

# Example plots with the IPython notebook

So I just use the lhpdf library for loading the PDFs and plot them

## 1. Imports and defs

Here are couple of modules that I usually load. The important one if lhpdf of course. Note the path to the python lhpdf library has to be added via the `sys.path.append` command

```
In [1]: import os
import sys
import glob
import numpy as np
from math import *
import matplotlib.pyplot as plt
```

```
In [2]: %matplotlib inline
```

This is the key to import the lhpdf library

```
In [3]: os.system("python --version")
```

```
Out[3]: 0
Python 3.10.12
```

```
In [4]: import sys
sys.path.insert(0, '/home/vboxuser/LHAPDF/LHAPDF-6.5.3/wrappers/python/bui
#sys.path.insert(0, '/home/vboxuser/Software/deps/lhapdf/lib/python3.10/si
```

```
In [5]: import lhpdf
print(lhpdf.version())
```

6.5.3

```
In [6]: lhpdf.__file__
```

```
Out[6]: '/home/vboxuser/LHAPDF/LHAPDF-6.5.3/wrappers/python/build/lhapdf/lhapd
f.so'
```

```
In [7]: print(lhpdf.paths())

['/home/vboxuser/Software/deps/lhapdf/share/LHAPDF']
```

```
In [8]: #lhpdf.pathsAppend('/home/olness/Dropbox/PYTHON/LHAPDF/xFitterPDF/')
#lhpdf.pathsAppend('/home/olness/Dropbox/PYTHON/LHAPDF/xFitterPDF/')
os.listdir('./PDFDIR/')
```

```
Out[8]: ['NNPDF31_nnlo_as_0118',
        'nCTEQ15FullNuc_208_82',
        'lhpdf.conf',
        'pdfsets.index',
        'nCTEQ15FullNuc_56_26',
        'NNPDF40_nnlo_as_01180',
        'MSHT20an3lo_as118',
        'ABMP16als118_5_nnlo',
        'HERAPDF20_NNLO_VAR',
        'rsFree',
        'CT18ANLLO',
        'rsFixed',
        'nCTEQ15FullNuc_1_1',
        'NNPDF30_nlo_as_0118',
        'nCTEQ15FullNuc_184_74',
        'MSHT20nnlo_as118',
        'PDF4LHC21_40',
        'CT18NNLO',
        'HERAPDF20_NNLO_EIG']
```

This is the key to import the lhpdf library

```
#lhpdf.pathsAppend('/home/olness/Dropbox/PYTHON/LHAPDF/xFitterPDF/')
#lhpdf.pathsAppend('/home/olness/Dropbox/PYTHON/LHAPDF/xFitterPDF/') os.listdir('./
PDFDIR')#lhpdf.pathsAppend('/home/olness/Dropbox/PYTHON/LHAPDF/xFitterPDF/')
#lhpdf.pathsAppend('/home/olness/Dropbox/PYTHON/LHAPDF/xFitterPDF/') os.listdir('/home/
vboxuser/Software/deps/lhpdf/share/LHAPDF')cwd = os.getcwd() cwd
```

```
In [9]: lhpdf.pathsAppend('./LHAPDF/')
```

```
print(lhpdf.paths())os.listdir('./PDFDIR')print(lhpdf.paths())os.listdir('/usr/local/share/
LHAPDF')print("Path at terminal when executing this file") print(os.getcwd() + "\n")
```

```
In [10]: print(glob.glob("*"))
```

```
['Read_LHAPDF.pdf', 'step4.png', 'step2.png', 'step1.png', 'Read_LHAPDF.ip
ynb', 'PDFDIR', 'bottom05.pdf', 'Read_LHAPDF_Virgile_Mahaut.ipynb', 'Read_
LHAPDF_Eric_Kolbusz.ipynb']
```

```
In [11]: print(lhpdf.paths())
```

```
['/home/vboxuser/Software/deps/lhpdf/share/LHAPDF', './LHAPDF/', '/home/v
boxuser/Software/deps/lhpdf/share/LHAPDF']
```

## 1.1 Functions and PDFs

Let's load all the PDFs sets that we need and in particular the nCTEQ15 full lead, iron and proton (deuterium).

```
In [12]: proton_ncteq = lhpdf.getPDFSet('nCTEQ15FullNuc_1_1')
```

```
In [13]: xf0 = lhpdf.getPDFSet('nCTEQ15FullNuc_1_1')
         xf0=xf0.mkPDFs()
         xf0[0].xfxQ(2,0.1,10.)
```

```
LHAPDF 6.5.3 loading all 1 PDFs in set nCTEQ15FullNuc_1_1
nCTEQ15FullNuc_1_1, version 1; 1 PDF members
```

```
Out[13]: 0.6504072208540947
```

In [ ]:

```
In [14]: xf0 = lhapdf.getPDFSet('rsFixed')
         xf0=xf0.mkPDFs()
         xf0[0].xfxQ(2,0.1,10.)
```

LHAPDF 6.5.3 loading all 1 PDFs in set rsFixed  
rsFixed, version 1; 1 PDF members

Out[14]: 0.66114855119381

```
In [15]: xf1 = lhapdf.getPDFSet('rsFree')
         xf1=xf1.mkPDFs()
         xf1[0].xfxQ(2,0.1,10.)
```

LHAPDF 6.5.3 loading all 1 PDFs in set rsFree  
rsFree, version 1; 1 PDF members

Out[15]: 0.6627243695017107

```
In [16]: xf2 = lhapdf.getPDFSet('nCTEQ15FullNuc_1_1')
         xf2=xf2.mkPDFs()
         xf2[0].xfxQ(2,0.1,10.)
```

LHAPDF 6.5.3 loading all 1 PDFs in set nCTEQ15FullNuc\_1\_1  
nCTEQ15FullNuc\_1\_1, version 1; 1 PDF members

Out[16]: 0.6504072208540947

```
In [17]: xf3 = lhapdf.getPDFSet('NNPDF30_nlo_as_0118')
         xf3=xf3.mkPDFs()
         xf3[0].xfxQ(2,0.1,10.)
```

LHAPDF 6.5.3 loading all 101 PDFs in set NNPDF30\_nlo\_as\_0118  
NNPDF30\_nlo\_as\_0118, version 2; 101 PDF members

Out[17]: 0.6200919115217465

```
In [18]: xf4 = lhapdf.getPDFSet('CT18NNLO')
         xf4=xf4.mkPDFs()
         xf4[0].xfxQ(2,0.1,10.)
```

LHAPDF 6.5.3 loading all 59 PDFs in set CT18NNLO  
CT18NNLO, version 1; 59 PDF members

Out[18]: 0.6372213241191952

```
In [19]: xvalues = np.logspace(-3,0,100)
         qvalues = np.logspace(0,2,500)
```

In [ ]:

In [ ]:

