Start using X4Pro – universal, fully relational, multi-model EXFOR database

Viktor Zerkin
International Atomic Energy Agency, Nuclear Data Section

WPEC SG50, Web Meeting, 30 March 2023

Part I

Introducing X4Pro

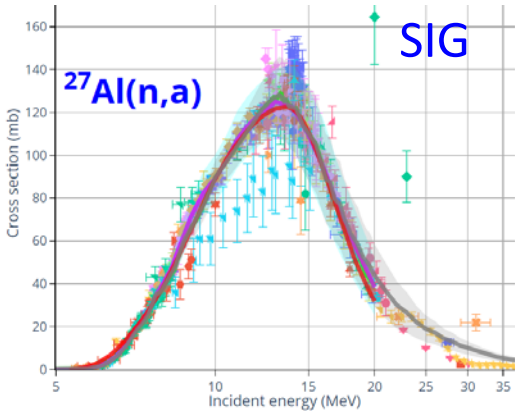
X4Pro concept

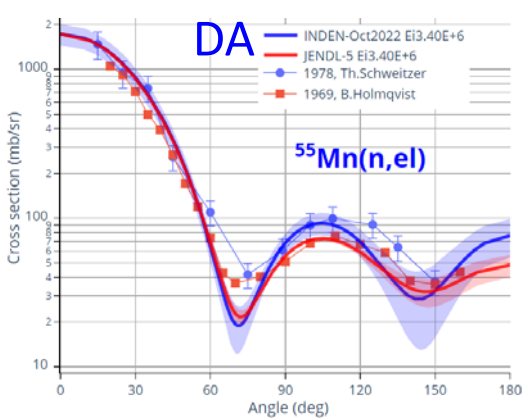
1. Relational. *Traditional SQL database storing data in tables. Continues and extends "EXFOR-Relational" project, 2000-2023.*
2. Fully relational. *All meta-data and numerical values presented in tables and accessible by SQL commands.*
3. Multi-model. *Table cells contain single values and also many values as semi-structured data in JSON. Note: "JSON" data type supported by modern relational DBMS via functions extending SQL commands (since ~2015).*
4. Universal. *Flexible SQL search, filtering, sorting allows to produce any data hierarchy on the fly; data in original and computational forms; includes monitor and decay data to be used for automatic renormalization, instructions for data modifications from experts. Implemented in MariaDB and SQLite, tested on Windows, Linux, MacOS. Can be used as starting point for other projects: from students with homework to professionals with advanced tasks.*
5. Powerful. *Oriented to programming users: they can do much more then using Web and GUI interfaces with fixed functionality.*
6. Rational. *No need EXFOR parsers for new languages. Can be used by programs on any language supporting SQL: Python, Java, JavaScript, Fortran, Perl, etc.*

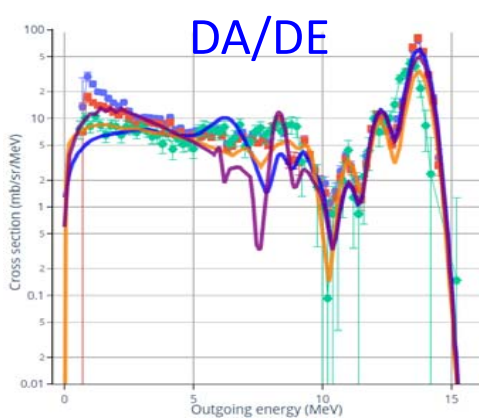
X4Pro offers

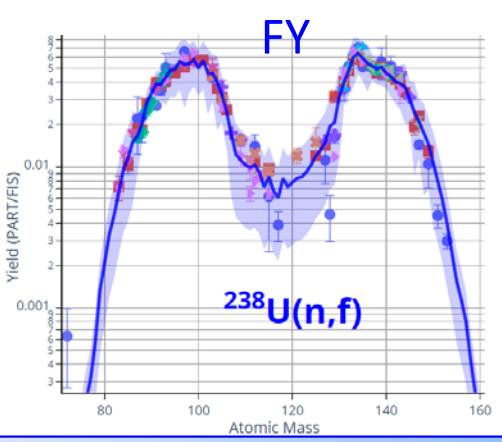
1. EXFOR data without EXFOR format.
   - *All data points, data for corrections, meta-data are provided in the database.*
   - *No need in original EXFOR for end-users.*
   - *No need in new EXFOR parsers/converters for new programming languages.*
   - *No need in intermediate files and formats with fixed structure (C5, XML, JSON).*
   - *Simple for programming on any language supporting SQL for data search, filtering, sorting, retrieval, renormalization.*
2. Local EXFOR database for programmatic access.
   Providing data for various tasks required automatization, "non-so-general" to be implemented under Web/GUI interface proposed by data centres; packages required access to all experimental data at once; evaluation software required data corrections
3. Examples.
   24 examples of Fortran and Python programs provided with source code (MIT licence) and "run-me" scripts retrieving and plotting data from local X4Pro and remote ENDF database via Web-API interface.
4. X5JSON.
   Comprehensive EXFOR data presentation in JSON form.
   Can be used for creating another systems built on JSON objects (e.g. NoSQL databases).
   Example of building CouchDB is provided.

