X4Pro – fully relational EXFOR database

Viktor Zerkin
International Atomic Energy Agency, Nuclear Data Section

Technical Meeting NRDC-2022 of the International Network of Nuclear Reaction Data Centres,
IAEA Headquarters + WebEx, Vienna, Austria, 14 - 17 June 2022

X4Pro concept

Concept:

Now: EXFOR relational present EXFOR meta-data in tables accessible for SQL commands, but data points (numerical data) are stored as part of SUBENT in BLOB. In order to be used, numerical data need to be extracted from BLOB to EXFOR file, converted computational form (like C4/C5). So, our current EXFOR relational database requires a lot of additional software: EXFOR reader, parser, converter.

New: EXFOR relational database extended by tables with
(a) data points in original EXFOR and (b) computational form,
(c) monitor and decay data for automatic renormalization,
(d) instructions for experts' data modifications.
So, the database allows to retrieve numeric data using only SQL commands (and even automatically renormalize data by SQL commands).

Plan: to provide X4Pro (SQLite) database with simple demo programs-examples in Python, Fortran, [maybe JavaScript] implementing basic tasks: data search, retrieval, plotting, renormalization, users' modifications.

Net result:

1. X4Pro: fully/truly relational EXFOR database (total size in SQLite: 7.1GiB)
2. SQL access to all meta-data and to all data points
3. Trivial multi-languages programming of data search/filter/sort/retrieval/renormalization

X4Pro implementation

Translation from MariaDB to SQLite is done automatically by a bash script
working ~4 hours and producing a single 7Gb file x4sqlite1.db with 7 new tables:

1. x4pro_ds 179, 372 rows 29Mb datasets
2. x4pro_hdr 1,467,092 rows 123Mb headers
3. x4pro_x4data 18,974,844 rows 1.8Gb EXFOR data points (json)
4. x4pro_x4cdat 18,974,844 rows 1.9Gb EXFOR data points in basic units (json)
5. x4pro_c5dat 18,974,844 rows 1.2Gb Computational data points (real)
   + total sys/stat/partial errors
   + old and new monitor CS data
6. x4pro_autocorr 16,586 rows 10Mb decay data for product and monitor
7. x4pro_expertcorr 3 rows 3Kb experts' corrections (Python)

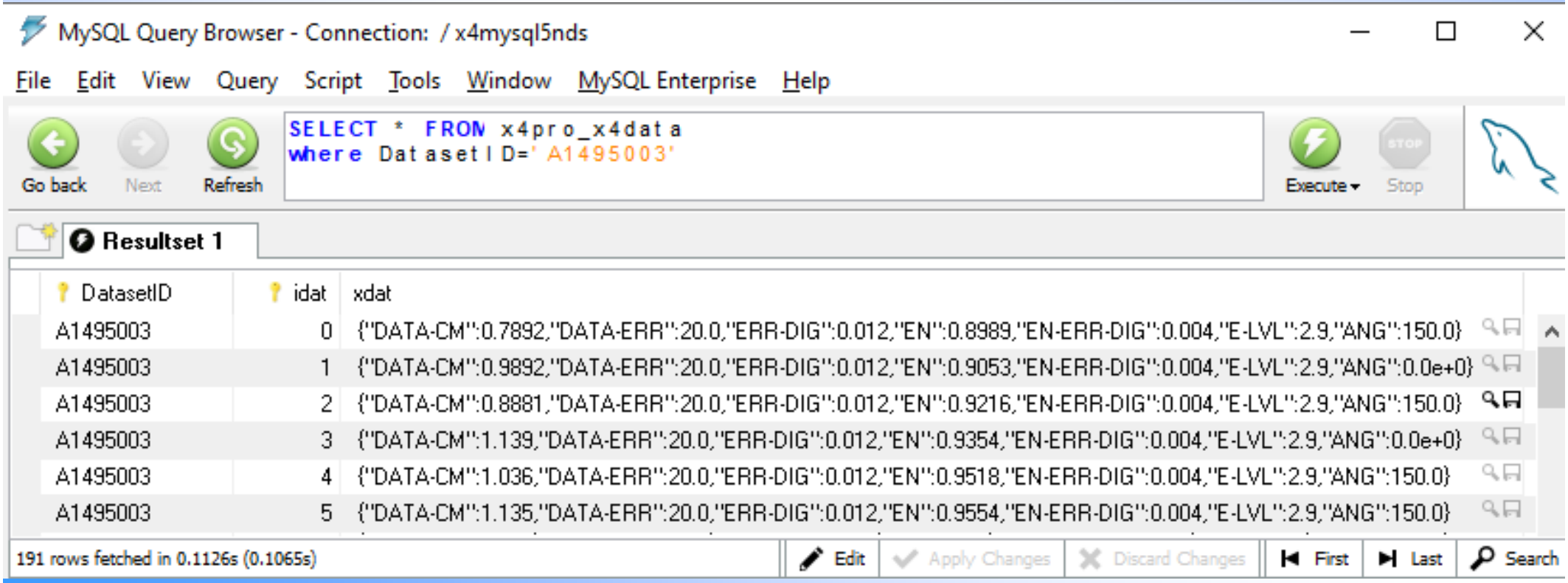
Total: 7 additional tables, size: +5Gb

Table Name	Engine	Rows	Data length	Index length	Update time
x4pro_autocorr	MyISAM	16586	10.2 MB	259 kB	2022-04-20 17:56:29
x4pro_c5dat	MyISAM	18974844	1.2 GB	368.2 MB	2022-04-20 17:56:49
x4pro_ds	MyISAM	179372	29.4 MB	2.7 MB	2022-04-20 17:56:49
x4pro_expertcorr	MyISAM	3	3.1 kB	2 kB	2022-04-20 16:18:50
x4pro_hdr	MyISAM	1467092	123.4 MB	23.9 MB	2022-04-20 17:56:49
x4pro_x4cdat	MyISAM	18974844	1.9 GB	368.2 MB	2022-04-20 17:56:49
x4pro_x4data	MyISAM	18974844	1.8 GB	368.2 MB	2022-04-20 17:56:49

Table x4pro_x4data

```
create table x4pro_x4data (
     DatasetID          varchar(9) not null
    ,idat               integer    null
    ,xdat               json null
    ,PRIMARY KEY        (DatasetID,idat)
)
```

Using MySQL Query Browser



Using SQLite DB Browser

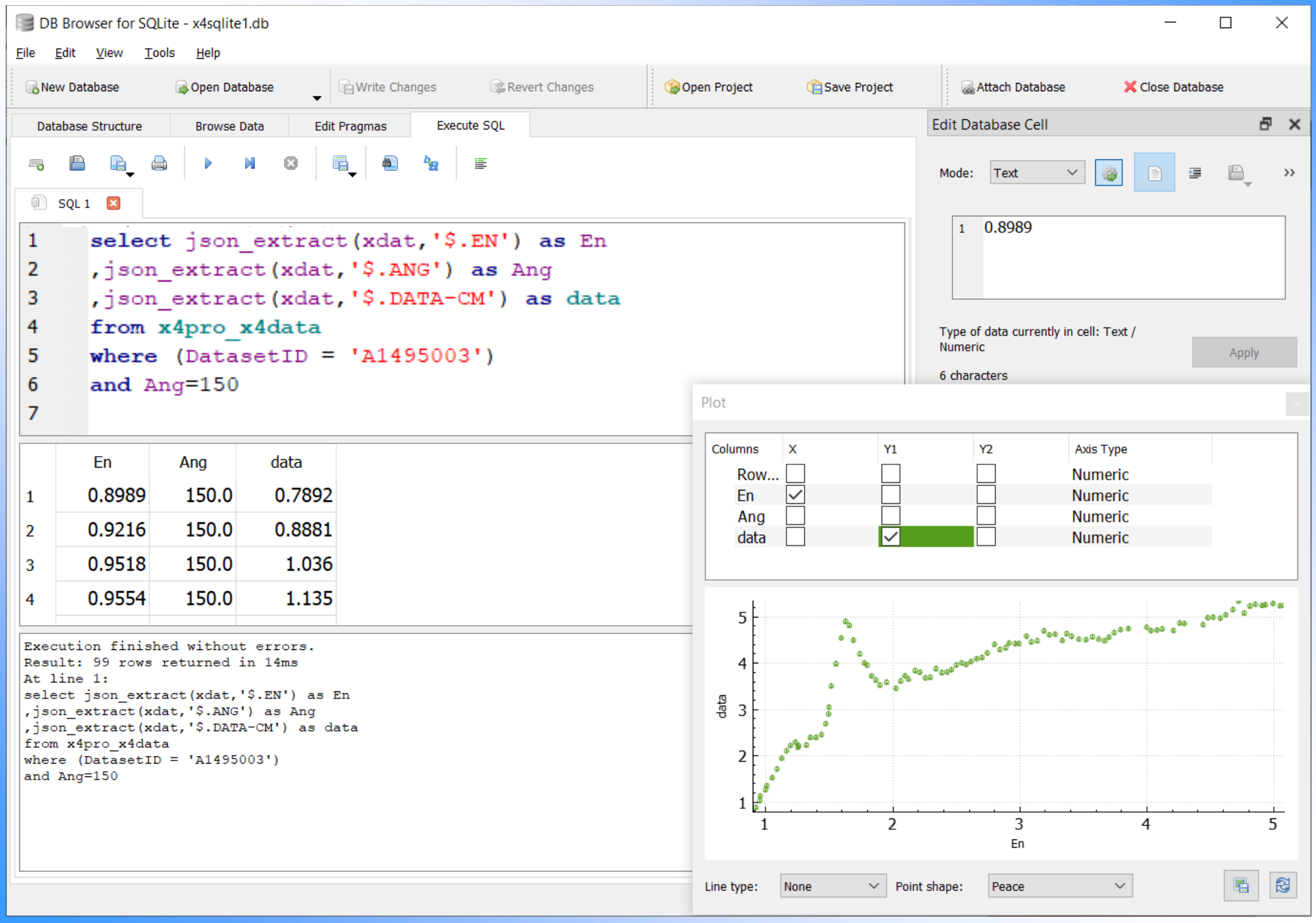


Table x4pro_c5dat

```
create table x4pro_c5dat (
    DatasetID           varchar(9) not null
   ,idat                integer    null
   ,y                   real null -- measured data
   ,dy                  real null
   ,x1                  real null -- ind. variable 1
   ,dx1                 real null
   ,x2                  real null -- ind. variable 2
   ,dx2                 real null
   ,x3                  real null -- ind. variable 3
   ,dx3                 real null
   ,x4                  real null -- ind. variable 4
   ,dx4                 real null
   ,x5                  real null -- ind. variable 5
   ,dx5                 real null
   ,dyerr               real null -- given data error
   ,dysys               real null -- total systematic error
   ,dystat              real null -- total uncorrelated error
   ,dyprt               real null -- total part.corr.error
   ,Em0                 real null -- energy of monitor point
   ,m0                  real null -- old monitor CS
   ,dm0                 real null
   ,m1                  real null -- new monitor CS
   ,dm1                 real null
   ,Fcm0                real null -- correction factor
   ,cdat                json null -- additional: ilevel, product,..
   ,PRIMARY KEY         (DatasetID,idat)
)
```

Table x4pro_c5dat

MySQL Query Browser - Connection: / x4mysql5nds

File Edit View Query Script Tools Window MySQL Enterprise Help

Go back | Next | Refresh | Execute | Stop

```
SELECT DatasetID, idat, y, dy, x1, dx1, dyerr,
Em0, m0, dm0, m1, dm1, Fcm0
FROM x4pro_c5dat
where DatasetID='30581004'
```

Resultset 1

DatasetID	idat	y	dy	x1	dx1	dyerr	Em0	m0	dm0	m1	dm1	Fcm0
30581004	0	0.0219	0.0021	13000000	50000	0.0021	13000000	0.112833	0.007	0.114678	0.00169076	1.01635
30581004	1	0.0218	0.0014	13300000	50000	0.0014	13300000	0.112909	0.007	0.115855	0.00152817	1.02609
30581004	2	0.0241	0.0015	13900000	100000	0.0015	13900000	0.106857	0.007	0.114767	0.00125861	1.07402
30581004	3	0.0277	0.0016	14500000	100000	0.0016	14500000	0.0990947	0.00602105	0.109563	0.00118709	1.10564
30581004	4	0.0292	0.0019	15100000	100000	0.0019	15100000	0.0902632	0.00566842	0.101554	0.00126434	1.12509
30581004	5	0.0249	0.0018	15500000	100000	0.0018	15500000	0.0876595	0.00541351	0.0954231	0.00130102	1.08857
30581004	6	0.0237	0.0022	15900000	100000	0.0022	15900000	0.0813719	0.00494375	0.0891203	0.00130266	1.09522
30581004	7	0.0239	0.0026	16600000	50000	0.0026	16600000	0.0723261	0.00406087	0.0784956	0.00127819	1.0853
30581004	8	0.0213	0.0026	17400000	100000	0.0026	17400000	0.0634344	0.00397812	0.0678528	0.00126376	1.06965
30581004	9	0.0181	0.0024	17800000	50000	0.0024	17800000	0.0592387	0.00359677	0.0632639	0.00125757	1.06795

10 rows fetched in 0.0072s (0.1015s) | Edit | Apply Changes | Discard Changes | First | Last | Search

1: 1

```
30581004  x4u:20090506  #1980,Zupranska #Pts:10
 #[0]#---Monitor xs-data
 #[0]#Reaction: 25-MN-55(N,A)23-V-52,,SIG
 #[0]#Monitor:  26-FE-56(N,P)25-MN-56,,SIG
 #m0: {20377002,H.LISKIEN+,J,JNE/AB,19,73,196502} $ fe56np;#[0]#old monit-ref
 m0: exfor$20377002_fe56np;      #[0]#old monitor(energy) in EXFOR
 m1: recom$fe56np;               #[0]#new monitor(energy)
 dy=dy/y;                        #[0]#to rel. uncertainties----
 y=y/m0*m1;                      #[0]#renormalizing CS
 dy=(dy**2-dm0**2+dm1**2)**0.5;  #[0]#replace monitor uncertainties
 dy=dy*y;                        #[0]#to abs. uncertainties
```