## Make some plots

In [20]: %matplotlib inline

```
In [21]: istr=3;
         ic=4;
```

```

ib=5;
x0=0.05;
sFix=[xf0[0].xfxQ(istr,x0,qq) for qq in qvalues]
sFree=[xf1[0].xfxQ(istr,x0,qq) for qq in qvalues]

plt.plot(qvalues,sFix,label="$r_{s \, , \, \mathrm{fixed}}$")
plt.plot(qvalues,sFree,label="$r_{s \, , \, \mathrm{free}}$")
plt.xscale("log")

plt.ylim((-0.002,0.20))
plt.xlim((1,500))

plt.xlabel('$Q$ [GeV]',fontsize=18)
plt.ylabel('$xf(x,Q)$ ($x={0:.2f}$)'.format(x0),fontsize=18)
plt.legend(fontsize=10)

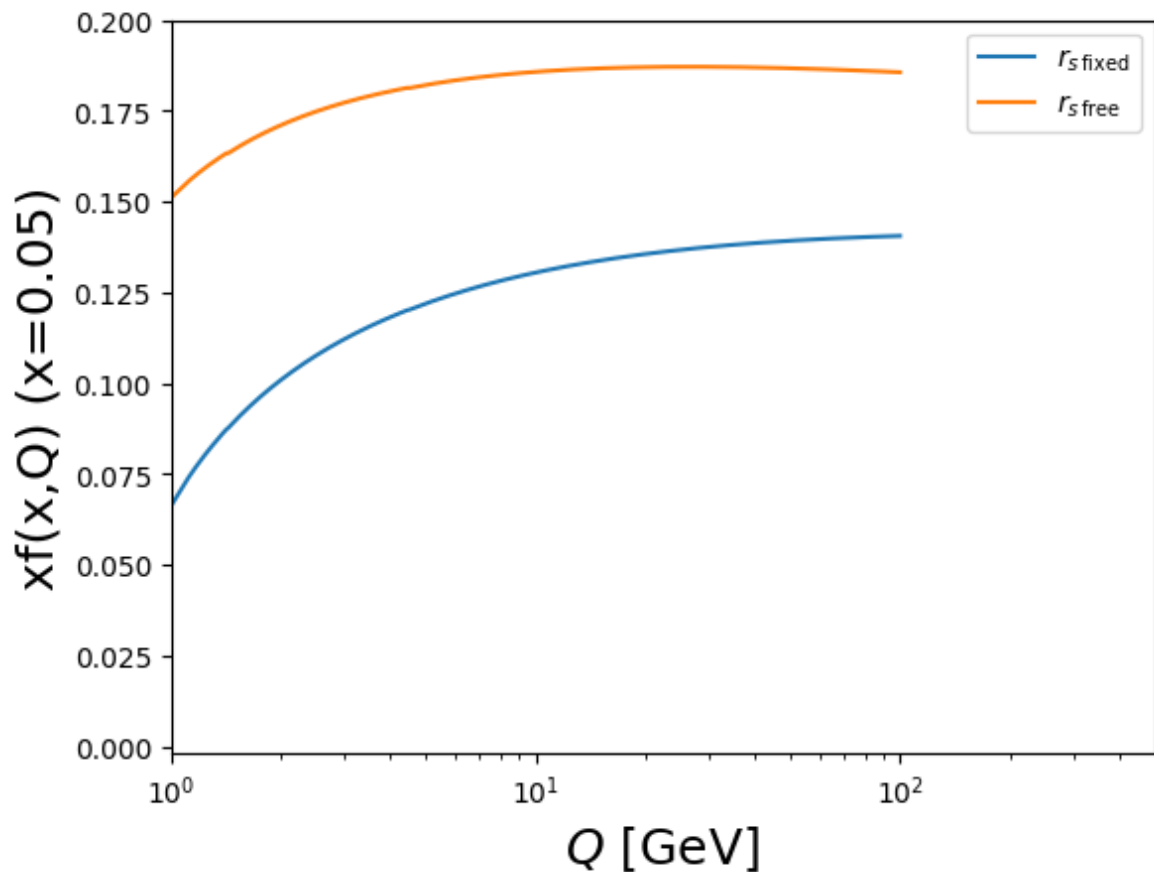
fig = plt.gcf()

#fig.show()

# show(block=False)

fig.savefig('bottom05.pdf')

```



```

In [22]: iglu=0;
istr=3;
ic=4;
ib=5;
x0=0.05;
sFix=[xf0[0].xfxQ(iglu,x0,qq) for qq in qvalues]
sFree=[xf1[0].xfxQ(iglu,x0,qq) for qq in qvalues]

```

```

plt.plot(qvalues,sFix,label="$r_{s \, , \, \mathrm{fixed}}$")
plt.plot(qvalues,sFree,label="$r_{s \, , \, \mathrm{free}}$")
plt.xscale("log")

plt.ylim((-0.002,3.0))
plt.xlim((1,500))

plt.xlabel('$Q$ [GeV]',fontsize=18)
plt.ylabel('$xf_g(x,Q)$ ($x={0:.2f}$)'.format(x0),fontsize=18)
plt.legend(fontsize=10)

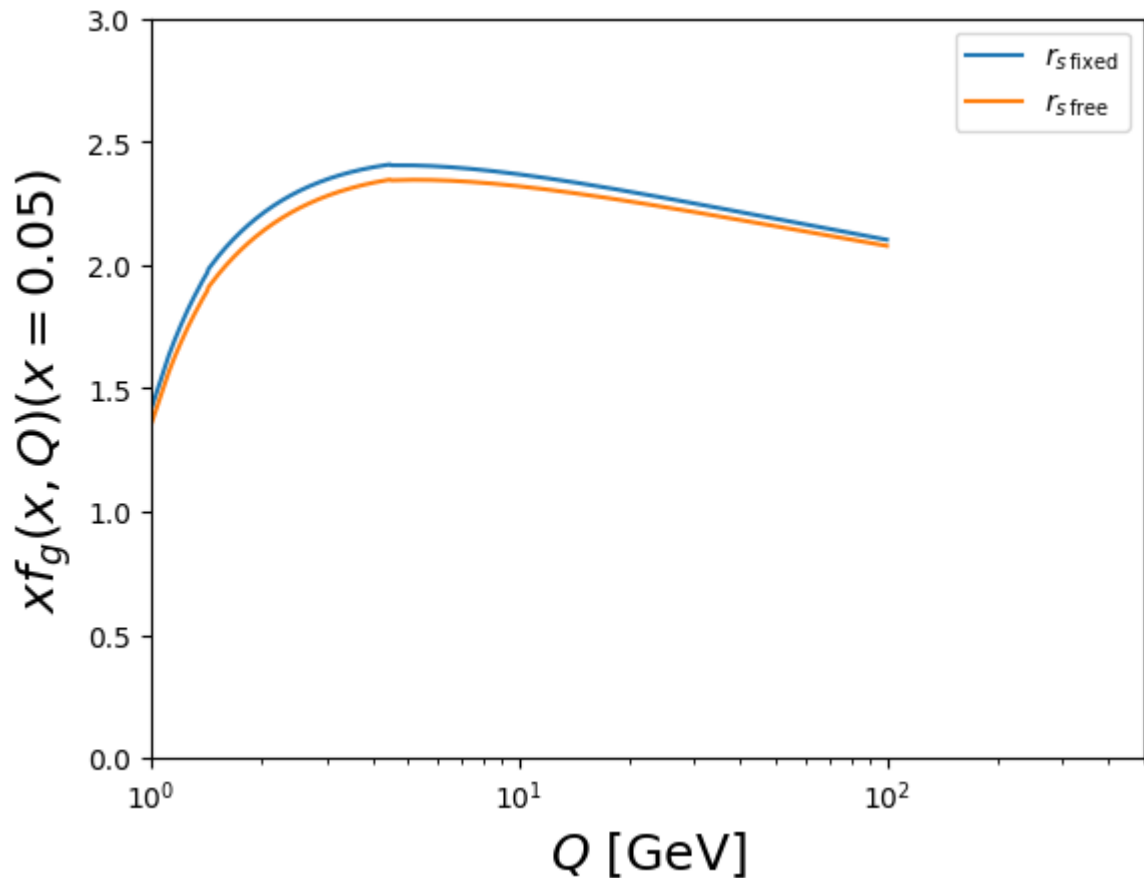
fig = plt.gcf()

#fig.show()

# show(block=False)

fig.savefig('bottom05.pdf')