X4Pro examples: EXFOR + ENDF/Web

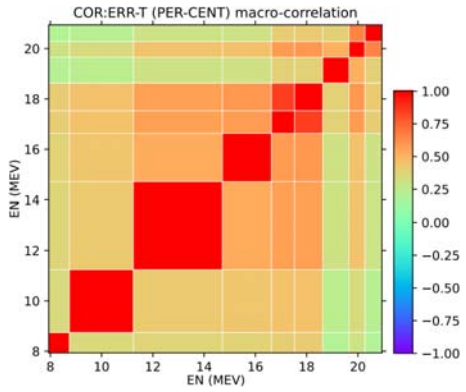


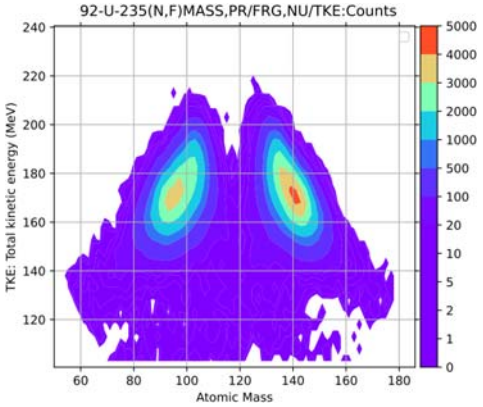


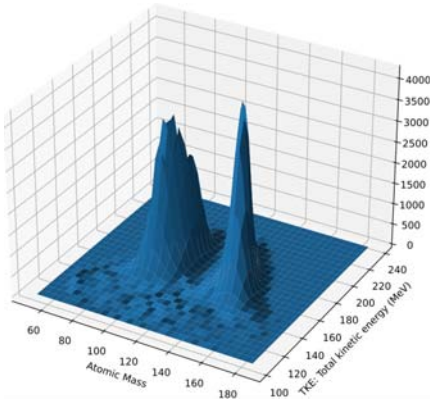




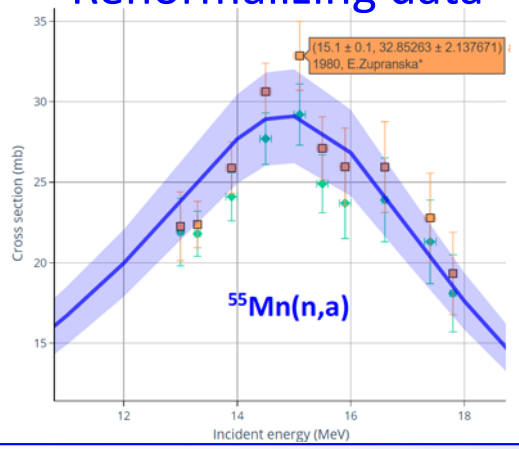
"Hidden" data



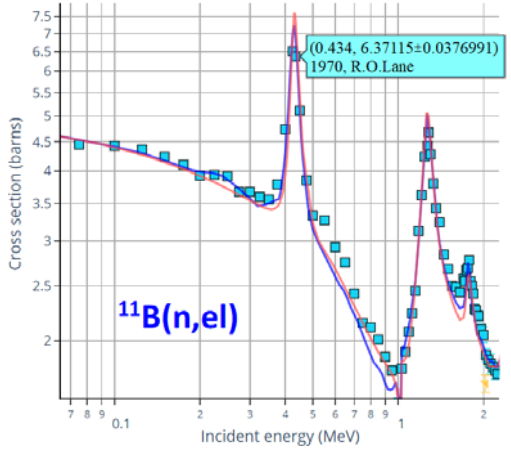


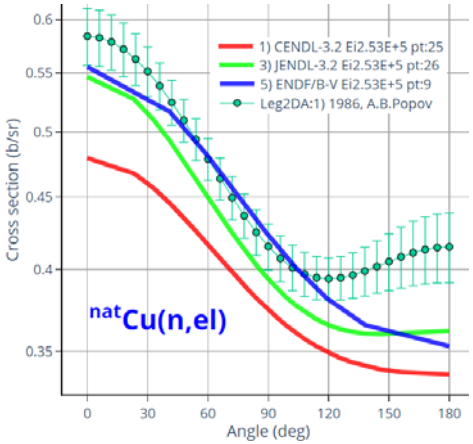


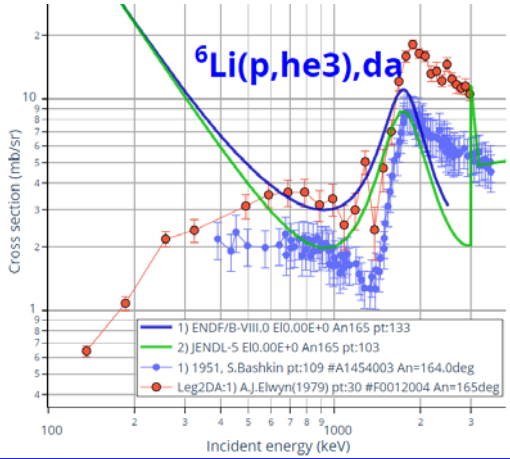
Renormalizing data



"Recalculating" data: LEG→SIG, LEG/RS+SIG→DA, LEG→DA:R33







Since December 2022

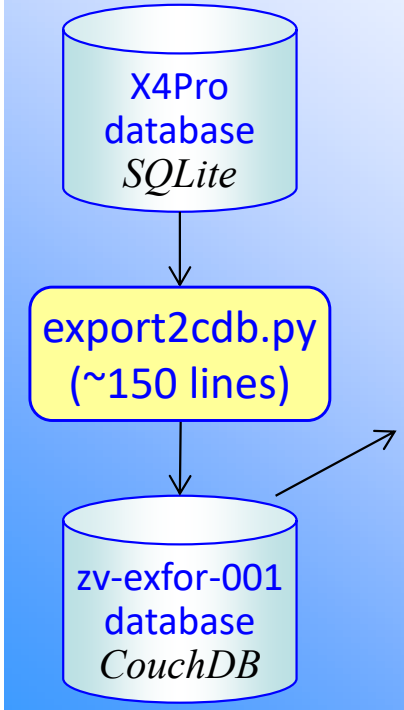
a) DB includes X5-JSON b) Python example: export to CouchDB