Now on Web

m0, m1: old and new monitor cross sections

m0, dm0: “old monitor” – monitor-reaction cross sections used by authors

Source of data:

1. DATA, COMMON sections: MONIT, MONIN-ERR (EN, EN-NRM)
2. MONIT-REF pointing to another EXFOR data
3. MONIT-REF pointing to ENDF library (e.g., ENDF/B-5 Standards sub-library)
4. MONIT-REF pointing to “a publication” //--> Archive of old Monitors

m1, dm1: “new monitor” – monitor-reaction cross sections “recommended” now

Source of data:

1. IAEA Standards-2017
2. IRDFF-II (2019)

Automatic renormalization of data point

```
def auto_corr_point(row):
    y=row['y']                  #initial value
    y0=y
    fc=row.get('FcNew')         #factor: m1/m0
    if fc is None: return 0 #unchanged
    y=y*fc;                     #correction exp. cs
    row['y']=y                  #store new value
    dy=row.get('dy')
    if (dy is None): return 1 #updated
    if (y0!=0):
        m0=row.get('m0')        #monitor: old cs
        dm0=row.get('dm0')
        m1=row.get('m1')        #monitor: new cs
        dm1=row.get('dm1')
        if (m0 is not None) and (dm0 is not None)and(m1 is not None)and(dm1 is not None):
            dy=dy/y0             #to rel. uncertainties
            dm0=dm0/m0
            dm1=dm1/m1
            if (dy>dm0):
                dy=dy**2-dm0**2+dm1**2; #determination the quadrature of new total error
            else:
                dy=dy**2+dm1**2; #determination the quadrature of new total error
            dy=dy**0.5*y;           #determination the absolute value of new total error
            row['dy']=dy            #store new uncertainty
    return 1 #updated
```

Using SQL Views

Using Tables only

```
select x4data_cdat.DatasetID,REACODE.fullCode
 ,ENTRY.YearRef1,ENTRY.Author1Ini,ENTRY.Author1
 ,x4data_cdat.idat as iPoint
 ,x4data_cdat.y    as Sig
 ,x4data_cdat.dy   as dSig
 ,x4data_cdat.x1   as En
 ,x4data_cdat.dx1  as dEn
 ,x4data_cdat.x2   as Eout
 ,x4data_cdat.dx2  as dEout
 ,x4data_cdat.x3   as An
 ,x4data_cdat.dx3  as dAn
from x4data_cdat
 inner join REACODE on
  REACODE.ReacodeID=x4data_cdat.DatasetID
 inner join REACSTR on
  REACSTR.ReacodeID=REACODE.ReacodeID
 inner join SUBENT on REACODE.SubentID=SUBENT.SubentID
 inner join ENTRY on ENTRY.EntryID=SUBENT.EntryID
where (REACSTR.Target like 'F-19')
  and (REACSTR.Reaction like 'n,x')
  and (REACSTR.Quant like 'DAE')
  and (REACODE.outParticles like '[n]')
  and (REACSTR.SF8='')
  and ((REACSTR.SF9='') or (REACSTR.SF9='EXP'))
  and (REACODE.nReacstr=1)
  and (An>=25) and (An<=45)
order by REACODE.fullCode,
  ENTRY.YearRef1 desc,x4data_cdat.DatasetID
  ,En,An,Eout,x4data_cdat.idat
```

Using Views

```
select * from dae1
where Target like 'F-19'
 and Reaction like 'n,x'
 and outParticles like '[n]'
 and (An>=25) and (An<=45)
order by fullCode,
 YearRef1 desc,DatasetID,
 En,An,Eout,iPoint
```

Example: SIG

```
select * from sig1
where Target='Mn-55'
and MT=107

select * from sig1
where Target='Mn-55'
and Reaction like 'n,a'
```

Test cases, platforms and technologies

I. Retrieve experimental data from local X4Pro, evaluated data from Web ENDF retrieval system, and plot using Python3 with Plotly

1. Cross sections (MF3)
2. Angular distributions (MF4)
3. Emission spectra (MF5)
4. Double differential cross sections (MF6)

for young

II. Retrieve data from local X4Pro using GFortran and GCC

1. Cross sections (MF3)
2. Double differential cross sections (MF6)

for old

III. Data renormalization/modification on Python (+ENDF +Plotly)

1. Automatic renormalization
2. User's modifications
3. Experts' modifications (taken from database)
4. Ratios to cross sections recalculations

for evaluators

for SG50?

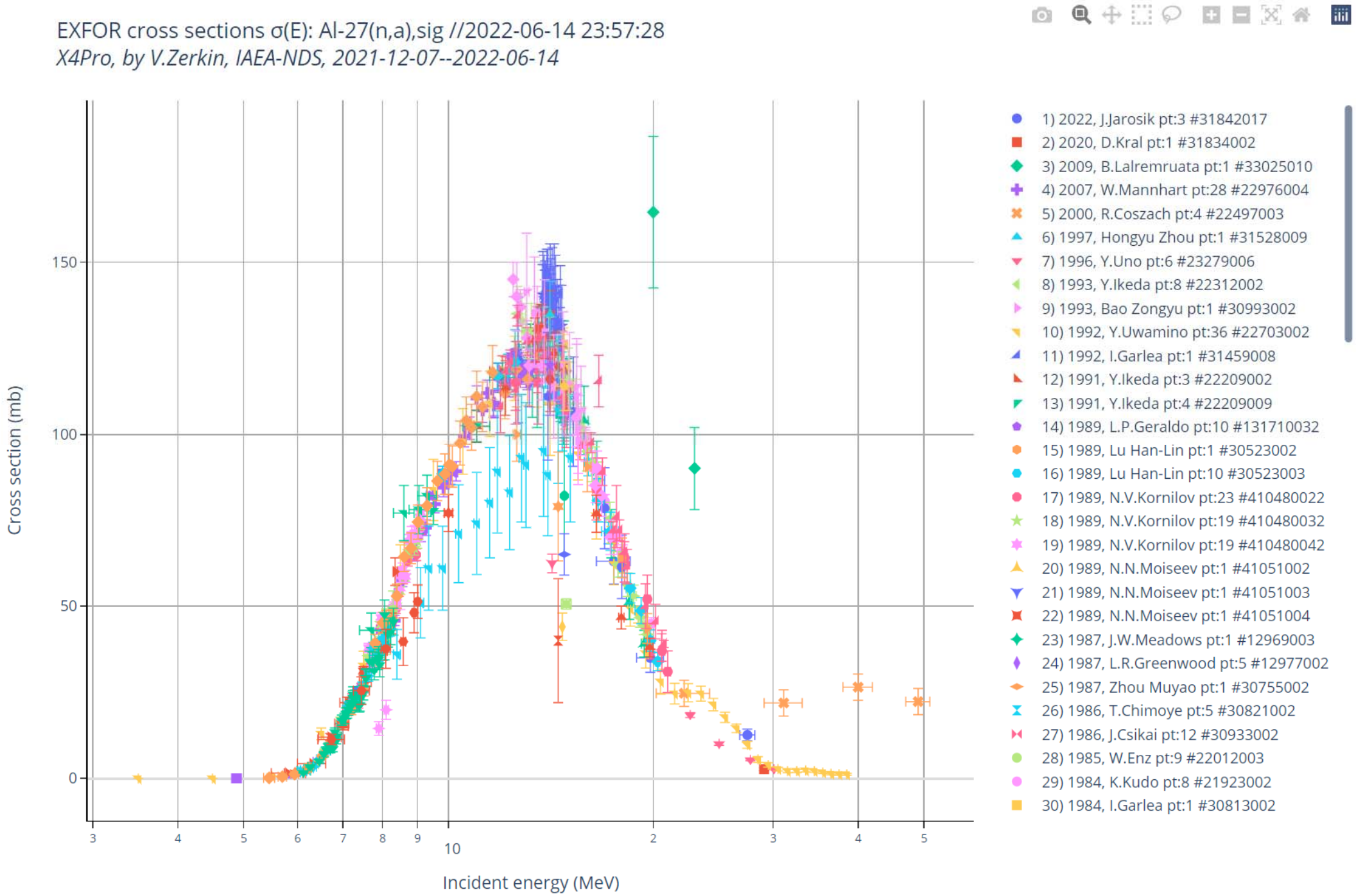
IV. Data retrievals from local X4Pro using javascript (+ENDF +Plotly)

1. Cross sections (MF3) with GUI/Html5
2. Retrievals from javascript under Node.js

for fun

Note. ***I, II, III, IV.1: on Windows (portable), Linux, MacOS***
IV.2 tested on MacOS only

Cross sections from EXFOR (SIG)



Source-code

#Main code

sig0x.py

```
import os
import sys
sys.path.append('./')
import dbConn
from rx4db import *
from exfor2plot import *
print("Program: sig0x.py, ver. 2022-06-14")
print("Author:  V.Zerkin, IAEA-NDS, Vienna, 2021-2022")
print("Purpose: Retrieve and plot EXFOR cross sections\n")

target="Al-27"; react="n,a"; quant=",sig"
xtype='linear'; ytype='linear'
if (len(sys.argv)>1) and (sys.argv[1]!=''): target=str(sys.argv[1])
if (len(sys.argv)>2) and (sys.argv[2]!=''): react=str(sys.argv[2])
if (len(sys.argv)>3) and (sys.argv[3]=='log'): xtype='log'
if (len(sys.argv)>4) and (sys.argv[4]=='log'): ytype='log'
plotTitle=target+'('+react+')'+quant;
outhtml=target.replace('-','')+react.replace(',','')+'-1.html'

print("---Retrieve EXFOR data from SQL database---")
conn=dbConn.getConnSQLx4db()
if conn is None:
    print("___0___No connection...")
    sys.exit(1)
print("Connected to: ["+dbConn.dbType+"]")
cursor=dbConn.getCursor(conn)

sql=getX4SqlSearchCS(target,react)
print("SQL:\n"+sql)
try:
    cursor.execute(sql)
    rows=cursor.fetchall()
except Exception as ex:
    print("___1___execute-SQL error: ", ex)
    rows=[]
conn.close()
print("\nEXFOR SQL executed: OK")

datasets=getDatasets(rows)
print('#datasets:',len(datasets))
ldata=len(datasets)
if (ldata<=0):
    print("---No data found---")
    sys.exit(2)

data1=prepareExforDataForPlot(datasets)
myOfflinePlot(data1,'EXFOR cross sections \u03c3(E): '+plotTitle
 +"<br><i>X4Pro, by V.Zerkin, IAEA-NDS, 2021-12-07--2022-06-20</i>"
        ,'Incident energy (MeV)'
        ,'Cross section (mb)'
        ,xtype=xtype,ytype=ytype,filename=outhtml)
print('\nProgram successfully completed')
```

exfor2plot.py

```
import plotly
from plotly.graph_objs import Scatter, Layout

#_________________Preparing EXFOR data for plot_________________
def prepareExforDataForPlot(datasets):
    ldata=len(datasets)
    ii=0
    data1=[]
    for dataset in datasets:
        #if (len(dataset['x'])<=1): continue
        error_y=dict(type='data',array=dataset['dy'],visible=True,thickness=0.9)
        error_x=dict(type='data',array=dataset['dx'],visible=True,thickness=0.9)
        tr=Scatter(x=dataset['x'],y=dataset['y'],error_y=error_y,error_x=error_x
        ,text=dataset['x4lbl']
        ,name=str(ii+1)+') '+dataset['x4lbl']+' pt:'+str(len(dataset['x']))
        +' #'+dataset['DatasetID']
        ,marker_symbol=str(ii%33)
        ,marker_size=8
        ,mode="markers"
        )
        data1.append(tr)
        ii+=1
        print('Plot:'+str(ii)+'/'+str(ldata)+') #'+str(dataset['DatasetID'])
        +' '+str(dataset['x4lbl'])+' pt:'+str(len(dataset['x'])))
    return data1

#_________________Plot data from EXFOR and ENDF_________________
def myOfflinePlot(data1,ptitle,xtitle,ytitle
,xtype='linear',ytype='linear',filename='temp-plot.html'):
    plot1={}
    plot1['data']=data1
    xaxis=dict(title=xtitle,showline=True,linecolor='black'
        ,ticks='outside',showgrid=True,gridcolor='#aaaaaa',type=xtype)
    yaxis={'title':ytitle,'showline':True,'linecolor':'black'
        ,'showgrid':True, 'gridcolor':'#aaaaaa','ticks':'outside','type':ytype
        ,'zeroline':True, 'zerolinecolor':'#dddddd', 'zerolinewidth':0.1
        }
    plot1['layout']=Layout(title=ptitle
        ,xaxis=xaxis,yaxis=yaxis
        ,plot_bgcolor='white'
        )
    plotly.offline.plot(plot1,filename=filename)
```

Source-code (cont.)