```



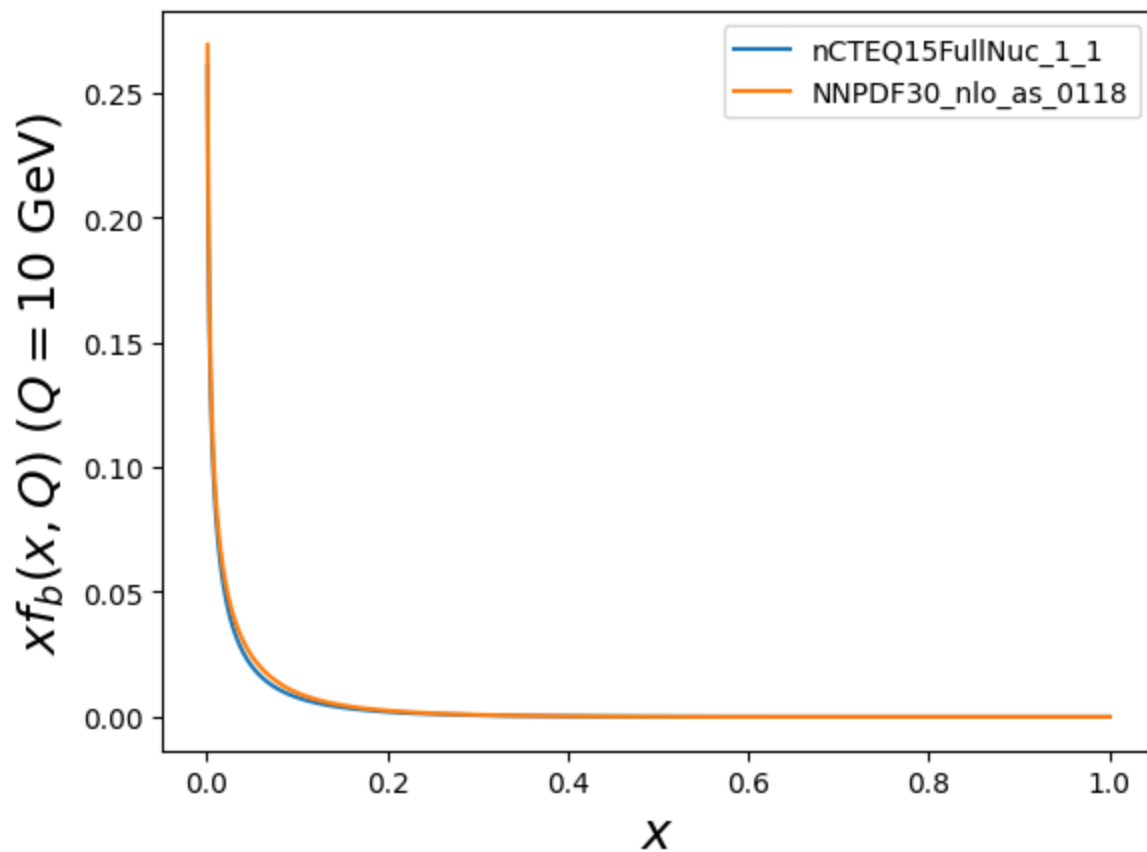
```

In [23]: s1=[xf2[0].xfxQ(5,xx,10.) for xx in xvalues]
          s2=[xf3[0].xfxQ(5,xx,10.) for xx in xvalues]

plt.plot(xvalues,s1,label="nCTEQ15FullNuc_1_1")
plt.plot(xvalues,s2,label="NNPDF30_nlo_as_0118")

plt.xlabel("$x$", fontsize=18)
plt.ylabel("$x f_b(x,Q)$ ($Q = {0:.0f}$ GeV)".format(10.), fontsize=18)
plt.legend(fontsize=10)
plt.show()

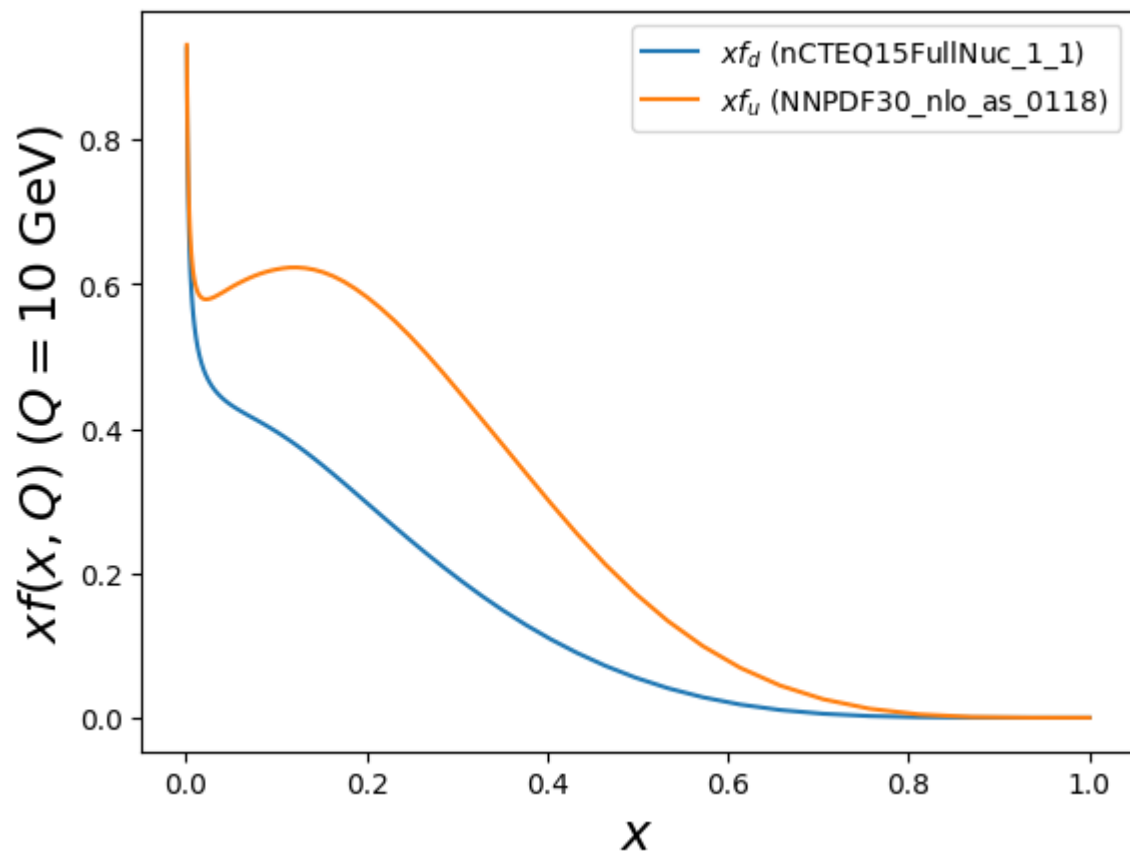
```



```
In [24]: s1=[xf2[0].xfxQ(1,xx,10.) for xx in xvalues]
s2=[xf3[0].xfxQ(2,xx,10.) for xx in xvalues]

plt.plot(xvalues,s1,label="$x f_{d}$ (nCTEQ15FullNuc_1_1)")
plt.plot(xvalues,s2,label="$x f_{u}$ (NNPDF30_nlo_as_0118)")

plt.xlabel("$x$", fontsize=18)
plt.ylabel("$x f(x,Q)$ ($Q = {0:.0f}$ GeV)".format(10.), fontsize=18)
plt.legend(fontsize=10)
plt.show()
```