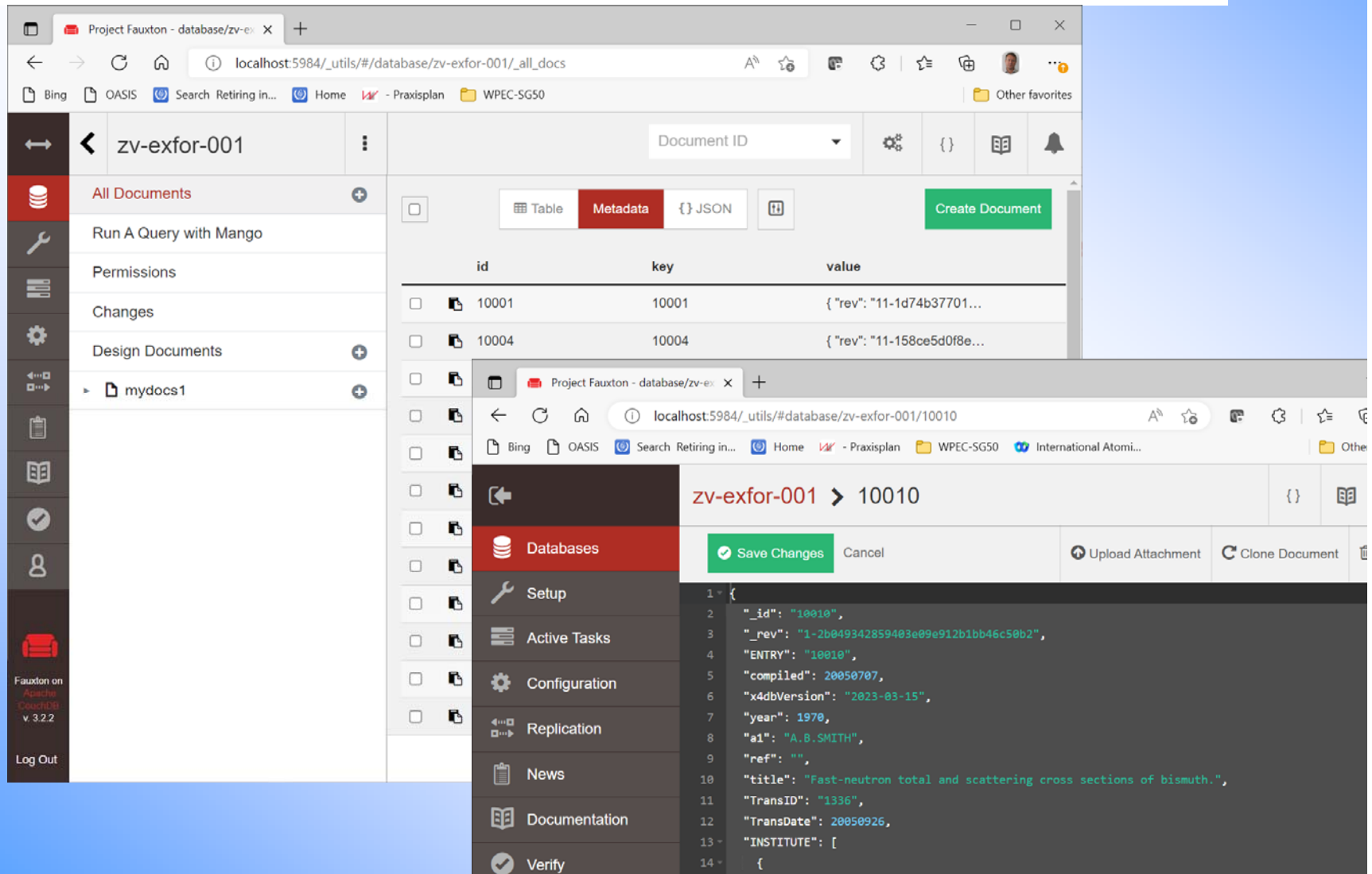
X5-JSON presents meta and numerical data:

1. *from EXFOR and Dictionaries structured as they are in EXFOR - to be useful by compilers*
2. *computational data by Datasets (~C5) including data for automatic correction by new monitor and decay data*

Available on Web-EXFOR as X4Z and X5Z

Example in X4Pro:





X4Pro summary

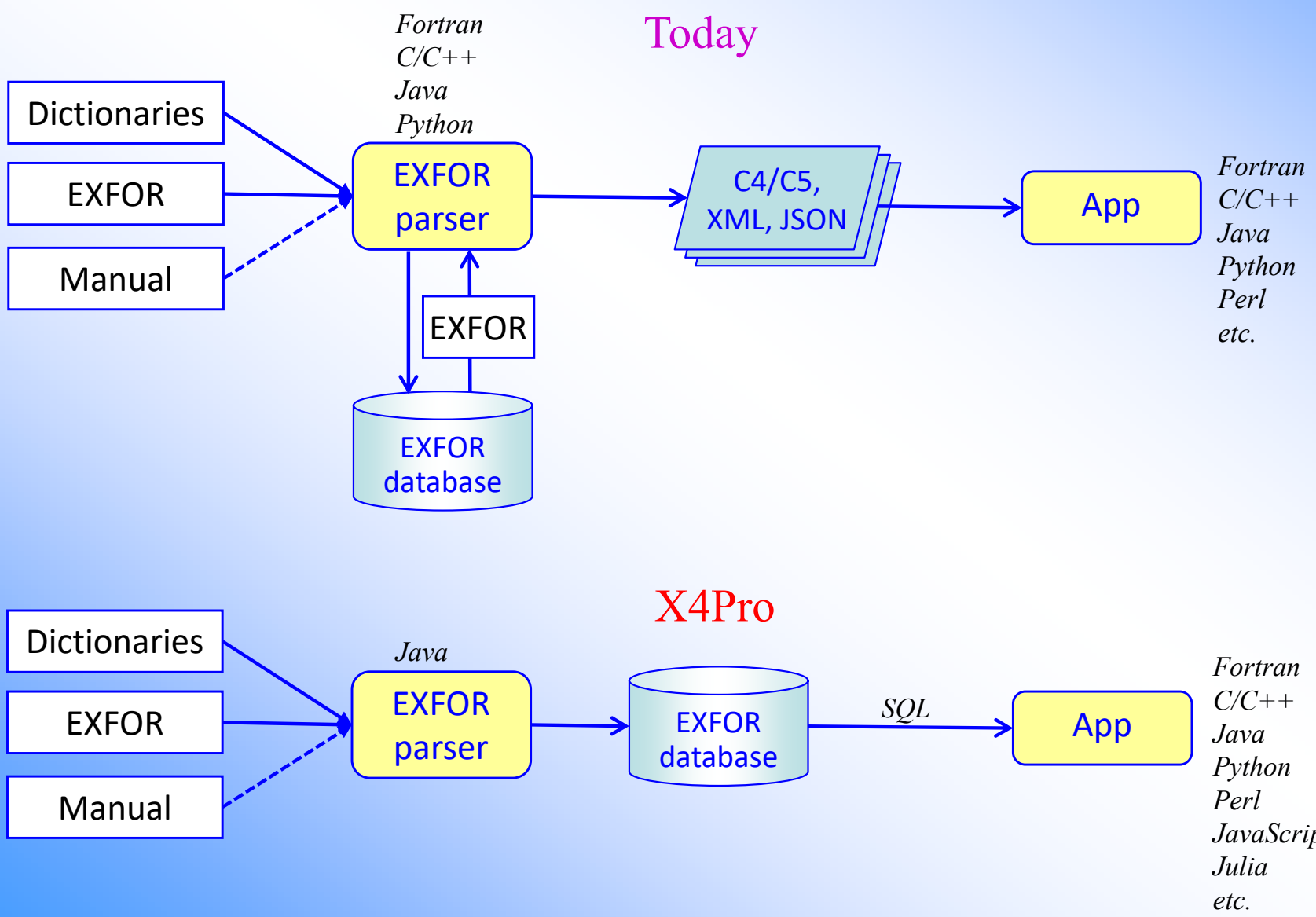
1. X4Pro-trial available for downloading:
   https://www-nds.iaea.org/cdroms/#x4pro1trial