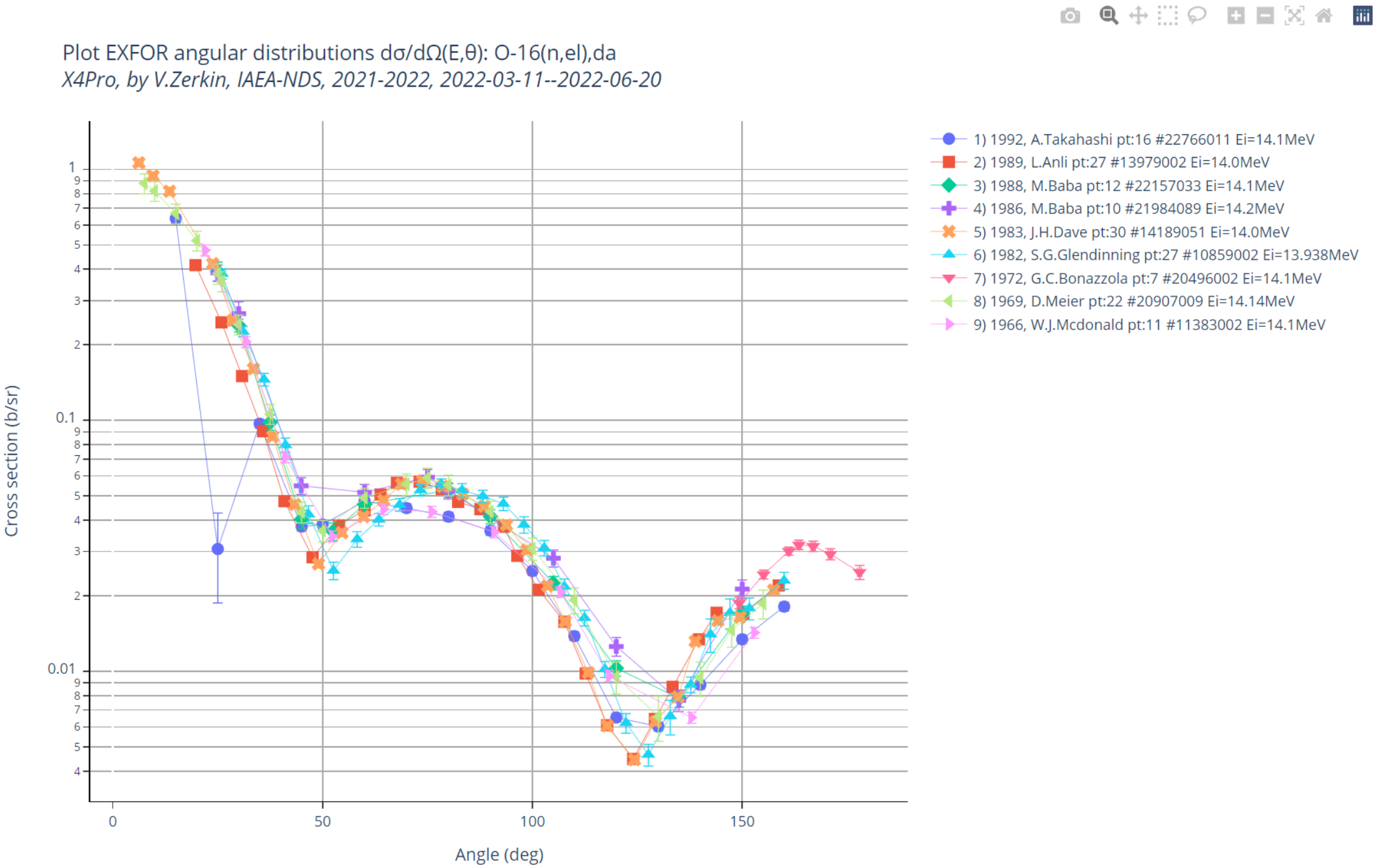
dbConn.py

```python
import sqlite3

dbType='?'

def getConnSQLx4db(fileName='x4sqlite1.db'):
    url='file:'+fileName+'?mode=ro'
    conn=getConnSQLite(url)
    return conn

def getConnSQLite(url):
    global dbType;
    print("___getConnSQLite:",url)
    try:
        conn=sqlite3.connect(url,uri=True)
        conn.row_factory=sqlite3.Row
        dbType='sqlite3'
    except sqlite3.Error as error:
        print("___0___sqlite3.connect.Error:\n",error)
        conn=None
    return conn

def getCursor(conn):
    global dbType;
    cursor=conn.cursor()
    return cursor
```

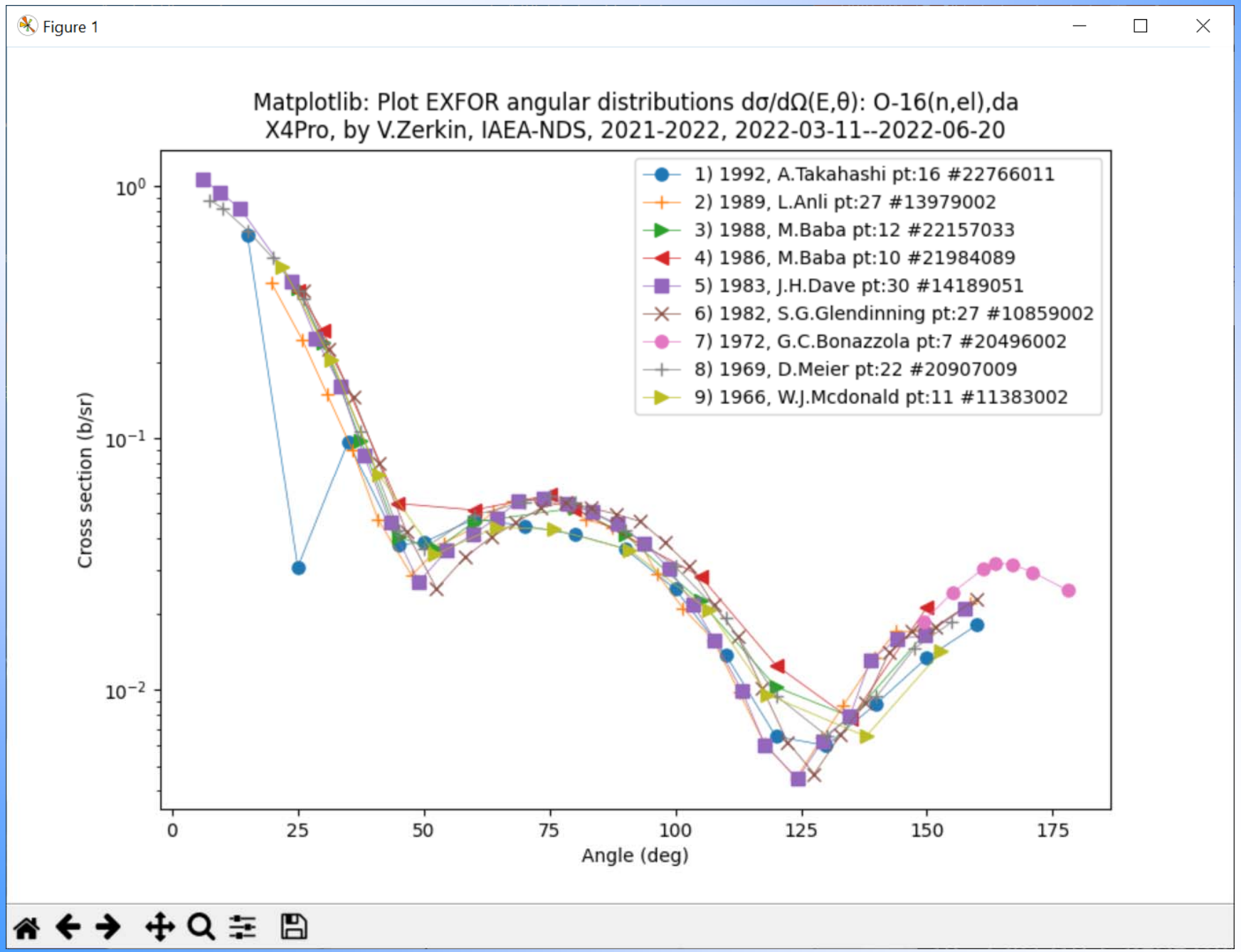
rx4db.py

```python
def getX4SqlSearchCS(target,react):
    print('\n___getX4SqlSearchCS: ['+target+'] ['+react+']')
    sql=str(""
        +"select *                          \n"
        +"from sig1                         \n"
        +"where (Target like '"+target+"')  \n"
        +"  and (Reaction like '"+react+"') \n"
        )
    return sql

def getDatasets(rows):
    lx=len(rows)
    datasets=[]
    ii=0; lastDatasetID=''; lastDataset={}
    fx=1e6; fy=1e-3;
    print('\n___getDatasets from datapoints:',len(rows))
    for row in rows:
        fullCode=row['fullCode']; DatasetID=row['DatasetID']; iPoint=row['iPoint']
        YearRef1=row['YearRef1']; Author1Ini=row['Author1Ini']; Author1=row['Author1'];
        xx=row['En'];  yy=row['Sig'];  dyy=row['dSig'];  dxx=row['dEn']
        if xx is None: continue;
        if yy is None: continue;
        if Author1Ini is not None: Author1=Author1Ini+Author1
        if DatasetID!=lastDatasetID:
            lastDataset={}
            lastDataset['DatasetID']=DatasetID
            lastDataset['Reacode']=fullCode
            lastDataset['x4lbl']=str(YearRef1)+', '+str(Author1)
            x=[];     lastDataset['x']=x
            y=[];     lastDataset['y']=y
            dy=[];    lastDataset['dy']=dy
            dx=[];    lastDataset['dx']=dx
            datasets.append(lastDataset);
            lastDatasetID=DatasetID
            print('DS:'+str(len(datasets))+') '+str(fullCode)+' #'+str(DatasetID)+' '+str(Y
        xx=float(xx)/fx; xx=round(xx,7)
        yy=float(yy)/fy; yy=round(yy,7)
        if dyy is not None: dyy=float(dyy)/fy; dyy=round(dyy,7)
        if dxx is not None: dxx=float(dxx)/fx; dxx=round(dxx,7)
        x.append(xx);
        y.append(yy);
        dy.append(dyy)
        dx.append(dxx)
        ii+=1
        print(' pt:'+str(ii)+'/'+str(lx)+') '+str(fullCode)+' '+str(DatasetID)+' '+str(Year
    return datasets
```

Angular distributions (DA) using Plotly plotting package



Angular distributions (DA) using Matplotlib plotting package



Plotly to Matplotlib: source-code modifications

```
import os
import sys
sys.path.append('./')
sys.path.append('../')
import dbConn
from rx5da        import *
#from exfor2plot  import *
from exfor2plot2 import *

print("Program: da0an2.py, ver. 2022-06-21")
print("Author:  V.Zerkin, IAEA-NDS, Vienna, 2021-2022")
print("Purpose: Retrieve and plot EXFOR angular distribution\n")
```

#Main code change one line

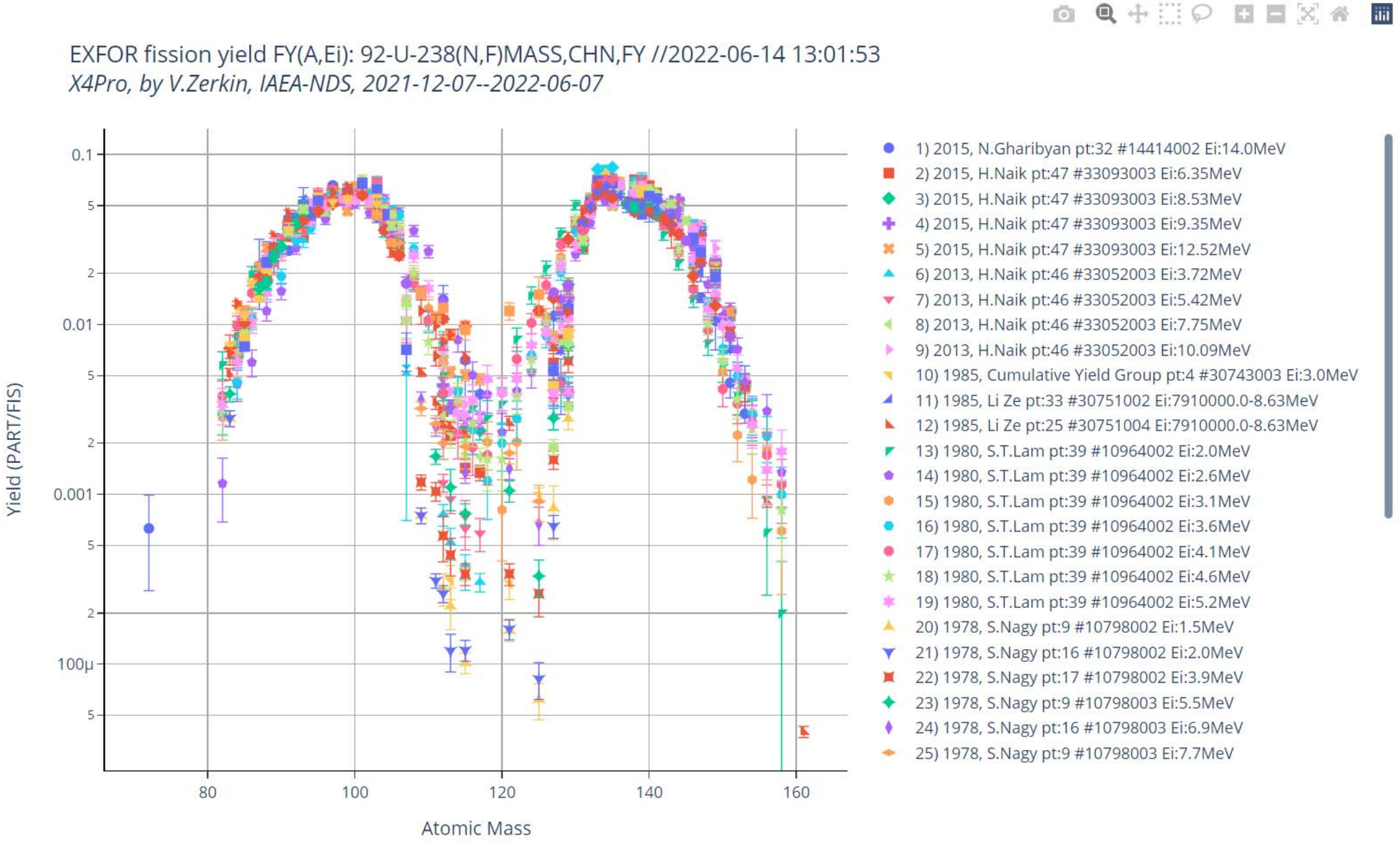
The rest of main code and other files remain the same