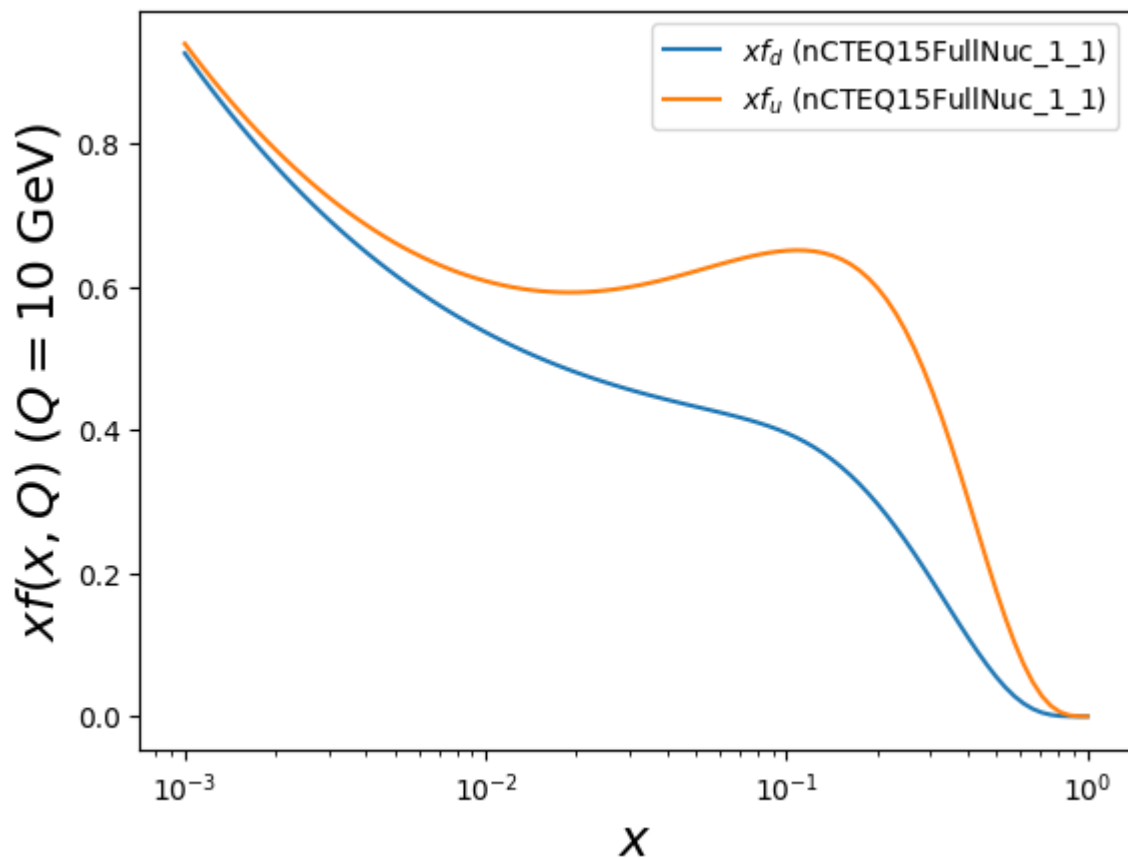


```
In [25]: s1=[xf2[0].xfxQ(1,xx,10.) for xx in xvalues]
s2=[xf2[0].xfxQ(2,xx,10.) for xx in xvalues]

plt.xscale('log')

plt.plot(xvalues,s1,label="$x f_{d}$ (nCTEQ15FullNuc_1_1)")
plt.plot(xvalues,s2,label="$x f_{u}$ (nCTEQ15FullNuc_1_1)")

plt.xlabel("$x$", fontsize=18)
plt.ylabel("$x f(x,Q)$ ($Q = {0:.0f}$ GeV)".format(10.), fontsize=18)
plt.legend(fontsize=10)
plt.show()
```