2. Advantages of X4Pro:
   a) universal, flexible, platform-independent, efficient, robust
   b) no need in original EXFOR: all info and data can be taken from the database
   c) no need in EXFOR parsers/converter on user's side
   d) no need for intermediate (C4/C5/XML/JSON) files with fixed structure: application create needed objects on the fly
   e) simple for programming on any programming language supporting SQL for data search, filtering, sorting, retrieval and even renormalization

3. X4Pro status and plans-2022/23:
   a) started public distribution of trial version
   b) presented on NRDC-2022, ND-2022, proposed for testing and feedback
   c) to take part in EXFOR workshop IAEA-2022 (practicing, feedback)
   d) to continue development
   e) to coordinate distribution with NRDC-2023

Key point of X4Pro

X4Pro makes every data point
directly accessible via SQL commands



Part II

Installation X4Pro

1. Download
2. [Linux/MacOS: Install system components]
3. Run test examples
4. Start using database
5. Start using code examples

Download X4Pro

https://www-nds.iaea.org/cdroms/#x4pro1trial

Pilot projects //under development

29 ☐	**X4Apps** 2021-12-15	Dec-2021	EXFOR-CINDA database (SQLite) with GUI (Java) retrieval system for Windows, Linux, MacOSX. Provides scripts and utility codes for EXFOR data search, retrieval and conversion to: Html, XML, JSON, C5. Includes Endver/GUI package integrated with Prepro, EXFOR, ZVView. No installation. [screen-shots] This is united package of: "EXFOR-CINDA for Windows", "EXFOR for Applications", "Endver/GUI" Download (zip, 523Mb)
30 ☐	**x4pro1 trial** 2022-12-22	Dec-2022	X4Pro - universal, fully relational EXFOR database (professional edition). //Trial version. The package includes: ✓ X4Pro/SQLite database, ver.2022-12-22, 7.6Gb (incomplete) ✓ Fortran/C/SQL EXFOR data retrieval demo programs: • execute SQL and retrieve from local X4Pro: F1, SIG, DAE, SIG→C4 • retrieve and recalculate EXFOR data [DA,LEG/RS]&[SIG]→[DA]→[C4] *#mod* ✓ Python demo programs, ver.2022-12-22: • retrieve and plot local EXFOR and remote ENDF data: SIG, DA, DE, DAE, [FY], [DA+MF34] *#new* • renormalize and correct EXFOR data: automatically, by user's and expert's codes • recalculate EXFOR data: [Ratio]→[SIG], [DA,LEG]→[SIG] • recalculate EXFOR data: [DA,LEG/RS]&[SIG]→[DA]; [DA,LEG]→[DA]→[R33] *#new* • retrieve and plot original EXFOR covariance data En×En:[pdf1] Reac×Reac:[pdf2] • export JSON-X5Z from X4Pro to CouchDB (NoSQL database):[all_docs][doc1][find] *#new* Environment: Windows/Linux/MacOS + Python3 with plotly, matplotlib, requests, couchdb See: [page] [readme] [copyright] [license] [version] Presentations:NRDC-2022, ND-2022, [EXFOR Workshop-2022] Download (zip:2.1Gb) Download/mini-version(zip:425Mb) Download/mini-2023(zip:426Mb)

readme.txt

Nuclear Data Section (NDS)
Department of Nuclear Sciences and Applications
International Atomic Energy Agency (IAEA)
Vienna International Centre, P.O. Box 100,
A-1400 Vienna, Austria
Tel:(+43 1) 2600-21714; Fax:(+43 1) 26007

WORKING MATERIALS ON THE DEVELOPMENT OF X4PRO DISTRIBUTION

"X4Pro - universal, fully relational EXFOR database"
Prepared by Viktor Zerkin, IAEA-NDS, 2021-2023
Experimental version for MS-Windows, Linux, MacOS
Last modified: 2023-03-29 by V.Zerkin

CONTENT

3 parts

1) x4sqlite1.db (SQLite): trial* relational database X4Pro (ver.2023-03-29)
   *Note. The database is under development and incomplete.
2) x4pro9: directory with set of python3 and gfortan demo programs to retrieve and modify EXFOR data and plot them together with evaluated data retrieved from ENDF Web system on http://www-nds.iaea.org/endf
3) MS-Windows system components*:
   3.a) MinGW with GCC and GNU Fortran-9.2.0
   3.b) Python-3.9.5 with installed packages: plotly-5.7.0, matplotlib-3.5.2 and requests-2.27.1
   *Note. Provided to avoid any installation on MS-Windows.

