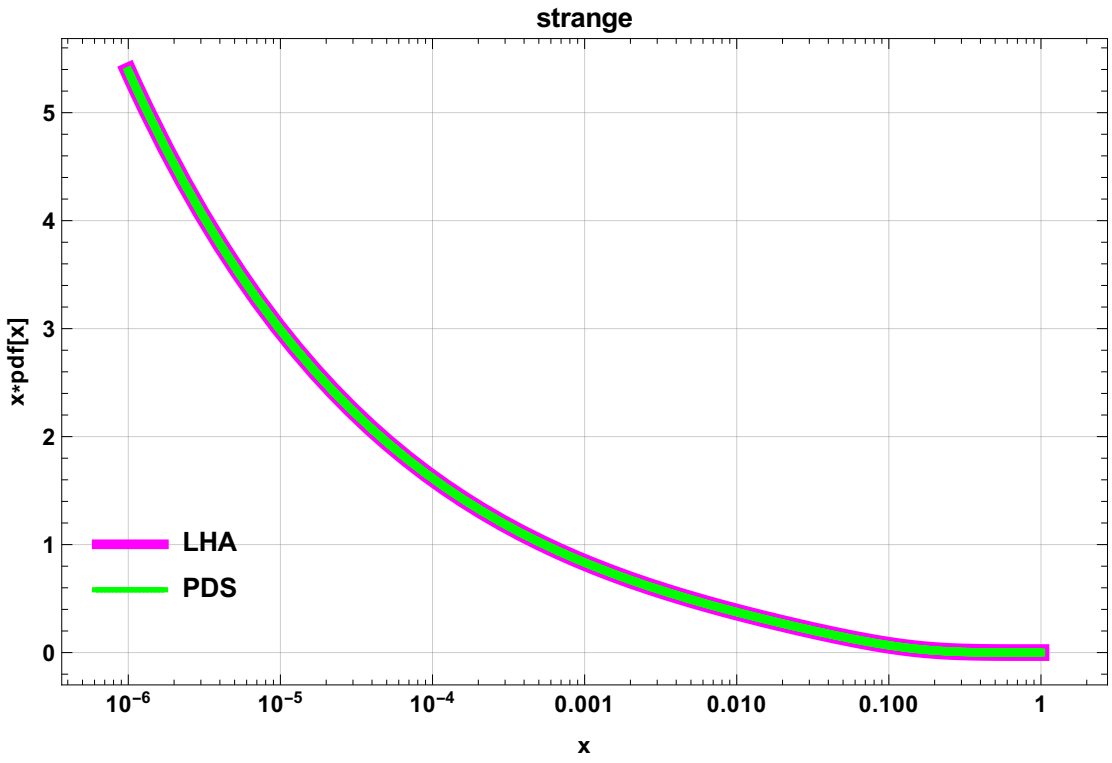

ubar
x*pdf[x]
x
LHA
PDS
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1

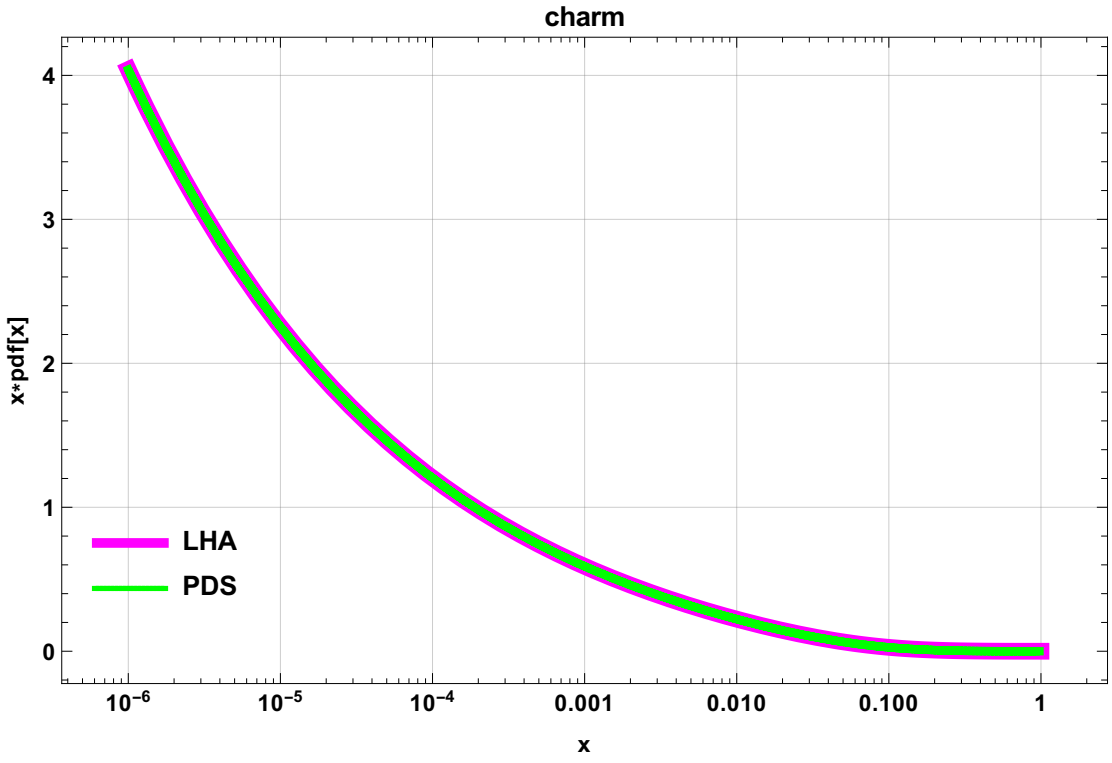

dbar
x*pdf[x]
x
LHA
PDS
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1

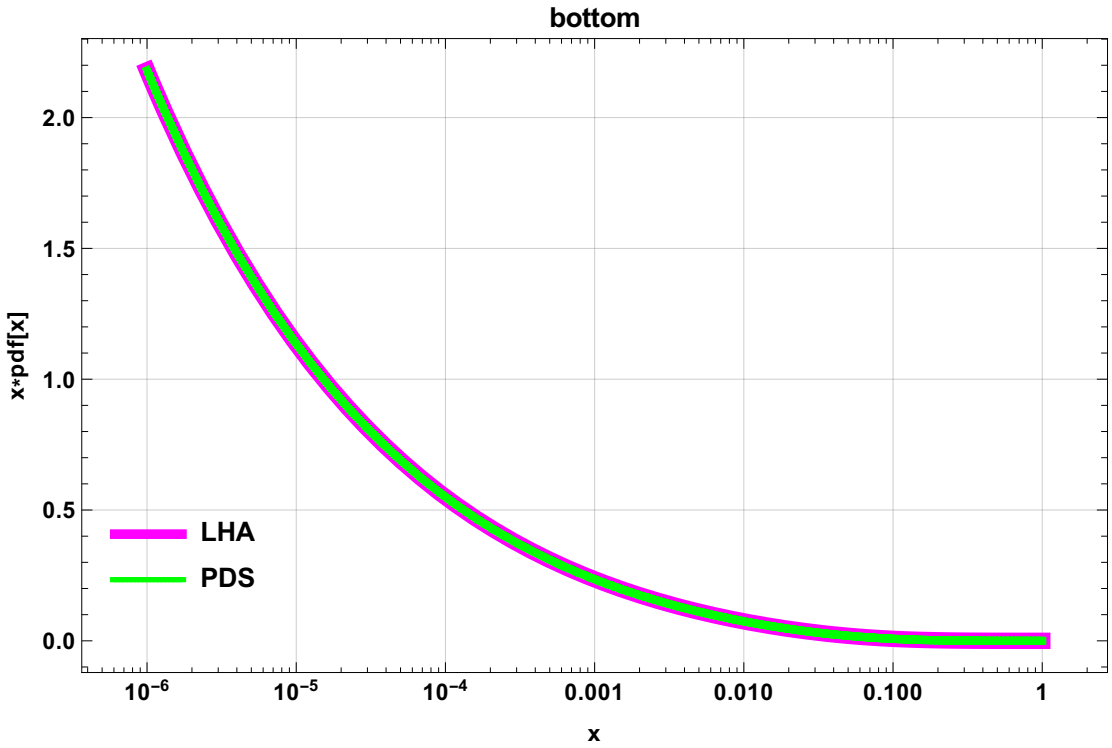

gluon
x*pdf[x]
120
100
80
60
40
20
0
LHA
PDS
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x


down
x*pdf[x]
5
4
3
2
1
0
LHA
PDS
10^-6
10^-5
10^-4
0.001
0.010
0.100
1
x


up
x*pdf[x]
x
LHA
PDS
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1


strange
x*pdf[x]
x
LHA
PDS
0
1
2
3
4
5
10^-6
10^-5
10^-4
0.001
0.010
0.100
1


charm
x*pdf[x]
x
LHA
PDS
4
3
2
1
0
10^-6
10^-5
10^-4
0.001
0.010
0.100
1


bottom
x*pdf[x]
x
LHA
PDS
2.0
1.5
1.0
0.5
0.0
10^-6
10^-5
10^-4
0.001
0.010
0.100
1