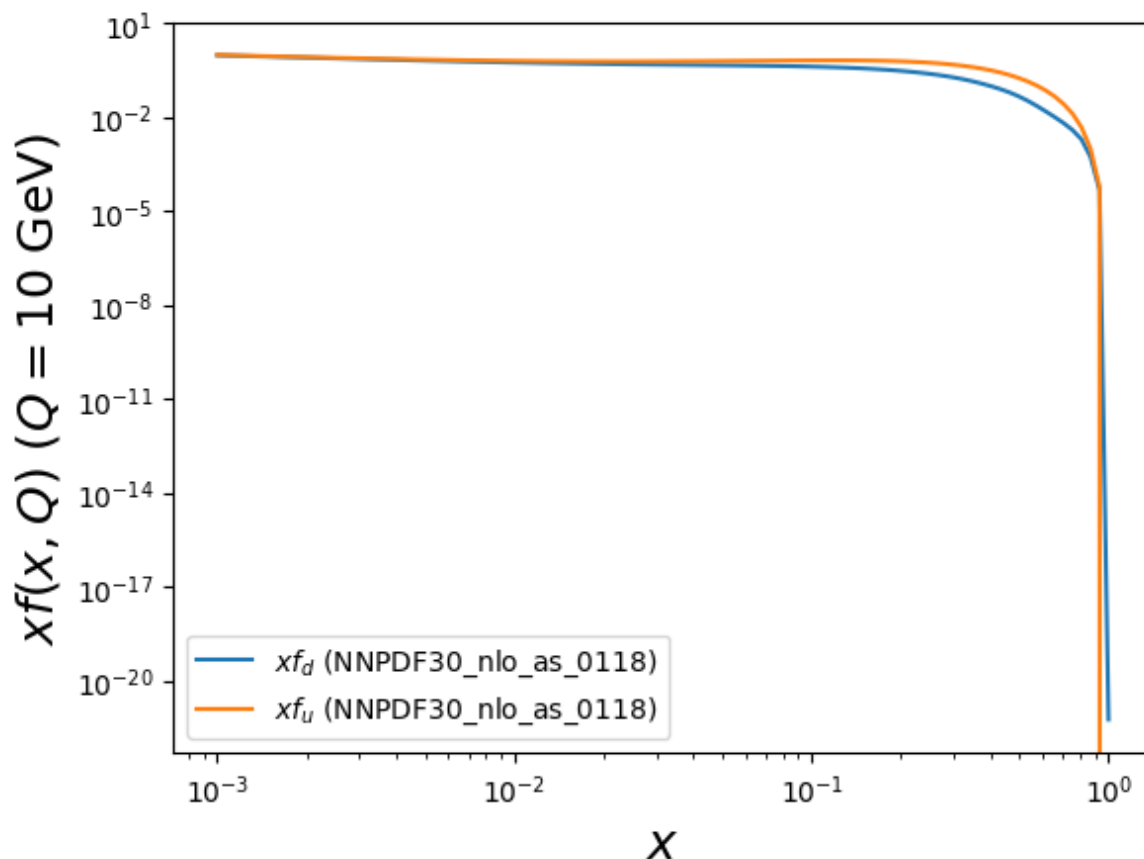
```
In [26]: s1=[xf3[0].xfxQ(1,xx,10.) for xx in xvalues]
s2=[xf3[0].xfxQ(2,xx,10.) for xx in xvalues]

plt.xscale('log')
plt.yscale('log')

plt.plot(xvalues,s1,label="$x f_{d}$ (NNPDF30_nlo_as_0118)")
plt.plot(xvalues,s2,label="$x f_{u}$ (NNPDF30_nlo_as_0118)")

plt.xlabel("$x$", fontsize=18)
plt.ylabel("$x f(x,Q)$ ($Q = {0:.0f}$ GeV)".format(10.), fontsize=18)
plt.legend(fontsize=10)
plt.show()
```





```
In [27]: print("$x f_b(x={0:.4f}, Q={1:.2f} GeV) = {2:.10f}$".format(0.0001, 4.5, xf1
# xf1[0].xfxQ(5, 0.0001, 4.5))
```

```
$x f_b(x=0.0001, Q=4.50 GeV) = -0.0936083615$
```

```
In [28]: t = np.arange(0.1, 1.0, 0.02)
```

```
t
```

```
Out[28]: array([0.1 , 0.12, 0.14, 0.16, 0.18, 0.2 , 0.22, 0.24, 0.26, 0.28, 0.3 ,
0.32, 0.34, 0.36, 0.38, 0.4 , 0.42, 0.44, 0.46, 0.48, 0.5 , 0.52,
0.54, 0.56, 0.58, 0.6 , 0.62, 0.64, 0.66, 0.68, 0.7 , 0.72, 0.74,
0.76, 0.78, 0.8 , 0.82, 0.84, 0.86, 0.88, 0.9 , 0.92, 0.94, 0.96,
0.98])
```

```
In [29]: # t = np.arange(0.1, 1.0, 0.02)
# s=xf1[0].xfxQ(2, t, 10.)
# plot(t, s)
```

```
In [30]: proton_tmp = lhpdf.getPDFSet('NNPDF30_nlo_as_0118')
```

```
In [31]: proton_tmp = proton_tmp.mkPDFs()[0]
```

```
LHAPDF 6.5.3 loading all 101 PDFs in set NNPDF30_nlo_as_0118
NNPDF30_nlo_as_0118, version 2; 101 PDF members
```

```
In [32]: # Let's load the proton
proton_ncteq = lhpdf.getPDFSet('nCTEQ15FullNuc_1_1')
proton_ncteq = proton_ncteq.mkPDFs()[0]
# Let's load the Full lead pdf sets
fulllead = lhpdf.getPDFSet('nCTEQ15FullNuc_208_82')
fullleads = fulllead.mkPDFs()
# Let's load the full Iron
```

```
fulliron = lhapdf.getPDFSet('nCTEQ15FullNuc_56_26')
fullirons = fulliron.mkPDFs()
```

```
LHAPDF 6.5.3 loading all 1 PDFs in set nCTEQ15FullNuc_1_1
nCTEQ15FullNuc_1_1, version 1; 1 PDF members
LHAPDF 6.5.3 loading all 33 PDFs in set nCTEQ15FullNuc_208_82
nCTEQ15FullNuc_208_82, version 1; 33 PDF members
LHAPDF 6.5.3 loading all 33 PDFs in set nCTEQ15FullNuc_56_26
nCTEQ15FullNuc_56_26, version 1; 33 PDF members
```

```
In [33]: # Let's just plot them now as an example. We first define the Q, and x va
```

```
In [34]: xvalues = np.logspace(-3,0,100)
Qval = [5,10]
# select the flavour
flav = 2
```

```
In [35]: # Let's calculate the PDFs on each one of these points
fulllead_x_vals = { qq: [fullleads[0].xfxQ(2, xx, qq) for xx in xvalues]
fulliron_x_vals = { qq: [fullirons[0].xfxQ(flav, xx, qq) for xx in xvalue
```

```
In [36]: %matplotlib inline
```

```
In [37]: %pylab inline
```

```
%pylab is deprecated, use %matplotlib inline and import the required libra
ries.
```

```
Populating the interactive namespace from numpy and matplotlib
```