LICENSES

1) See LICENSE.TXT
2) For third party software, please see the README, "license.terms" files that come in the associated directories.

readme.txt INSTALL

```
INSTALL
Windows*:
 1) Uncompress file "x4pro1-trial-2023-03-29.zip" to HD disk
    (required free space on HD disk: ~8Gb)
 2) start Explorer and create icon "run-x4pro.bat":
    right-click on the file run-x4pro.bat and select "Send to: Desktop"
 3) start Explorer and create icon "start-x4pro.bat":
    right-click on the file start-x4pro.bat and select "Send to: Desktop"
 *Note. Windows distribution is portable (having embedded python3, mingw, gfortran)

 Linux/MacOS:
 1) Uncompress file "x4pro1-trial-2023-03-29.zip" to HD disk
    (required free space on HD disk: ~8Gb)
      $ unzip x4pro1-trial-2023-03-29.zip
      $ rm -rf x4pro1-trial/win-*
 2) check python3 (install if you don't have it)
      $ python3 --version
 3) install needed pip3 and python3 packages:
      $ sudo apt-get install python3-pip
      $ pip3 install plotly
      $ pip3 install requests
      $ pip3 install matplotlib
        Note. If pip3 install fails with erro-message:
        "connection error: [SSL: CERTIFICATE_VERIFY_FAILED]..."
      $ pip3 install --trusted-host files.pythonhosted.org \
        --trusted-host pypi.org --trusted-host pypi.python.org 'plotly>=4.0.0'
 4) install and check gfortran
      $ gfortran --version
 5) make fortran executables
      $ cd x4pro1-trial/x4pro9
      $ ./init-lin.sh
```

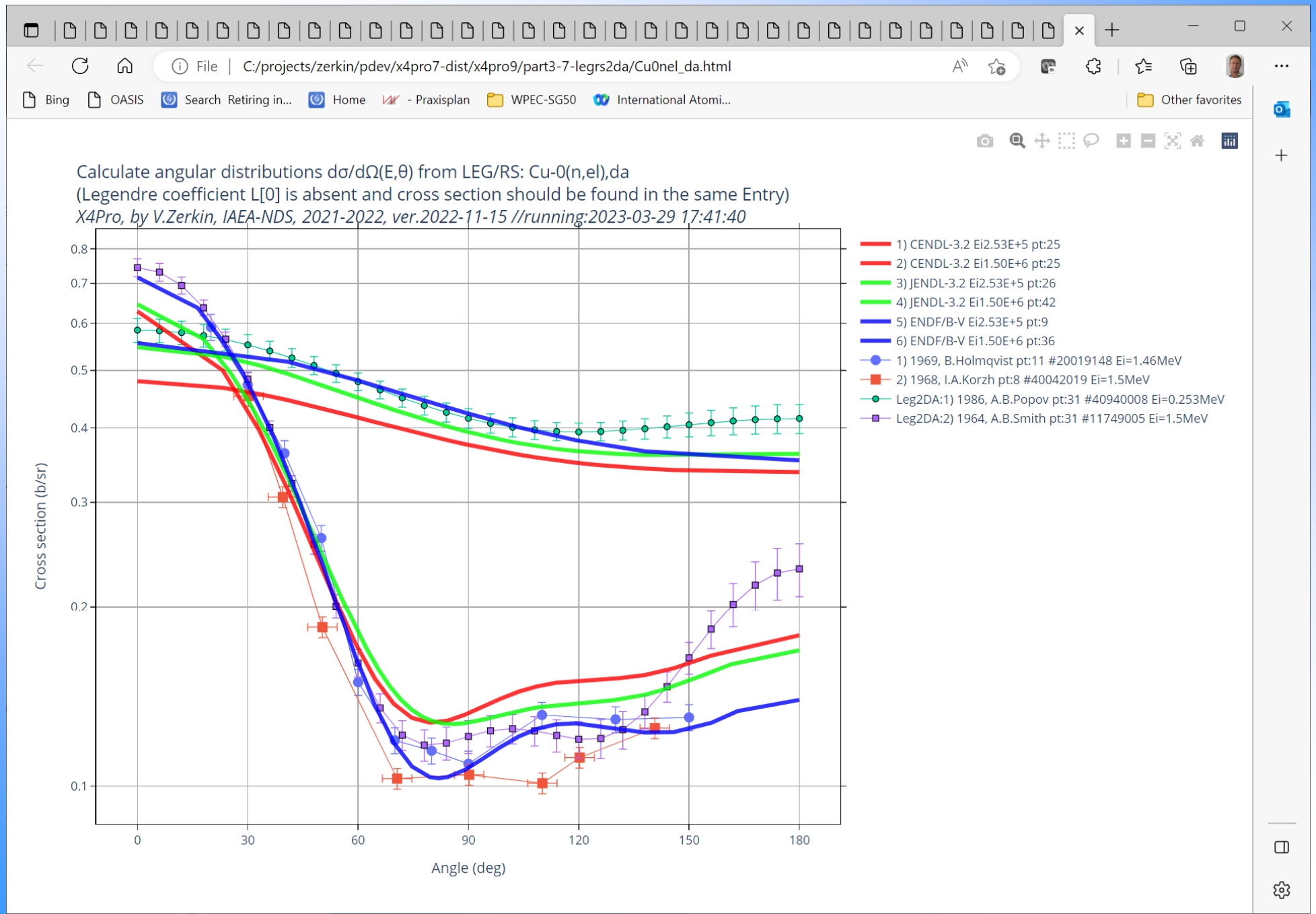
readme.txt STARTUP

```
STARTUP
 Windows:
 1) Run all demo-examples:
    double click on the icon "run-x4pro.bat - short-cut"
    or run Explorer and double-click on the file "run-x4pro.bat"
 2) Start using package. Set environment and continue working:
    click on the icon "start-x4pro.bat - short-cut"
    or run Explorer and double-click on the file "start-x4pro.bat"
 ...you should have opened cmd - terminal window
    2a) Quick demo test
        x4pro9> bash run-all.sh quick
        ...script will run one test and open Html file in your Web browser...
    2b) Start all demos (or go to selected dir and start individual demo-program)
        x4pro9> bash run-all.sh
        ...script will run all tests and open Html files in your Web browser...
        ...the total running time should be approximately 2-3 minutes....
    2c) Run single example, e.g. script sig1x.py from test-10 in part2-1-sig1/
        x4pro9> cd part2-1-sig1
        part2-1-sig1> python -B sig1x.py "A1-27" "n,a" log lin

 Linux/MacOS:
 1) $ cd x4pro1-trial/x4pro9
 2) Quick demo test
    $ bash run-all3.sh quick
 ...script will run one test and open Html file in your Web browser...
 3) Start all demos (or go to selected dir and start individual demo-program)
    $ source run-all3.sh
      or
    $ bash run-all3.sh
      or
    $ ./run-all3.sh
 ...programs run one after another and open Html files in your Web browser...
 ...the total running time should be approximately 2-3 minutes....
```