New file: exfor2plot2.py

```
import matplotlib.pyplot as plt

#___________________Preparing EXFOR data for plot___________________
def prepareExforDataForPlot(datasets,msize=8,groupReac=False,lines=False,lwidth=0):
    ldata=len(datasets)
    markers=["o", "+", ">", "<", "s", "x"]
    data1=[]; ii=0
    for dataset in datasets:
        mrk=markers[ii%6]
        lbl=str(ii+1)+') '+dataset['x4lbl']+' pt:'+str(len(dataset['x'])) \
                +' #'+dataset['DatasetID']
        plt.plot(dataset['x'],dataset['y'], markersize=msize/1.5, marker=mrk
        ,linestyle='-',linewidth=lwidth
        ,label=lbl)
        ii+=1
        print('Plot:'+str(ii)+'/'+str(ldata)+')'+' #'+str(dataset['DatasetID'])+' '
    return data1

#___________________Plot data from EXFOR and ENDF___________________
def myOfflinePlot(data1,ptitle,xtitle,ytitle
    ,xtype='linear',ytype='linear',filename='temp-plot'):

    plt.title('Matplotlib: '+ptitle)
    plt.xlabel(xtitle)
    plt.ylabel(ytitle)
    plt.xscale(xtype)
    plt.yscale(ytype)

    plt.gcf().set_size_inches(9,6)

    plt.legend()
    #plt.savefig(filename+'.png')
    plt.savefig(filename+'.pdf')
    plt.show()
```

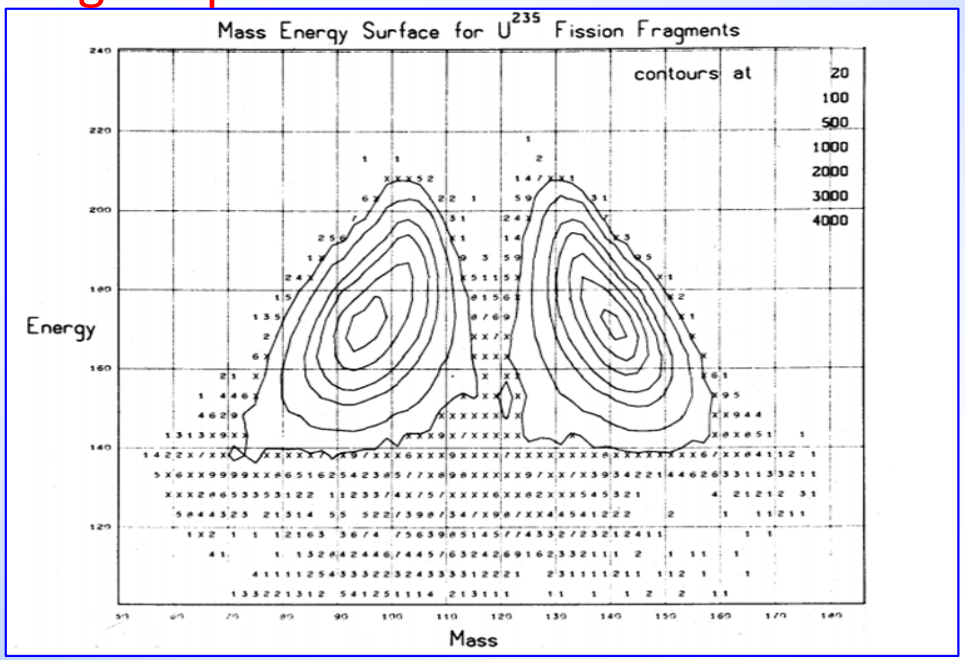
Fission-product yield from EXFOR (FY)

92-U-238(N,F)MASS,CHN,FY : Total chain yield of fission products



Example of “Native” EXFOR plotting (MASS-TKE)

Original publication:



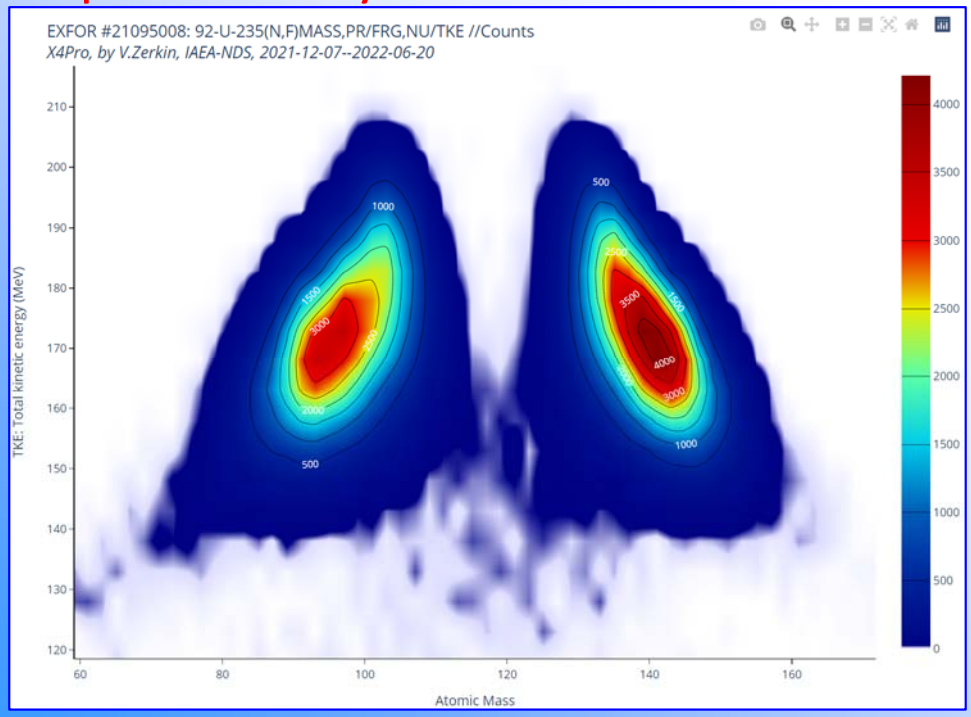
EXFOR: #21095008

92-U-235(N,F)MASS,PR/FRG,NU/TKE

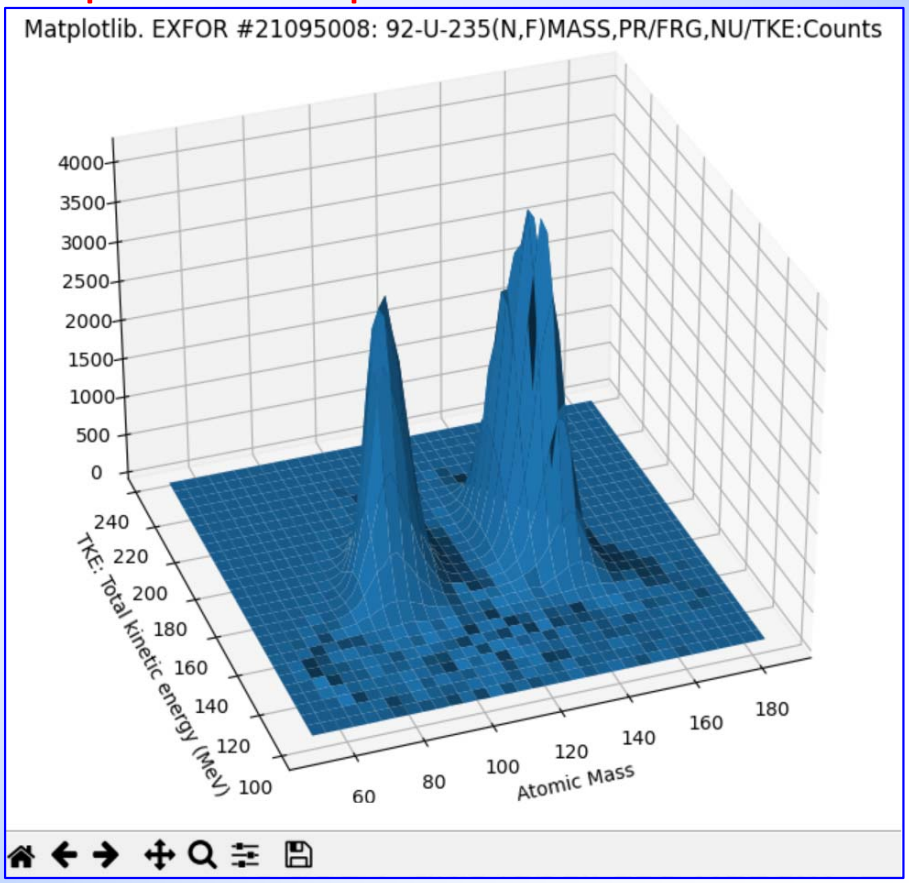
Mass-Energy distribution for both fission fragments

X: MASS(NO-DIM):Atomic mass of nuclide
Y: E(MEV):Energy of outgoing particle, lab. system
Z: MISC(NO-DIM):Number of events detected

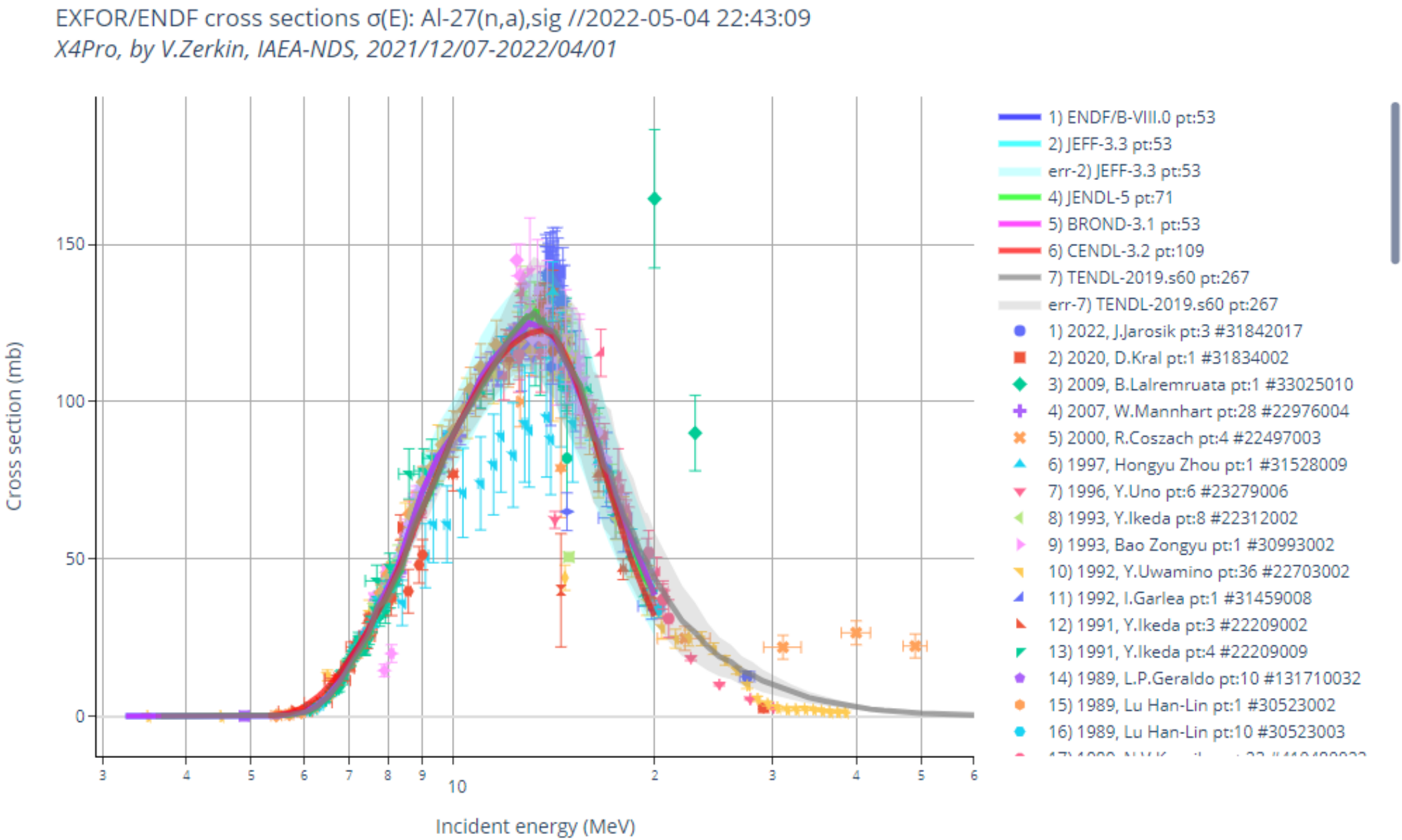
X4pro → Plotly:



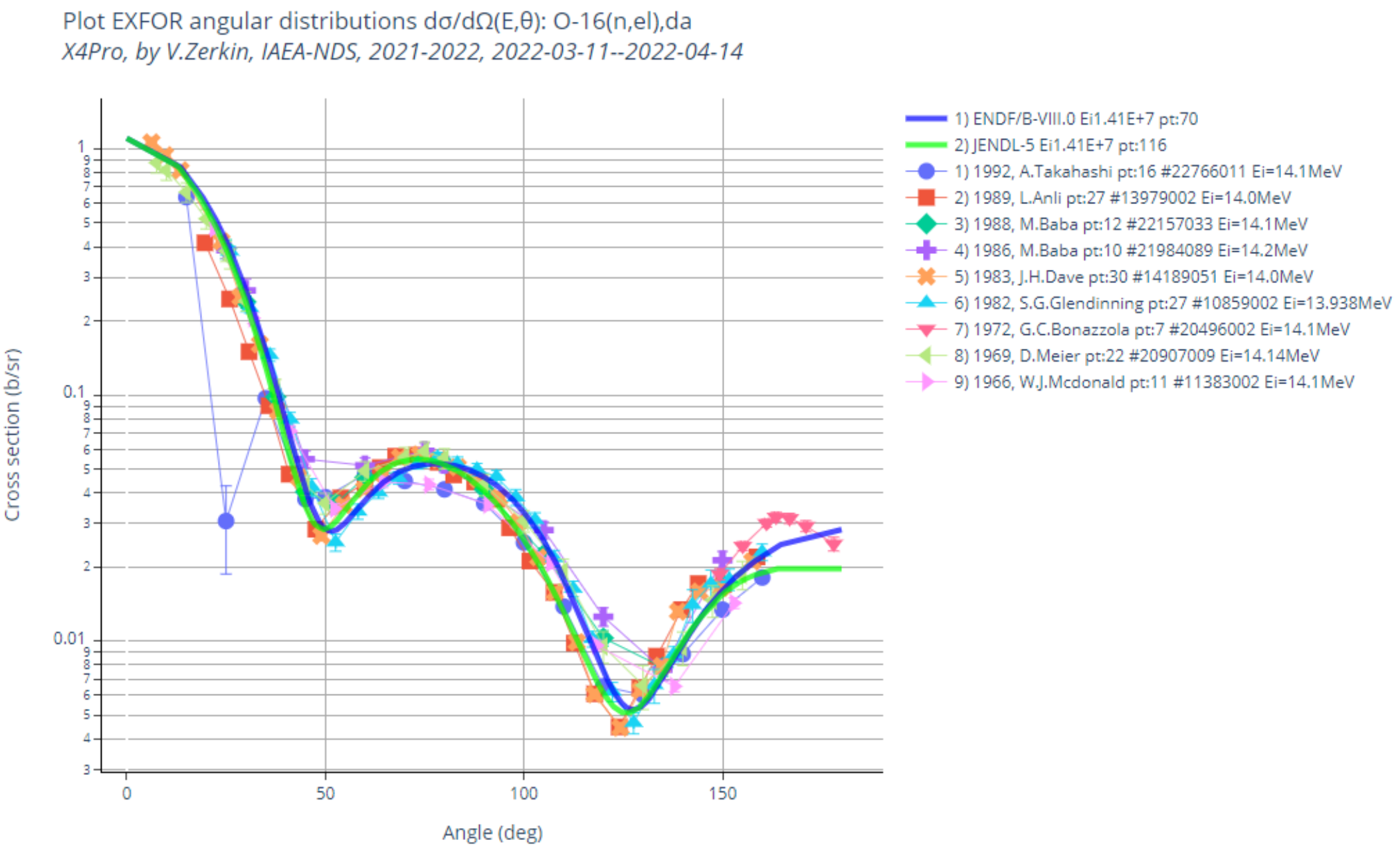
X4pro → Matplotlib:



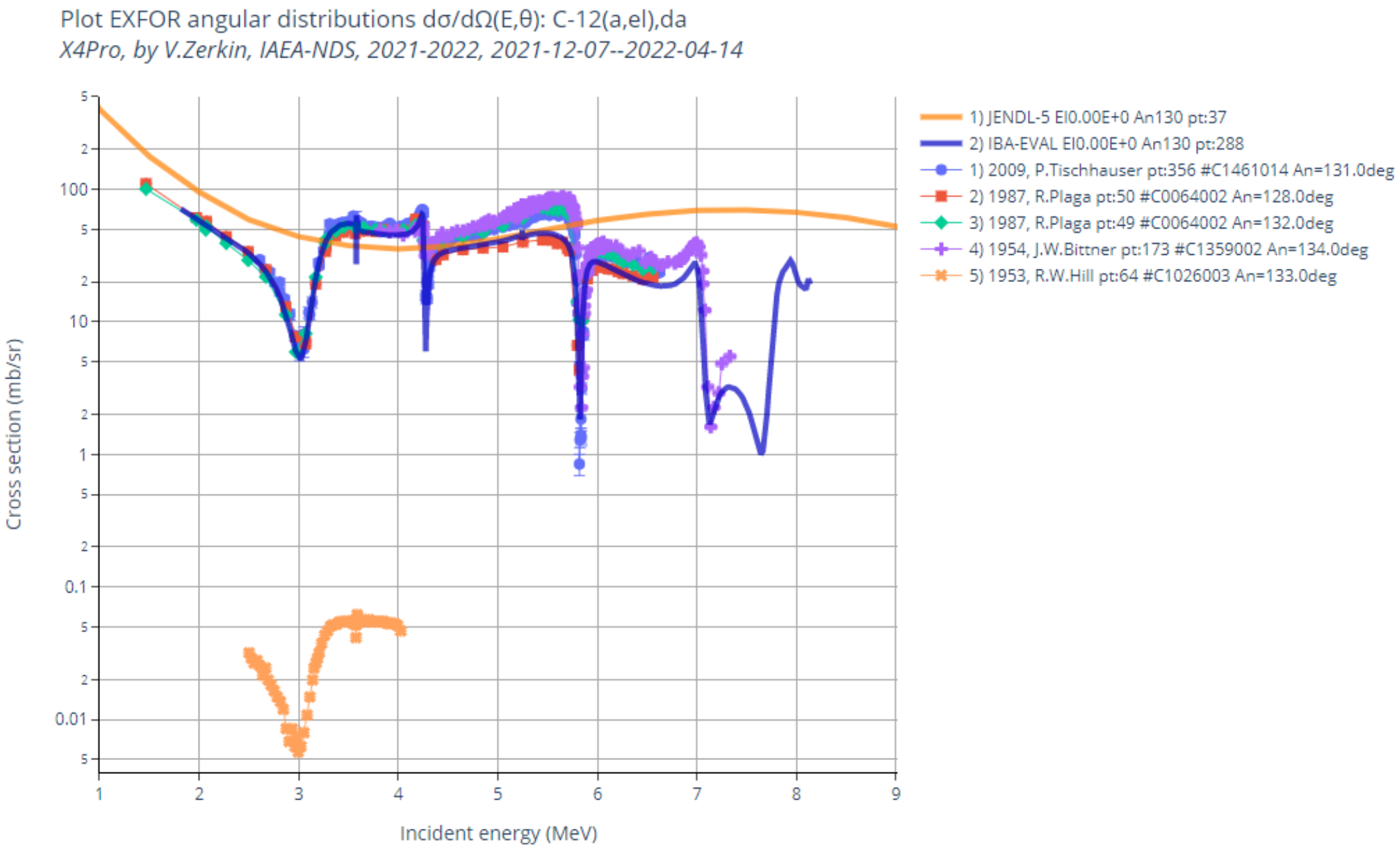
Cross sections (SIG+MF3)



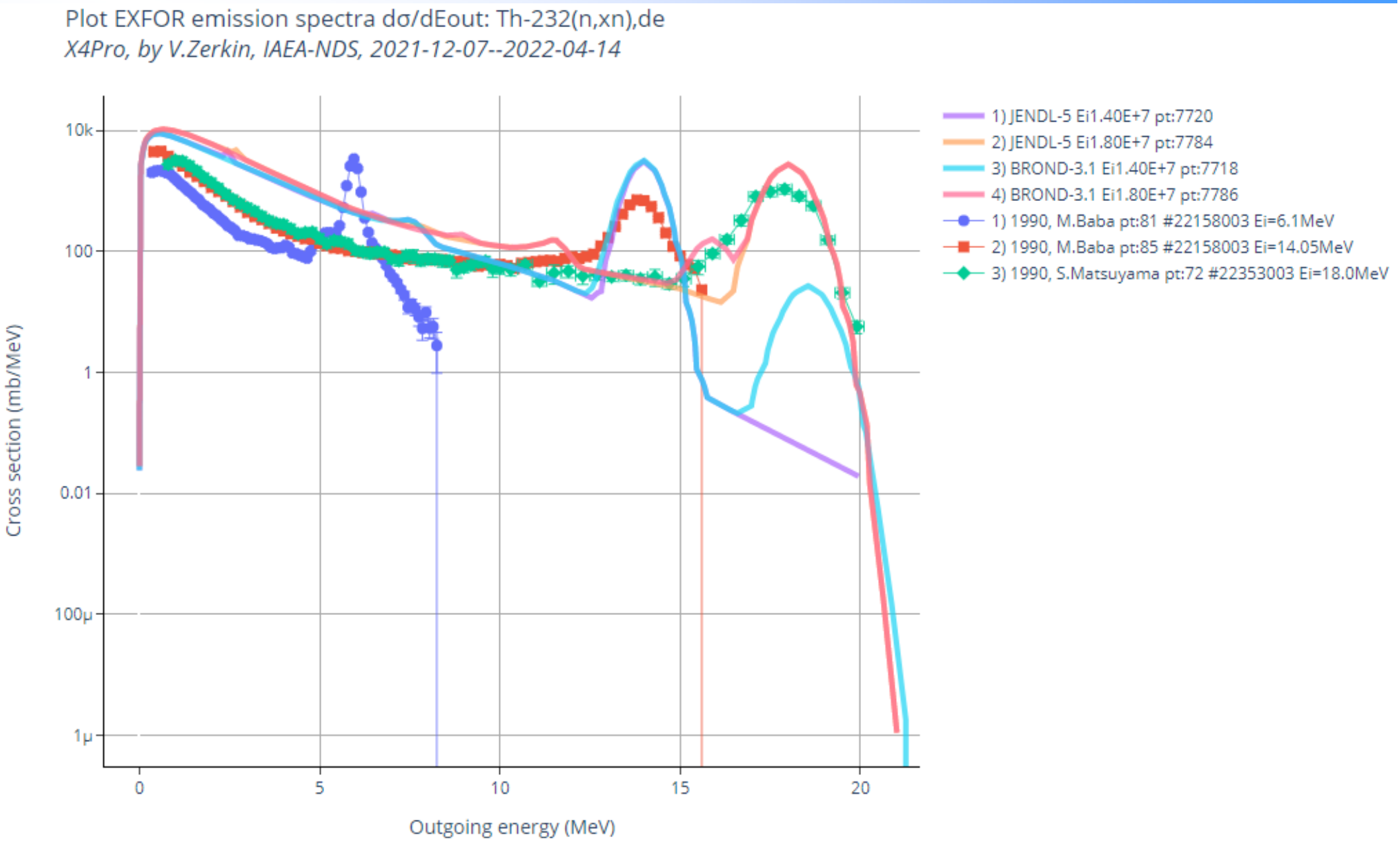
Angular distributions (DA+MF4)



Angular distributions (DA+MF4)



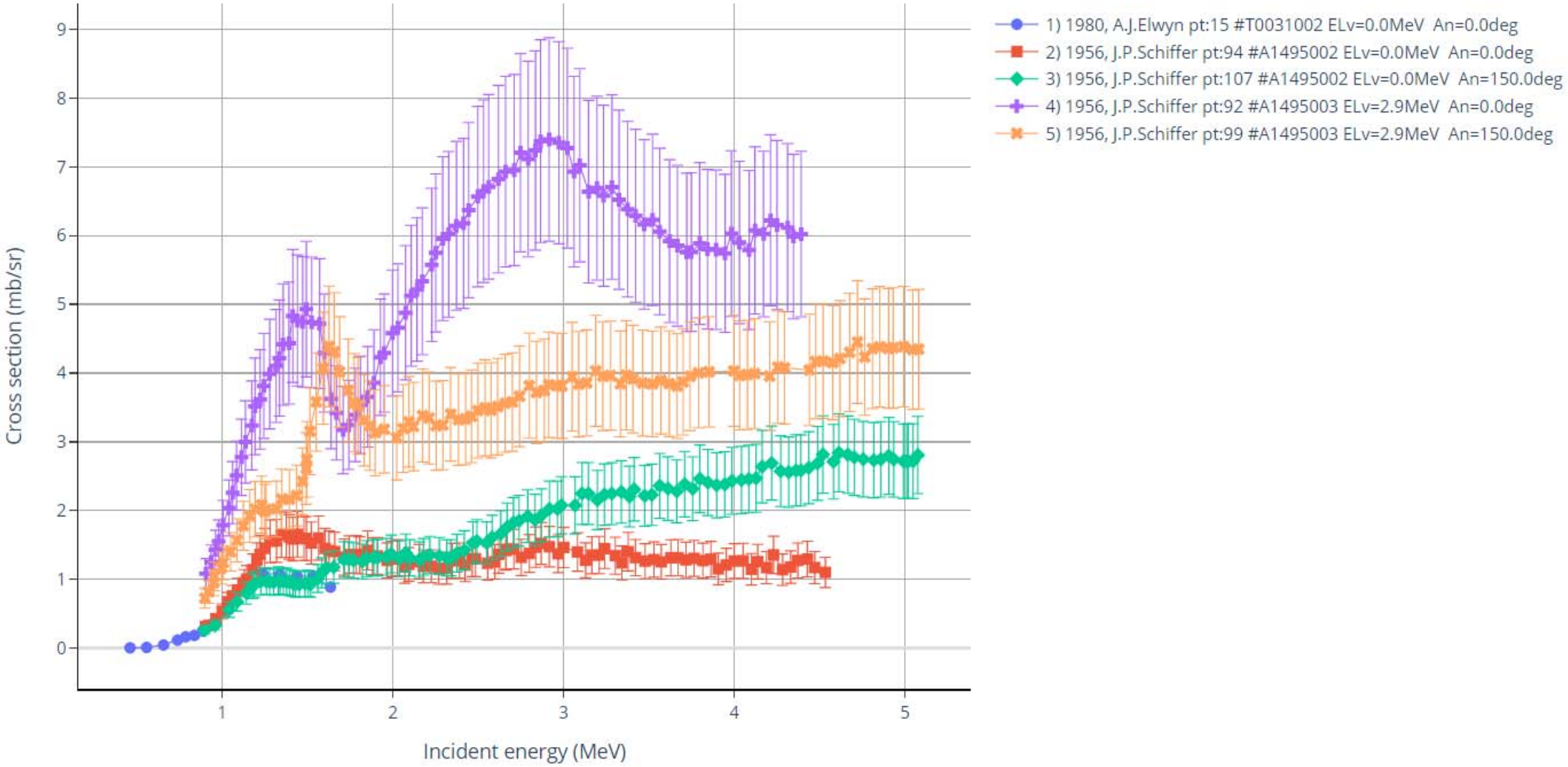
Emission spectra (DE+MF5)