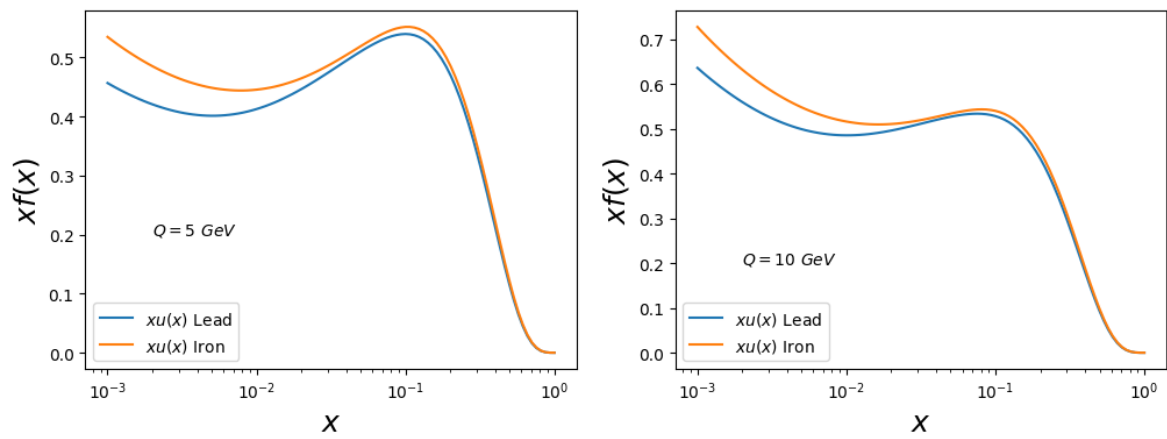
```
/usr/local/lib/python3.10/dist-packages/IPython/core/magics/pylab.py:166:
```

```
UserWarning: pylab import has clobbered these variables: ['isnan', 'inf',
'ceil', 'sin', 'sinh', 'modf', 'pow', 'gamma', 'frexp', 'fmod', 'trunc', '
tanh', 'e', 'nan', 'gcd', 'prod', 'copysign', 'isfinite', 'lcm', 'floor',
'ldexp', 'log1p', 'log10', 'remainder', 'hypot', 'tan', 'fabs', 'nextafter
', 'exp', 'isinf', 'isclose', 'radians', 'log', 'cos', 'sqrt', 'cosh', 'ex
pm1', 'log2', 'degrees', 'pi']
`%matplotlib` prevents importing * from pylab and numpy
```

```
warn("pylab import has clobbered these variables: %s" % clobbered +
```

```
In [38]: # create a new figure
figure(1,figsize=(12,4))

# loop over Q values
for iq,qq in enumerate(Qval):
    # divide it into two subplot vertically
    subplot2grid((1,2),(0,iq))
    semilogx(xvalues, fulllead_x_vals[qq], label='$x_u(x) \ \mathrm{{Lead}}$')
    semilogx(xvalues, fulliron_x_vals[qq], label='$x_u(x) \ \mathrm{{Iron}}$')
    legend(loc=3)
    text(2e-3,0.2, "$Q={0}\ \mathrm{{GeV}}$".format(qq))
    xlabel('$x$', fontsize=18)
    ylabel('$x_f(x)$', fontsize=18)
```



In [39]: *# Python program to demonstrate working  
# of map.*

```
# Return double of n
def addition(n):
    return n + n

# We double all numbers using map()
numbers = (1, 2, 3, 4)
result = map(addition, numbers)
print(list(result))
```

[2, 4, 6, 8]

```
In [40]: def calcerrorpdf(PdfSet,iq,xx, Q, SymAsym='Asym',valence=False):
    """
    • Returns the Observables with errors.
    • Pdfset must contain the central value in zeroth position
    • Pdfset then contains in odd position (1,3,5,...) the + error vector
    • PDFset contains in even positions (2,4,...) the - error vectors.
    """

    def DFfuncp(x):
        return max([x[0]-Fs0,x[1]-Fs0,0])**2
    def DFfuncm(x):
        return max([Fs0-x[0],Fs0-x[1],0])**2
    def DFSym (x):
        return (x[1]-x[0])**2

    def Observable (pdfset):
        return pdfset.xfxQ(iq,xx,Q)

    def Observable_valence (pdfset):
        return pdfset.xfxQ(iq,xx,Q)-pdfset.xfxQ(-iq,xx,Q)

    if not(valence):
        #get the central value
        Fs0 = Observable(PdfSet[0])
        #Get the plus and minus sets
        Fsp = map(Observable,PdfSet[1::2])
        Fsm = map(Observable,PdfSet[2::2])
    else :
        Fs0 = Observable_valence(PdfSet[0])
        Fsp = map(Observable_valence,PdfSet[1::2])
```

```

        Fsm = map(Observable_valence,PdfSet[2:2])

#zip the two vectors returns (f1[0],f2[0]),(f1[1],f2[1]).... so usefu
        Fscombined = zip(Fsp,Fsm)

        if SymAsym == 'Asym':
            DFp,DFm = sqrt(sum(list(map(DFfuncp,Fscombined)))) ,sqrt(sum(list(
                return DFp,DFm,Fs0
        elif SymAsym == 'sym':
            DFp = 0.5 * sqrt(sum(list(map(DFsym,Fscombined))))
            return DFp, DFp, Fs0

```

```
In [41]: fullleads[0].xfxQ(1,0.1,10.)
```

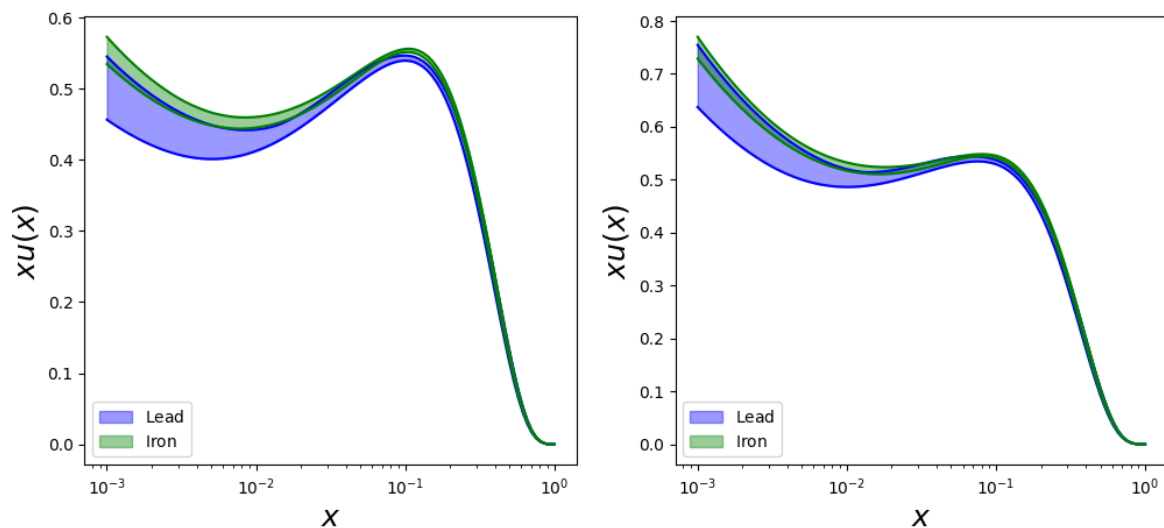
```
Out[41]: 0.5809487232475684
```

```
In [42]: # Let's add some errors
lead_error = {qq: np.array([calcerrorpdf(fullleads, flav, xx, qq) for xx
iron_error = {qq: np.array([calcerrorpdf(fullirons, flav, xx, qq) for xx

```

```
In [43]: figure(2, figsize(12,5))
for iq, qq in enumerate(Qval):
    subplot2grid((1,2),(0,iq))
    #####
    #LEAD
    #####
    # plot central + Deltap
    semilogx(xvalues, fulllead_x_vals[qq]+lead_error[qq][:,0],color='b')
    #plot central - Delta m
    semilogx(xvalues, fulllead_x_vals[qq]-lead_error[qq][:,1],color='b')
    # fill in between
    fill_between(xvalues, fulllead_x_vals[qq] + lead_error[qq][:,0], full
    #####
    # IRON
    #####
    # plot central + Deltap
    semilogx(xvalues, fulliron_x_vals[qq]+iron_error[qq][:,0],color='g')
    #plot central - Delta m
    semilogx(xvalues, fulliron_x_vals[qq]-iron_error[qq][:,1],color='g')
    # fill in between
    fill_between(xvalues, fulliron_x_vals[qq] + iron_error[qq][:,0], full
    p1 = Rectangle((0,0),0,1,color='b',alpha=0.4)
    p2 = Rectangle((0,0),0,1,color='g',alpha=0.4)
    legend([p1,p2],[' $\mathrm{Lead}$ ','$',' $\mathrm{Iron}$ ','$'],loc=3)
    xlabel('$x$', fontsize=18)
    ylabel('$xu(x)$', fontsize=18)