$ bash run-all.sh → starts X4Pro examples

After 2-3 minute you should finally get 34 Tabs in your Web browser:



Part III

Access X4Pro data with SQL only

(no traditional programming)

Start using database

SQLite database: all in single file "x4sqlite1.db"

We can use "x4sqlite1.db" independently from X4Pro codes with 2 products:

1. sqlite3 (SQL interpreter using command line)
2. DB Browser for SQLite

SQLite Download Page:
https://www.sqlite.org/download.html

DB Browser for SQLite.
Download: https://sqlitebrowser.org/dl/

Install using command lines.
Linux/Ubuntu:
$ sudo apt-get install sqlite3
$ sudo apt-get install sqlitebrowser

Using sqlite3 by command lines

SQL *Hello, World!*

```
select 'Hello, world'
```

How to run it

```
$ sqlite3 x4sqlite1.db "select 'Hello, world',5*3.5-2"
Hello, world|15.5
```

How to execute SQL command from a text file and write result to another file

```
$ sqlite3.exe x4sqlite1.db <sql1.txt >result1.txt
```

Create table using existing tables

```
create table AAA as select * from AUTHORS where Author like 'A%';
select * from AAA;
10017|0|1969|3|10017|R.O.|Akyuz|R.O. Akyuz
10126|0|1970|2|10126|G.P.|Arnold|G.P. Arnold
10141|0|1971|1|10141|B.J.|Allen|B.J. Allen
```

Delete existing table

```
drop table AAA
```

Optimize disk space (after modifications only: drop table, update, delete)

```
vacuum -- SQLite compress database (free unused disk space) very slow
optimize table ENTRY - MySQL, MariaDB
```

Our main interest: SELECT command

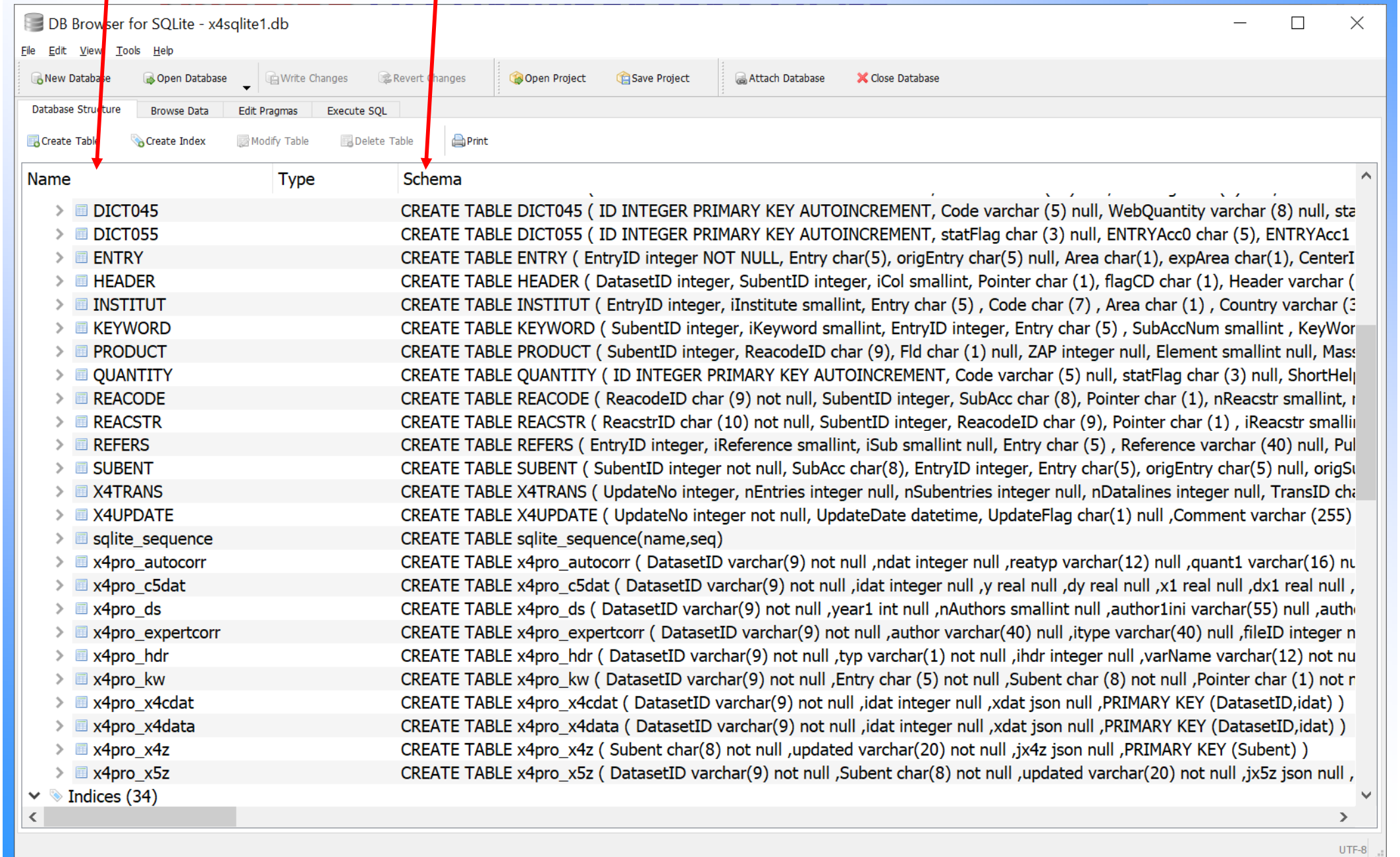
```
SELECT 'Example 1',Entry,DateDebut,Author1ini,Author1,TransDate,TransFile,Reference1,nAuthors
FROM  ENTRY
WHERE Author1='Korzh' AND YearRef1>=1977 AND nAuthors<5
ORDER BY Entry
LIMIT 1,4
Example 1|32230|2009-12-11|I.A.|Korzh|20100126|EXFOR-2015-05-05.bck||J,AE,62,(6),417,1987|3
Example 1|40618|1983-03-29|I.A.|Korzh|20210914|trans.4197||J,UFZ,25,(1),109,1980|3
Example 1|40619|1982-11-17|I.A.|Korzh|20180525|trans.4178||J,YF,35,1097,1982|4
Example 1|40858|1985-05-05|I.A.|Korzh|20050926|EXFOR-2015-05-05.bck||C,83KIEV,3,167,198310|3
```