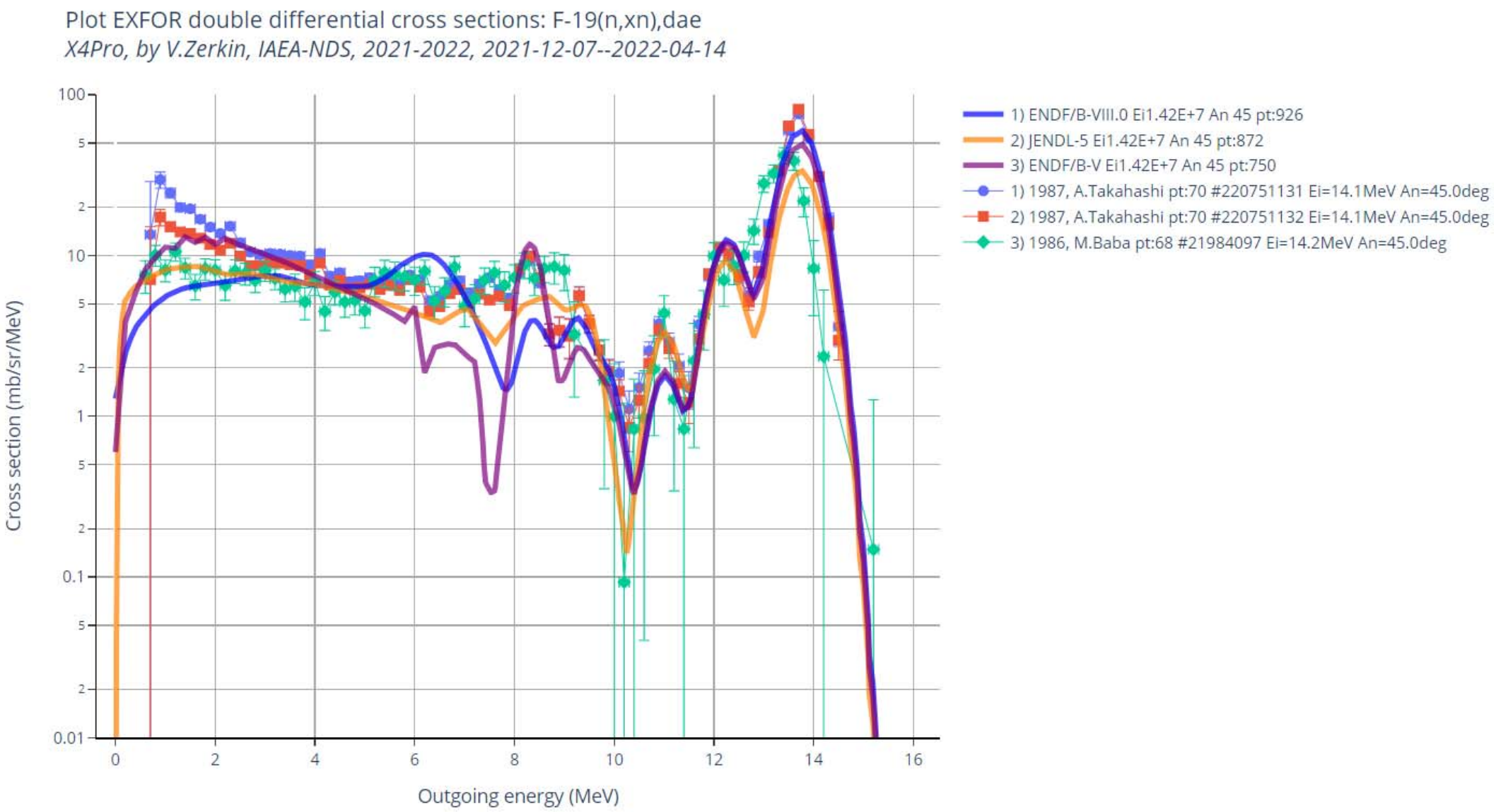
Angular distribution (partial:DAP)

Plot EXFOR angular distributions dσ/dΩ(E,θ): Li-6(he3,p)par,da

X4Pro, by V.Zerkin, IAEA-NDS, 2021-2022, 2022-03-24--2022-04-14



Double differential cross sections (MF6)



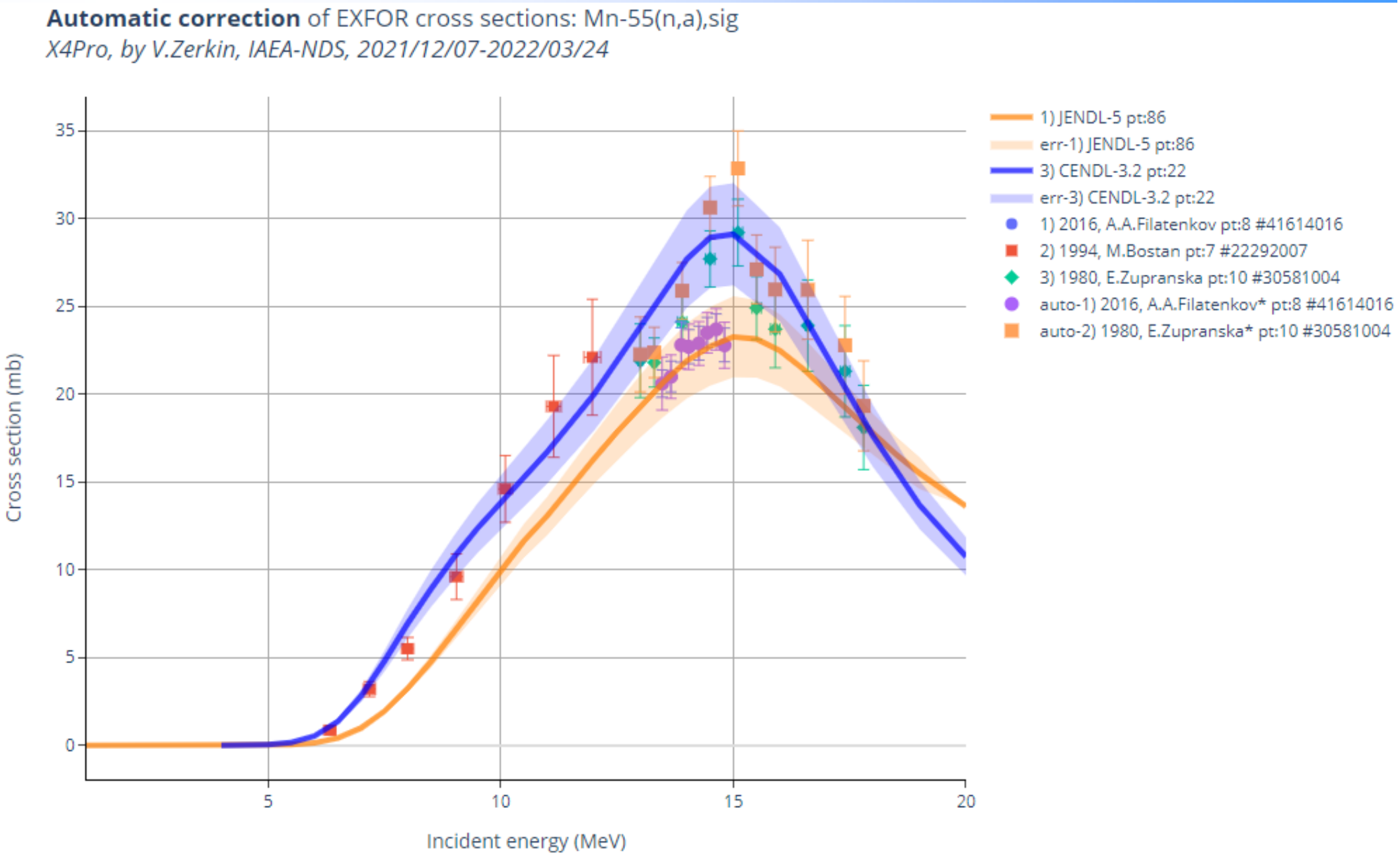
Double differential cross sections in Fortran

```
Program: dae1e2.f (ver.2022-04-26)
by V.Zerkin, IAEA-NDS, 2021-2022
Running: 2022-05-04,22:46:00

Open database: ../../x4sqlite1.db ierr=           0

SQL command:
select * from dae1
where (Target like 'F-19')
and (Reaction like 'n,x')
and (outParticles like '%[n]%')
order by fullCode,
 YearRef1 desc,DatasetID,
 En,An,Eout,iPoint

Operation done successfully:        3285  points

Read data:         3285  points
---new trace--- 220751131 Ei=   14100000.0      An=   15.0000000      1987,Takahashi
          1          71 220751131 1987,Takahashi    Eo=   1100000.00      XS=   2.99000007E-08
          2          70 220751131 1987,Takahashi    Eo=   1300000.00      XS=   2.89000006E-08
          3          69 220751131 1987,Takahashi    Eo=   1500000.00      XS=   3.03999990E-08
          4          68 220751131 1987,Takahashi    Eo=   1700000.00      XS=   2.64000004E-08
          5          67 220751131 1987,Takahashi    Eo=   1900000.00      XS=   2.03999999E-08
          6          66 220751131 1987,Takahashi    Eo=   2100000.00      XS=   1.43000003E-08
          7          65 220751131 1987,Takahashi    Eo=   2300000.00      XS=   1.40000003E-08
          8          64 220751131 1987,Takahashi    Eo=   2500000.00      XS=   1.22000001E-08
          9          63 220751131 1987,Takahashi    Eo=   2700000.00      XS=   1.12000000E-08
         10          62 220751131 1987,Takahashi    Eo=   2900000.00      XS=   1.11000000E-08
         11          61 220751131 1987,Takahashi    Eo=   3100000.00      XS=   1.58999995E-08
         12          60 220751131 1987,Takahashi    Eo=   3300000.00      XS=   1.17000001E-08
         13          59 220751131 1987,Takahashi    Eo=   3500000.00      XS=   6.49000009E-09
         14          58 220751131 1987,Takahashi    Eo=   3700000.00      XS=   6.11000006E-09
         15          57 220751131 1987,Takahashi    Eo=   3900000.00      XS=   1.12000000E-08
         16          56 220751131 1987,Takahashi    Eo=   4100000.00      XS=   1.60999996E-08
         17          55 220751131 1987,Takahashi    Eo=   4300000.00      XS=   1.09000000E-08
         18          54 220751131 1987,Takahashi    Eo=   4500000.00      XS=   1.39000003E-08
         19          53 220751131 1987,Takahashi    Eo=   4700000.00      XS=   6.50000009E-09
         20          52 220751131 1987,Takahashi    Eo=   4900000.00      XS=   6.78000012E-09
         21          51 220751131 1987,Takahashi    Eo=   5100000.00      XS=   6.04000006E-09
```

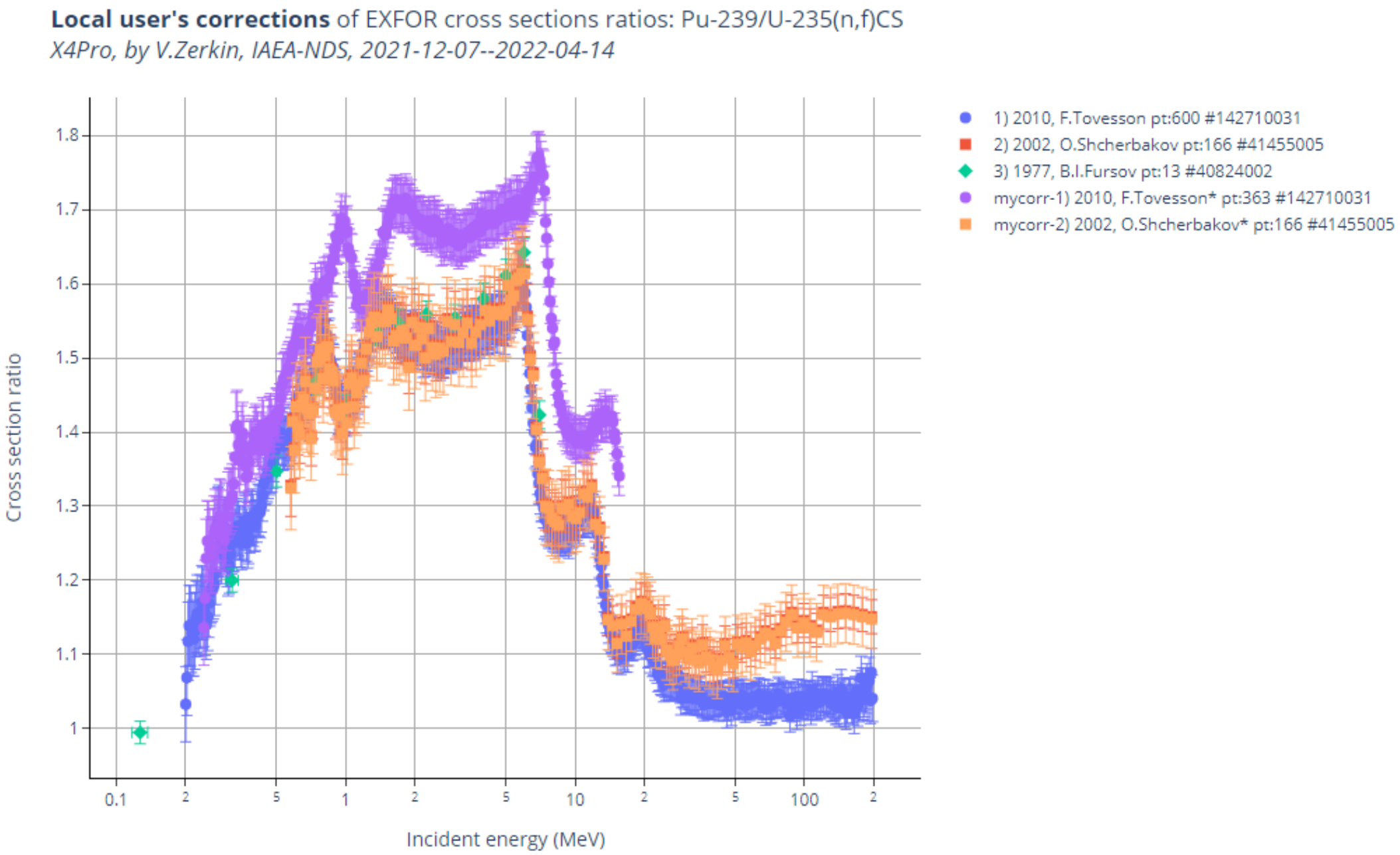
Cross sections (MF3) in Fortran

```
Program: sig1.f (ver.2022-04-26)
by V.Zerkin, IAEA-NDS, 2021-2022
Running: 2022-05-04,22:46:03

Open database: ../../x4sqlite1.db ierr=           0

SQL command:
select * from sig1 where (Target like 'Mn-55') and (MT= 107)
Operation done successfully:          82  points
Read data points:          82
---Dataset--- 25-MN-55(N,A)23-V-52,,SIG #33151007  2020,Pasha
          1           0 33151007  2020,Pasha        MF=           3  MT=         107  En=   14540000.0      XS=   2.75999997E-02
---Dataset--- 25-MN-55(N,A)23-V-52,,SIG #41614016  2016,Filatenkov
          2           0 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   13470000.0      XS=   2.06000004E-02
          3           1 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   13660000.0      XS=   2.09999997E-02
          4           2 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   13880000.0      XS=   2.28000004E-02
          5           3 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   14040000.0      XS=   2.27000006E-02
          6           4 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   14260000.0      XS=   2.29000002E-02
          7           5 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   14440000.0      XS=   2.34999992E-02
          8           6 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   14630000.0      XS=   2.37000007E-02
          9           7 41614016  2016,Filatenkov   MF=           3  MT=         107  En=   14810000.0      XS=   2.28000004E-02
---Dataset--- 25-MN-55(N,A)23-V-52,,SIG #32701002  2012,Yanbin Zhang
         10           0 32701002  2012,Yanbin Zhang MF=           3  MT=         107  En=   14100000.0      XS=   2.12999992E-02
         11           1 32701002  2012,Yanbin Zhang MF=           3  MT=         107  En=   14700000.0      XS=   2.38000005E-02
---Dataset--- 25-MN-55(N,A)23-V-52,,SIG #22414016  2000,Fessler
         12           0 22414016  2000,Fessler      MF=           3  MT=         107  En=   16065000.0      XS=   2.22200006E-02
         13           1 22414016  2000,Fessler      MF=           3  MT=         107  En=   17028000.0      XS=   1.96400005E-02
         14           2 22414016  2000,Fessler      MF=           3  MT=         107  En=   17777000.0      XS=   1.76400002E-02
         15           3 22414016  2000,Fessler      MF=           3  MT=         107  En=   19101000.0      XS=   1.33300005E-02
         16           4 22414016  2000,Fessler      MF=           3  MT=         107  En=   20313000.0      XS=   8.44000001E-03
---Dataset--- 25-MN-55(N,A)23-V-52,,SIG #41240011  1999,Filatenkov
         17           0 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   13470000.0      XS=   2.06000004E-02
         18           1 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   13660000.0      XS=   2.08999999E-02
         19           2 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   13880000.0      XS=   2.26000007E-02
         20           3 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   14040000.0      XS=   2.27000006E-02
         21           4 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   14260000.0      XS=   2.29000002E-02
         22           5 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   14440000.0      XS=   2.32999995E-02
         23           6 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   14630000.0      XS=   2.34999992E-02
         24           7 41240011  1999,Filatenkov   MF=           3  MT=         107  En=   14810000.0      XS=   2.28000004E-02
---Dataset--- 25-MN-55(N,A)23-V-52,,SIG #22292007  1994,Bostan
         25           0 22292007  1994,Bostan       MF=           3  MT=         107  En=   6330000.00      XS=   8.59999971E-04
         26           1 22292007  1994,Bostan       MF=           3  MT=         107  En=   7180000.00      XS=   3.19999992E-03
```