```



## Print all flavours

### Fixed Q

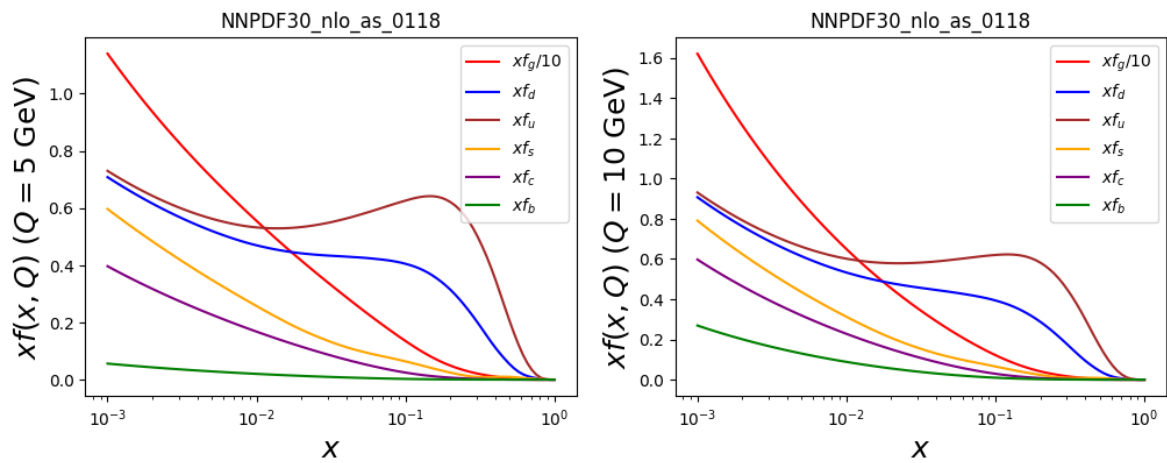
```
In [44]: xvalues = np.logspace(-3,0,100)
        Qval = [5,10]
        # Flavours:
        # 0: gluon, 1: down, 2: up, 3: strange, 4: charm, 5: bottom
        # Here, I don't show the following flavours: dbar (-1), ubar (-2)
        flavours = np.arange(6)
        flavours_label = ["g/10", "d", "u", "s", "c", "b"]
        flavours_color = ["red", "blue", "brown", "orange", "purple", "green"]

In [45]: # Let's calculate the PDFs on each one of these points and for each flavour
        xf3_x_vals = {fl: [{qq: [xf3[0].xfxQ(fl, xx, qq) for xx in xvalues] for qq
        # Divide the gluon PDF by 10
        for qq in Qval:
            for i in range(len(xf3_x_vals[0][0][qq])):
                xf3_x_vals[0][0][qq][i] /= 10

In [46]: # create a new figure
        figure(1,figsize=(12,4))

        # Let's draw all the PDFs (without error))
        for iq,qq in enumerate(Qval):
            subplot2grid((1,2),(0,iq))
            for fl in flavours:
                plt.plot(xvalues, xf3_x_vals[fl][0][qq], label='$x f_{0}$'.format(f
                plt.xscale("log")

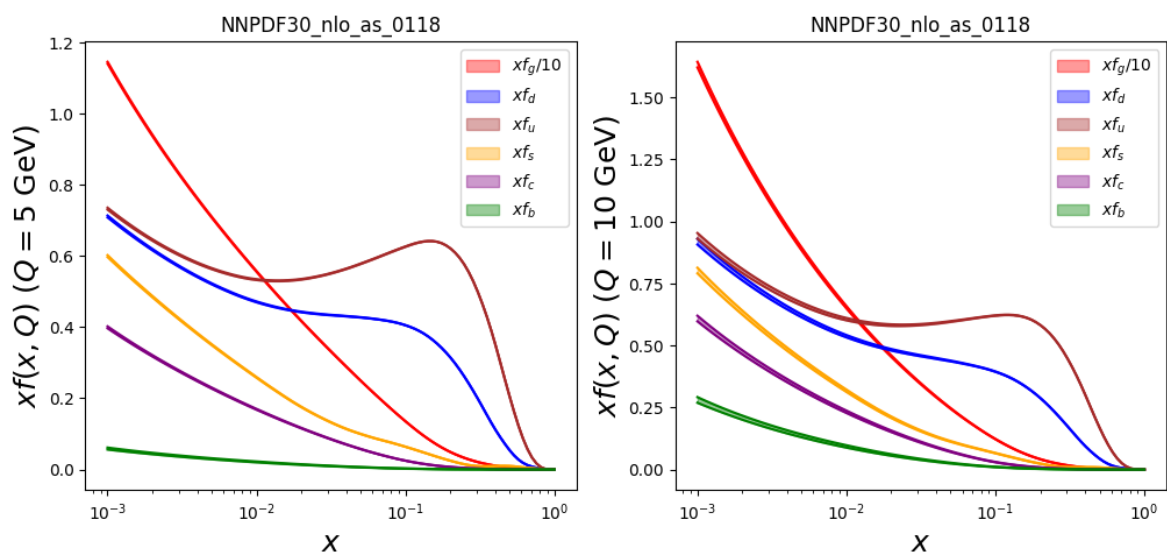
            plt.xlabel("$x$", fontsize=18)
            plt.ylabel("$xf(x,Q)$ ($Q={0}$ GeV)".format(qq), fontsize=18)
            plt.legend(fontsize=10)
            plt.title("NNPDF30_nlo_as_0118")
```



```
In [47]: # Let's add some errors for NNPDF30_nlo_as_0118
for flav in range(5+1):
    error3 = {fl: [{qq: np.array([calcerrorpdf(xf3, flav, xx, qq) for xx in
```

```
In [48]: figure(2, figsize(12,5))
for iq, qq in enumerate(Qval):
    subplot2grid((1,2),(0,iq))
    for fl in flavours:
        #####
        # NNPDF30_nlo_as_0118
        #####
        # plot central + Deltap
        plt.plot(xvalues, xf3_x_vals[fl][0][qq]+error3[fl][0][qq][:,0],co
        #plot central - Delta m
        plt.plot(xvalues, xf3_x_vals[fl][0][qq]-error3[fl][0][qq][:,1],co
        # fill in between
        plt.fill_between(xvalues, xf3_x_vals[fl][0][qq] + error3[fl][0][q
                        color=flavours_color[fl], alpha=0.4,
                        label="$xf_{\text{" + fl + "}}$".format(flavours_label[fl]))

    plt.xscale("log")
    plt.legend(fontsize=10)
    plt.xlabel('$x$', fontsize=18)
    plt.ylabel('$xf(x,Q)$ ($Q = {0}$ GeV)'.format(qq), fontsize=18)
    plt.title("NNPDF30_nlo_as_0118")
```