Using DB Browser for SQLite

How to start it

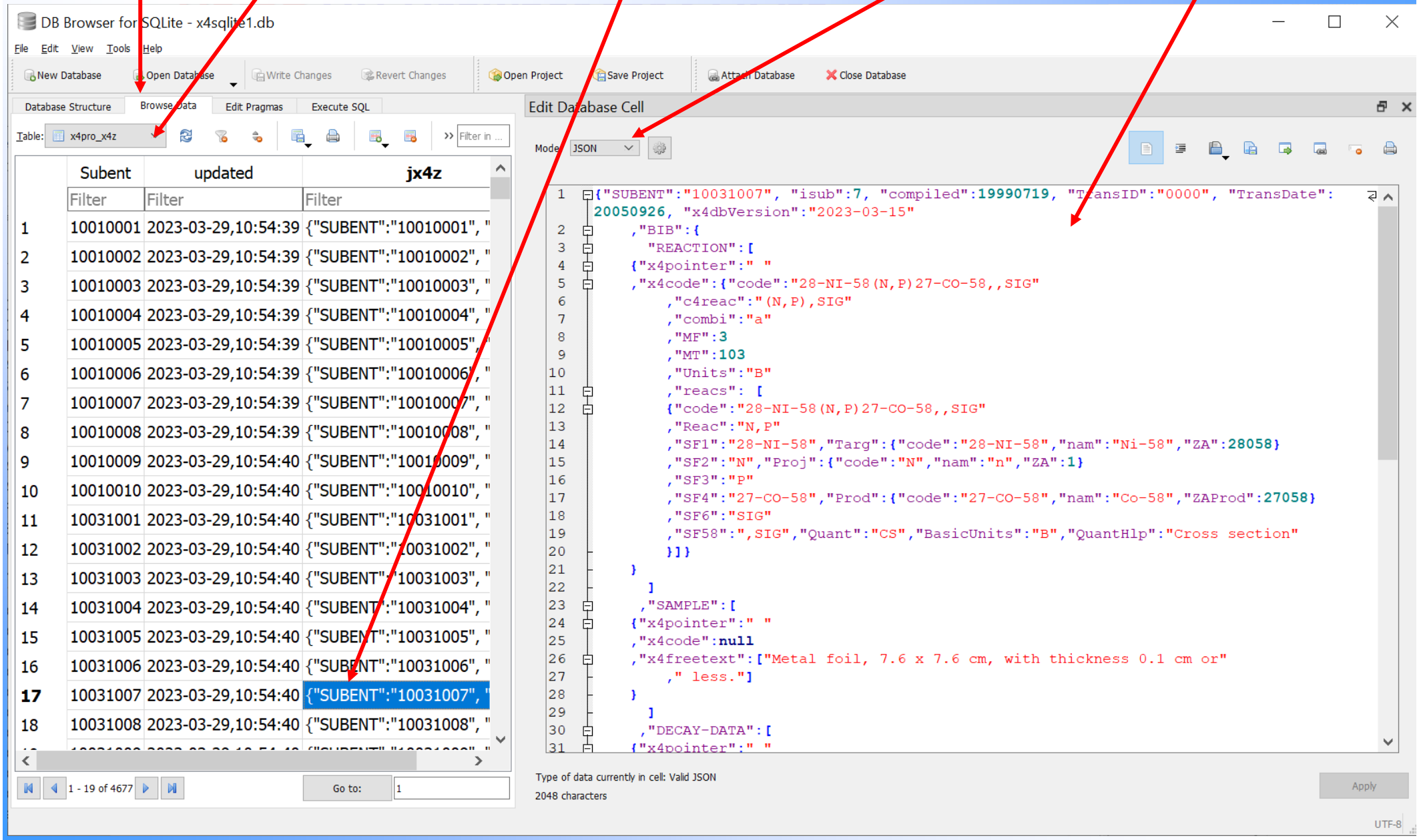
```
$ "C:\projects\zerkin\sqlite-dev\dev-02\DB Browser for SQLite\DB Browser for SQLite.exe" x4sqlite1.db
```