Automatic renormalization in Python



User's modifications in Python

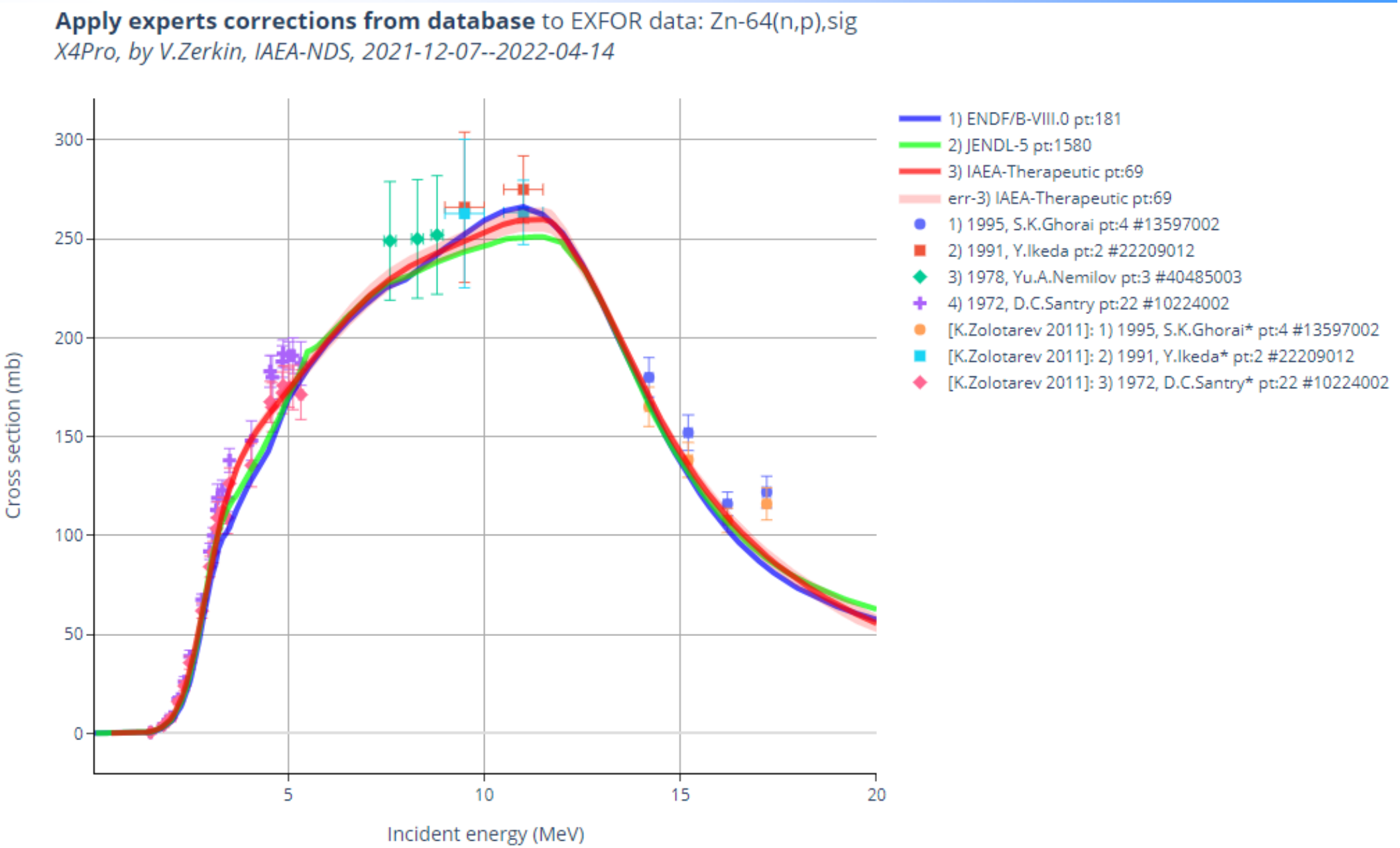


User's modifications in Python

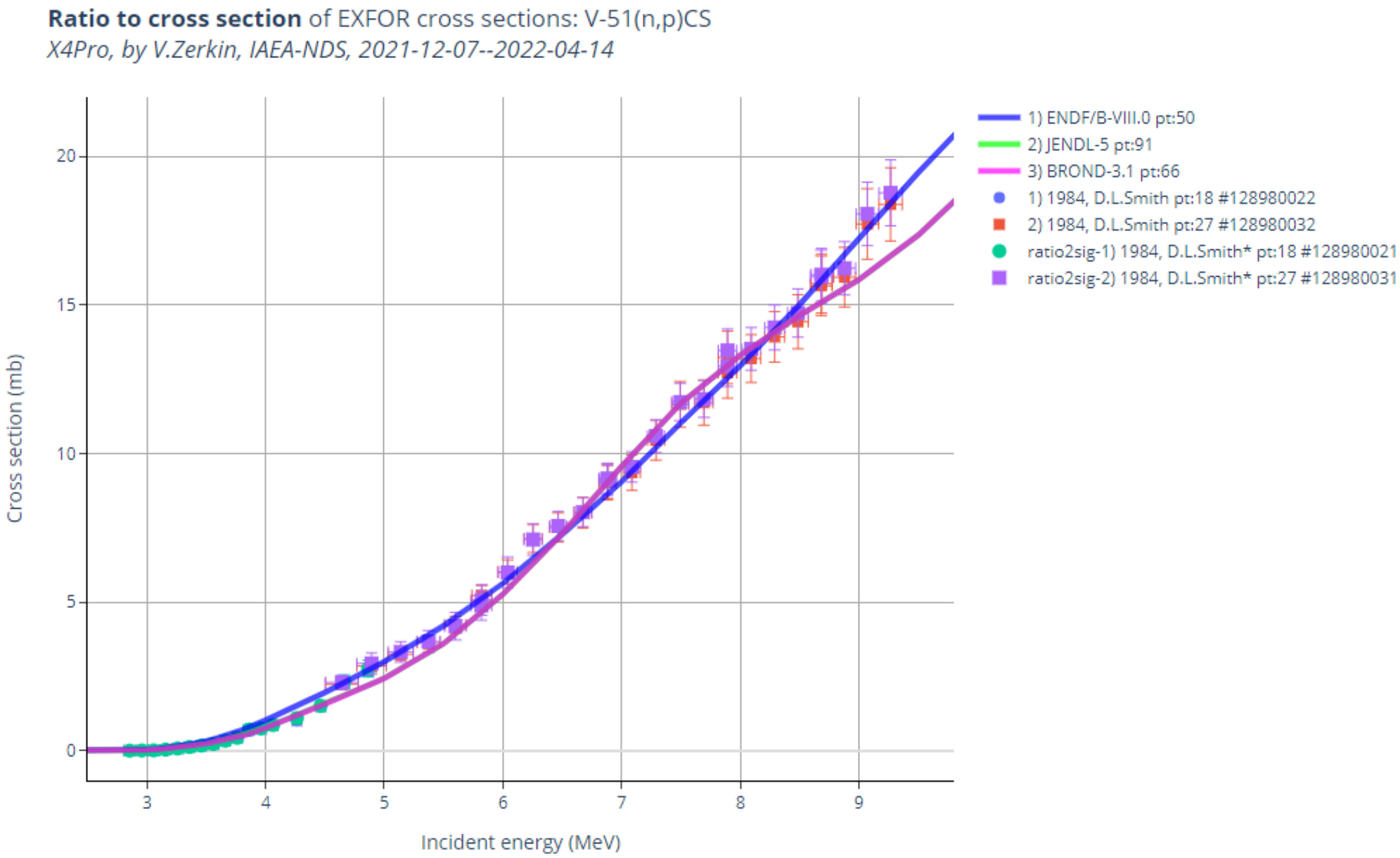
```
def fcorr_142710031(row):      #2010,F.Tovesson
    #Reaction: (94-PU-239(N,F),,SIG)/(92-U-235(N,F),,SIG)
    ene=row['En']
    if ene>13e6:
        print("___________fcorr_142710031:"+' En='+str(row['En']/1e6)+'MeV'+' -REJECTED-')
        return -1              #data above 13 MeV rejected in Neutron Standard evaluation (2017)
    row['En']=row['En']*1.2    #just as example
    row['y']=row['y']*1.1      #just as example
    #print("___________fcorr_142710031:"+' En='+str(row['En']/1e6)+'MeV'+' -ACCEPTED-')
    return 1                   #updated == accepted
```

```
def fcorr_41455005(row):       #x4u:20170724 #2002,O.Shcherbakov
    # REACTION    ((94-PU-239(N,F),,SIG)/(92-U-235(N,F),,SIG))
    # MONITOR     ((94-PU-239(N,F),,SIG)/(92-U-235(N,F),,SIG))
    # MONIT-REF   (,,3,JENDL-3.2,,1994)
    # COMMENT     Of Authors.
    #             The fission cross-section ratio normalization
    #             has been done in the 1.75-4.0 MeV energy interval
    #             using data of JENDL-3.2.
    y=row['y']
    dy=row['dy']
    dy=dy/y;          #convert abs. uncertainty in cs-ratio to rel. uncertainty
    a0=1.535;         #used ratio normalization factor (using data of JENDL-3.2), E:1.75-4.0 MeV
    c0=1.668/100;     #1.535 +-1.668% (this uncertainty is not included to error analysis)
    a1=1.5393;        #ratio normalization factor (using data of ENDF/B-VIII.0), E:1.75-4.0 MeV
    c1=2.82/100;      #1.5393 +-2.82% (uncertainty should be added)
    fc=a0/a1;         #total correction factor
    y=y*fc;           #correction exp. cs
    dy=dy**2+c1**2; #calc. new quadrature of total uncertainty
    dy=dy**0.5*y;     #back to absolute uncertainty
    print("___________fcorr_41455005:"+' En='+str(row['En'])
          +'          y0='+str(row['y'])+'          y1='+str(round(y,5))+'
          Fc='+str(round(fc,5))
          +'          dy0='+str(row['dy'])+'          dy1='+str(round(dy,5)))
    row['y']=y        #save y
    row['dy']=dy      #save dy
    return 1          #updated
```