## Fixed x

```
In [49]: xvalues = [0.001,0.2]
Qval = np.logspace(0,2,100)
flavours = np.arange(6)
flavours_label = ["g/10","d","u","s","c","b"]
flavours_color = ["red","blue","brown","orange","purple","green"]

In [50]: # Let's calculate the PDFs on each one of these points and for each flavo
xf3_x_vals = {fl: [{xx: [xf3[0].xfxQ(fl, xx, qq) for qq in Qval] for xx in i

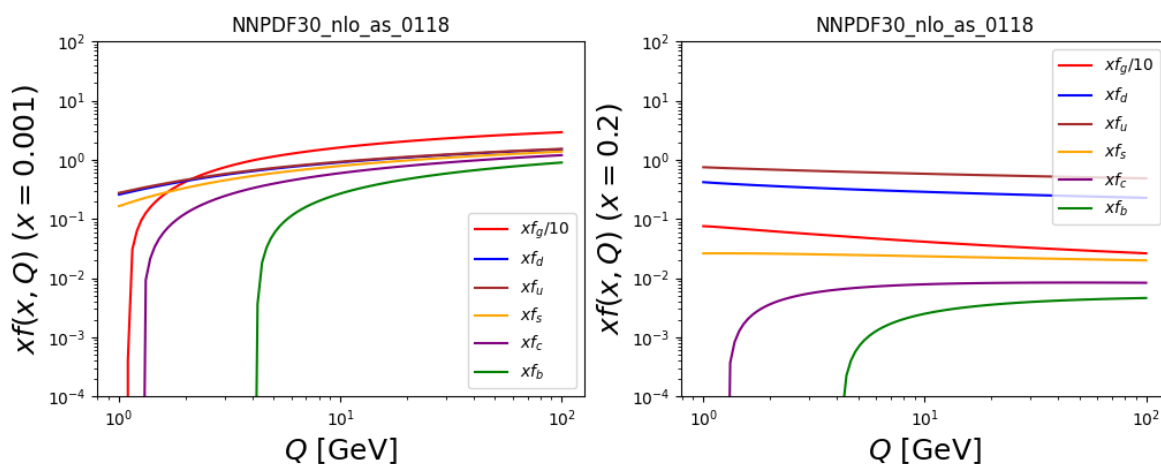
# Divide the gluon PDF by 10
for xx in xvalues:
    for i in range(len(xf3_x_vals[0][0][xx])):
        xf3_x_vals[0][0][xx][i] /= 10

In [51]: # create a new figure
figure(1,figsize=(12,4))

# Let's draw all the PDFs (without error))
for ix,xx in enumerate(xvalues):
    subplot2grid((1,2),(0,ix))
    for fl in flavours:
        plt.plot(Qval, xf3_x_vals[fl][0][xx], label='$x f_{0}$'.format(flav

plt.xscale("log")
plt.yscale("log");plt.ylim((1e-4,1e2))

plt.xlabel("$Q$ [GeV]", fontsize=18)
plt.ylabel("$xf(x,Q)$ ($x={0}$)".format(xx), fontsize=18)
plt.legend(fontsize=10)
plt.title("NNPDF30_nlo_as_0118")
```



## Play with Integration

```
In [52]: from scipy import integrate
```

```
In [53]: xvalues = np.logspace(-3,0,100)
xvalues=xvalues[:-1]
print( len(xvalues) )
```

```
print( xvalues[0::10] )
```

```
99
[0.001      0.00200923 0.00403702 0.00811131 0.01629751 0.03274549
 0.06579332 0.13219411 0.26560878 0.53366992]
```

```
In [54]: xf0[0].xfxQ(3,0.1,5)
```

```
Out[54]: 0.06396391472409065
```

```
In [55]: istr=3
q0=1.4
sFix=[xf0[0].xfxQ(istr,xx,q0) for xx in xvalues]
dbFix=[xf0[0].xfxQ(-1,xx,q0) for xx in xvalues]
ubFix=[xf0[0].xfxQ(-2,xx,q0) for xx in xvalues]

sFree=[xf1[0].xfxQ(istr,xx,q0) for xx in xvalues]
dbFree=[xf1[0].xfxQ(-1,xx,q0) for xx in xvalues]
ubFree=[xf1[0].xfxQ(-2,xx,q0) for xx in xvalues]

print( len(sFix) , len(xvalues) )
print( sFix[0::20] )
```

```
99 99
[0.20579890141608911, 0.16608209702147544, 0.1262576859409809, 0.075113879
5872819, 0.013859900322095598]
```

```
In [56]: print( len(sFix) , len(xvalues) )
```

```
99 99
```

```
In [57]: sInt0=integrate.simpson(sFix, x=xvalues)
dbInt0=integrate.simpson(dbFix, x=xvalues)
ubInt0=integrate.simpson(ubFix, x=xvalues)
print(sInt0, dbInt0, ubInt0)
print(2.0*sInt0/(dbInt0+ubInt0))
```

```
0.015703466659388297 0.028751734603037038 0.03329210432038654
0.5062055131298371
```

```
In [58]: sInt1=integrate.simpson(sFree, x=xvalues)
dbInt1=integrate.simpson(dbFree, x=xvalues)
ubInt1=integrate.simpson(ubFree, x=xvalues)
print(sInt1, dbInt1, ubInt1)
print(2.0*sInt1/(dbInt1+ubInt1))
```

```
0.027587341926872434 0.017188489825357298 0.03139653720696964
1.1356314326435053
```

## Compute the momentum fraction of all flavors vs. Q and plot

```
In [59]: Qval = np.logspace(0,4,100)
xvalues = np.logspace(-3,0,100)
flavours = np.arange(6)
flavours_label = ["g", "d", "u", "s", "c", "b"]
flavours_color = ["red", "blue", "brown", "orange", "purple", "green"]
```

```
In [60]: # I use NNPDF30_nlo_as_0118
```



```

# Check that the sum is always equal to 1.
mom_sum=np.zeros(len(Qval))
for fl in flavours:
    mom_frac = np.zeros(len(Qval))
    for iq,Q in enumerate(Qval):
        s3=[xf3[0].xfxQ(fl,xx,Q) for xx in xvalues]
        mom_frac[iq]=100*integrate.simpson(s3,x=xvalues)
        mom_sum[iq]+=mom_frac[iq]
    plt.plot(Qval,mom_frac,color=flavours_color[fl],label="$\int x f_{\{\}}$")

# Print the sum of the momentum fraction for every flavor (and check that
plt.plot(Qval,mom_sum,color='black',label='sum')
plt.legend()

plt.xscale("log")
plt.yscale("log")
plt.xlim((1e0,1e4))
plt.ylim((1e-1,1e2+1))
plt.xlabel("$Q$ [GeV]",fontsize=18)
plt.ylabel("momentum fraction %",fontsize=18)

```

Out[60]: Text(0, 0.5, 'momentum fraction %')