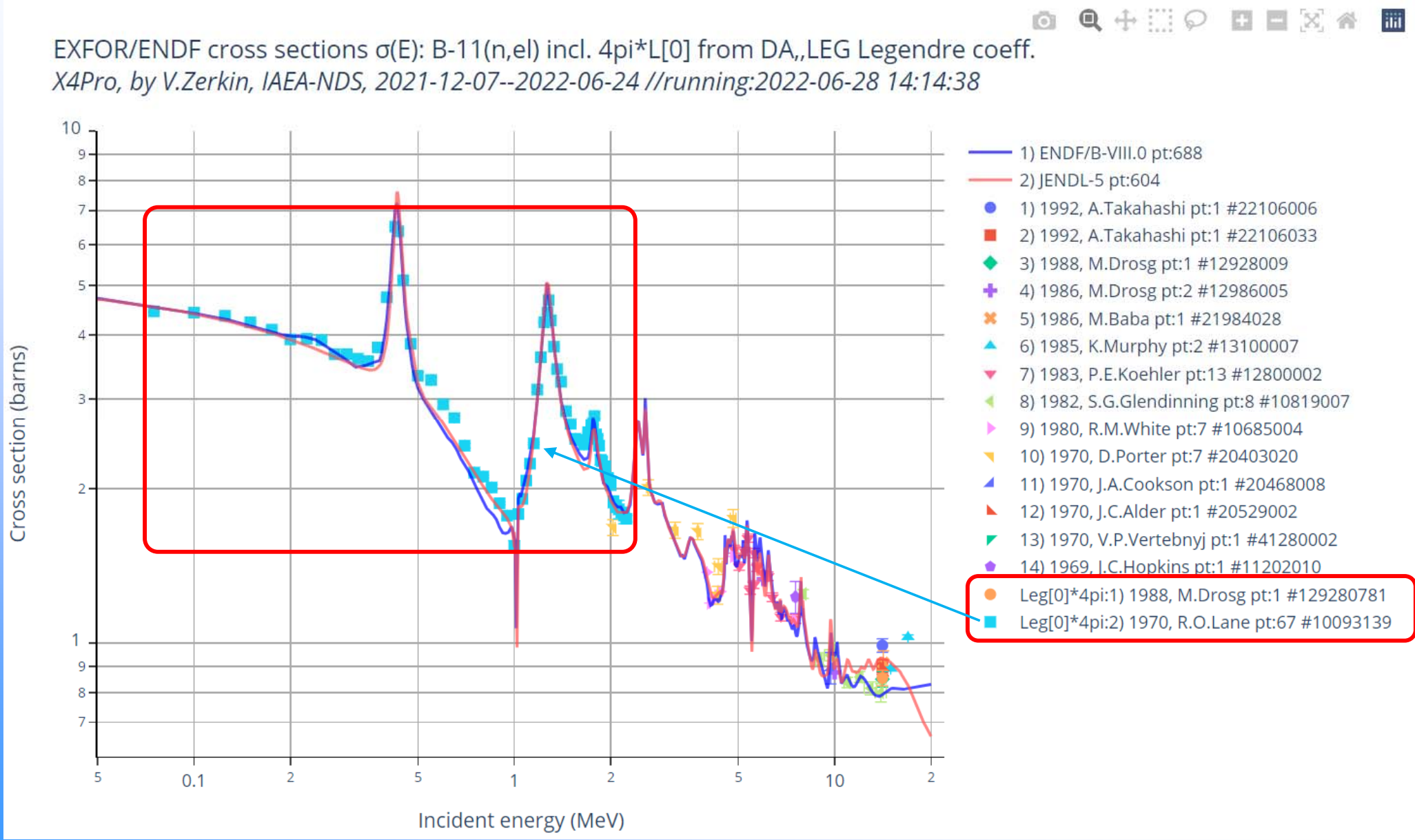
Experts' modifications in Python



Ratios to cross sections in Python



Cross section DATA from [,SIG] together with search/filter/calc.: 4π·L(0) from [,DA,,LEG]



Retrieval on JavaScript with GUI/html5

Access local X4Pro/sqlite3 via html/GUI using javascript
/by V.Zerkin, IAEA-NDS, 2022-04-28/

EXFOR database: "../x4sqlite1c.db", 1.46 GiB, ver:2022-03-23

Target: mn-55
Reaction: n,2n
Quantity: cs
OK. Search

SQL

```
select * from sig1
where (Target like 'mn-55')
  and (Reaction like 'n,2n')
```

Execute

SQL-Result

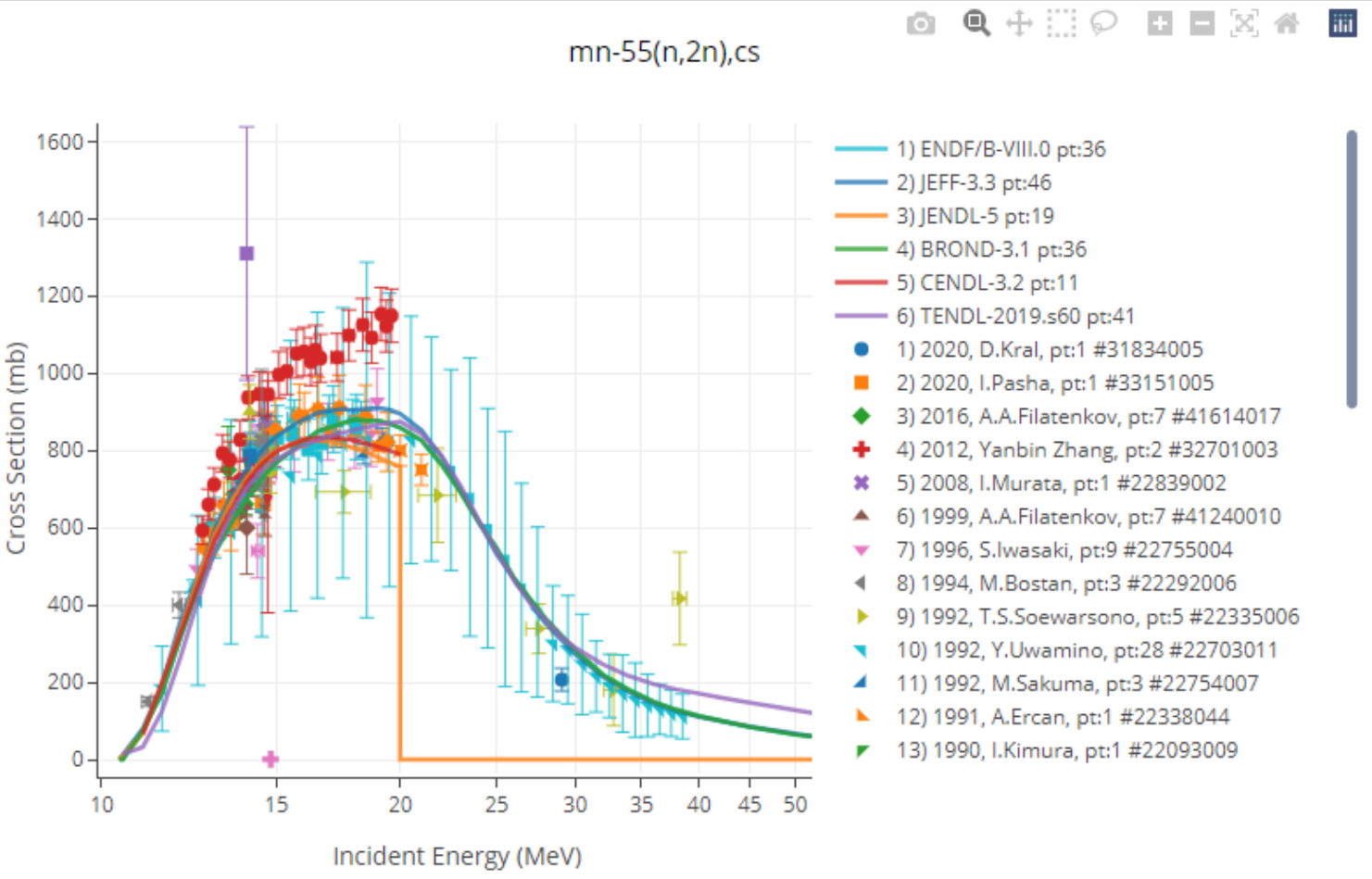
Build your own Web interface working on local PC

Datasets

1) 2020, D.Kral, E:[29.1], pt:1, #31834005
2) 2020, I.Pasha, E:[14.54], pt:1, #33151005
3) 2016, A.A.Filatenkov, E:[13.56-14.78], pt:7, #41614017
4) 2012, Yanbin Zhang, E:[14.1-14.7], pt:2, #32701003
5) 2008, I.Murata, E:[14.2], pt:1, #22839002
6) 1999, A.A.Filatenkov, E:[13.56-14.78], pt:7, #41240010
7) 1996, S.Iwasaki, E:[12.465-18.951], pt:9, #22755004
8) 1994, M.Bostan, E:[11.14-12.85], pt:3, #22292006
9) 1992, T.S.Soewarsono, E:[17.55-38.26], pt:5, #22335006
10) 1992, Y.Uwamino, E:[11.5-38.5], pt:28, #22703011
11) 1992, M.Sakuma, E:[17.226-19.147], pt:3, #22754007
12) 1991, A.Ercan, E:[14.6], pt:1, #22338044
13) 1990, I.Kimura, E:[14.05], pt:1, #22093009
14) 1988, Y.Ikeda, E:[13.35-14.93], pt:8, #22089039
15) 1987, J.W.Meadows, E:[14.74], pt:1, #12969011
16) 1987, L.R.Greenwood, E:[14.5-14.9], pt:5, #12977008
17) 1985, B.M.Bahal, E:[14.7], pt:1, #21936010
18) 1984, M.Berrada, E:[14.6], pt:1, #30805002
19) 1980, Lu Hanlin, E:[14.58], pt:1, #30615002
20) 1980, Lu Hanlin, E:[12.37-18.26], pt:14, #30615003
21) 1979, K.Kayashima, E:[14.6], pt:1, #21300013
22) 1977, G.F.Auchampaugh, E:[14.7-21], pt:7, #12936006
23) 1976, O.Schwerer, E:[14.6], pt:1, #20811006
24) 1975, F.Deak, E:[14.7], pt:1, #30333002
25) 1974, G.N.Maslov, E:[14.6], pt:1, #40136005
26) 1973, J.Araminowicz, E:[14.6], pt:1, #30264011
27) 1971, O.A.Salnikov, E:[14.36], pt:1, #40037133
28) 1969, R.C.Barrall, E:[14.6], pt:1, #10022008
29) 1969, R.C.Barrall, E:[14.8], pt:1, #10031004
30) 1969, M.Bormann, E:[12.99-18.06], pt:8, #20835003
31) 1968, H.Vonach, E:[14.1], pt:1, #21533002
32) 1967, H.O.Menlove, E:[12.7-19.39], pt:10, #11421005
33) 1967, J.Csikai, E:[13.41], pt:1, #30033008
34) 1965, A.Paulsen, E:[12.63-19.59], pt:23, #20378004
35) 1963, B.Granger, E:[14], pt:1, #21514005
36) 1962, R.Wenusch, E:[14], pt:1, #20091003
37) 1961, J.Nix, E:[14.8], pt:1, #11684003
38) 1960, E.Weigold, E:[14.5], pt:1, #31039006
39) 1958, V.J.Ashby, E:[14.1], pt:1, #11632003

Plot

Axis x: ☑ log y: ☐ log Range x: 10 52 y: -36 1647 Size: 800 500 Repaint



Concluding remarks

1. X4Pro: EXFOR relational database extended with
   a) EXFOR data points in original and computational form
   b) data for renormalizing EXFOR data (monitor and decay data)
   c) instructions for data corrections (Python text stored in DB-cells)
2. Implemented in MariaDB and SQLite
3. Advantages of X4Pro
   a) no need in original EXFOR: all info and data can be taken from the database
   b) no need in EXFOR parsers on user side
   c) no need for intermediate (JSON) files: application uses SQL SELECT commands to retrieve data and create objects within reading program
   d) simple for programming on many languages using SQL for data search, filtering, sorting, retrieval, renormalization
4. Status and plans-2022
   a) now: preparing trial version for limited distribution (NDS+)
   b) present on NRDC-2022, propose for testing and feedback
   c) present on ND-2022
   d) continue development
   e) cooperate with SG50 (?)

Preparing trial distribution

1. Part-1. Retrieve data from EXFOR: Python, Fortran

- Python. Retrieve data from X4Pro (SQLite, MariaDB) and plot using Plotly:
  1) CS, 2) DA, 3) DAP, 4) DE, 5) DAE, 6) FY
- Fortran/C. Retrieve data from X4Pro (SQLite):
  1) CS, 2) DAE

2. Part-2. Retrieve data from EXFOR and ENDF: Python

- Retrieve local EXFOR and remote ENDF data (using API from Web in JSON):
  1) CS, 2,3) DA, 3) DAP, 4) DE, 5) DAE

3. Part-3. EXFOR data modification: Python

1) Retrieve cross sections from X4Pro; automatically renormalize data using old monitor and new standard data
2) Retrieve cross sections ratios from X4Pro; correct data using user's algorithms
3) Retrieve cross sections from X4Pro; apply experts corrections from the database
4) Retrieve cross sections from X4Pro; automatically renormalize data and uncertainties using old/new monitor data
5) Retrieve cross sections ratios from EXFOR database, calculate cross section data and uncertainties using latest standards

4. Part-4. Personal EXFOR-ENDF retrieval system: Html+Javascript

Currently no plans for distribution

For discussion

Reminder. https://www-nds.iaea.org/nrdc/
NRDC objective: The primary goal of the Network is the dissemination of nuclear reaction data and associated documentation to users.

NRDC support for X4Pro

1. Trial version for testing
2. Potential users (Python):
   a) Useful subject for EXFOR Workshop (NRDC interested?)
   b) Useful for students (?): Vladimir, Olena, Sophia
3. Distribution parts: 1 or 1 + 2 or all 3 in one package?
4. Licence type: needed? Or just mention: working materials

NRDC policy for off-line EXFOR data distribution

1. Version of Master file (date + web-link to source)
2. Acknowledgment database source: NRDC, version of dictionaries if used
3. Licence type (?)
4. Authorized (trusted) distributors – list on NRDC site
5. etc.

Thank you.