Tables, Indices, Views Schema



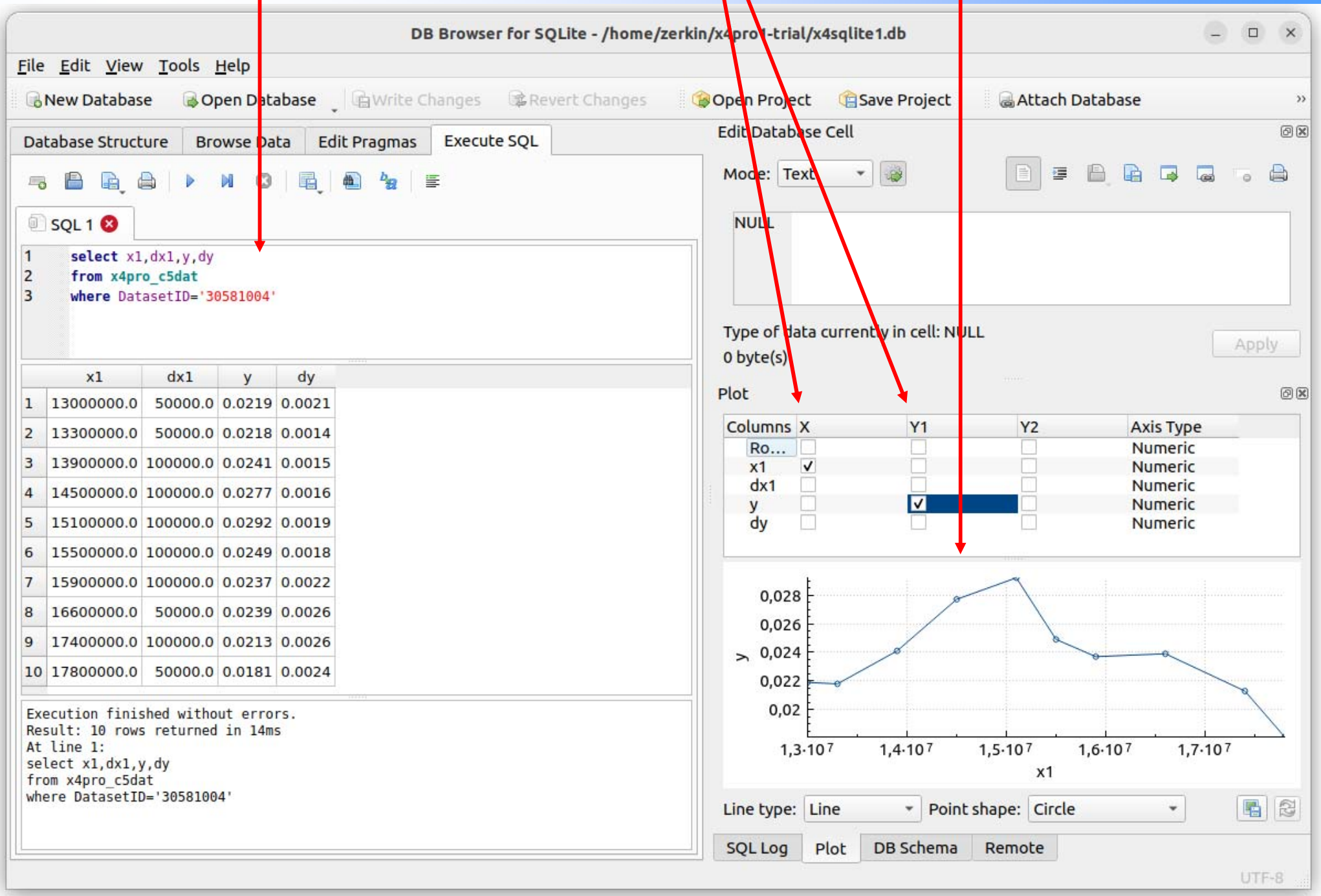
Using DB Browser for SQLite

[Browse data]→ [Table: x4pro_x4z]→ [value in column: jx4z]→ [Mode: JSON]→ [display value]



Plotting in SQLite DB Browser

[Execute SQL] → [Select columns for plot] → [Plot with options]



SELECT statement with JSON_EXTRACT()

```
select json_extract(xdat,'$.EN') as En
,json_extract(xdat,'$.ANG') as Ang
,json_extract(xdat,'$.DATA-CM') as Dat
,xdat
from x4pro_x4data
where DatasetID='A1495003'
and Ang=150 -- MariaDB: and json_extract(xdat,'$.ANG')=150
order by En -- MariaDB: order by json_extract(xdat,'$.EN')
```

```
create table x4pro_x4data (
    DatasetID   varchar(9) not null,
    idat        integer null,
    xdat        JSON null,
    primary key (DatasetID,idat)
)
```

Database Structure | Browse Data | Edit Pragmas | Execute SQL

SQL 1

```
1  select json_extract(xdat,'$.EN') as En
2  ,json_extract(xdat,'$.ANG') as Ang
3  ,json_extract(xdat,'$.DATA-CM') as Dat
4  ,xdat
5  from x4pro_x4data
6  where DatasetID='A1495003' and Ang=150
7  order by En
```

	En	Ang	Dat	xdat
1	0.8989	150.0	0.7892	{"DATA-CM":0.7892,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
2	0.9216	150.0	0.8881	{"DATA-CM":0.8881,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
3	0.9518	150.0	1.036	{"DATA-CM":1.036,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
4	0.9554	150.0	1.135	{"DATA-CM":1.135,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
5	0.9971	150.0	1.271	{"DATA-CM":1.271,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
6	1.008	150.0	1.357	{"DATA-CM":1.357,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
7	1.05	150.0	1.531	{"DATA-CM":1.531,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
8	1.088	150.0	1.716	{"DATA-CM":1.716,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
9	1.125	150.0	1.95	{"DATA-CM":1.95,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...

```
Execution finished without errors.
Result: 99 rows returned in 4ms
At line 1:
select json_extract(xdat,'$.EN') as En
,json_extract(xdat,'$.ANG') as Ang
,json_extract(xdat,'$.DATA-CM') as Dat
,xdat
from x4pro_x4data
where DatasetID='A1495003' and Ang=150
order by En
```

Edit Database Cell

Mode: Text

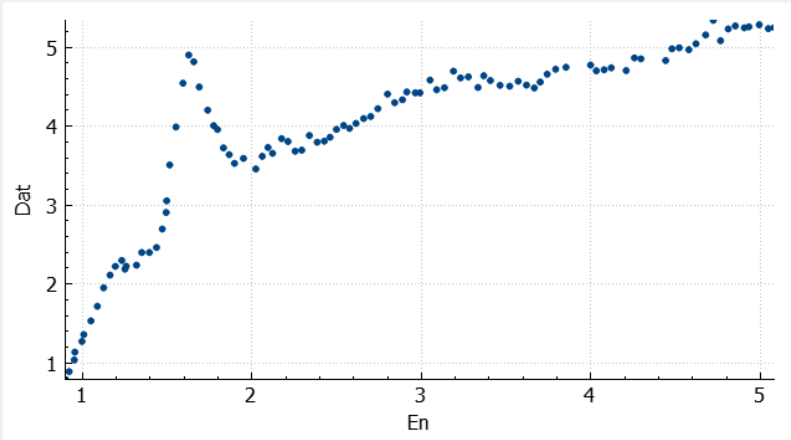
```
1  {"DATA-CM":1.036,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":0.9
   518,"EN-ERR-DIG":0.004,"E-LVL":2.9,"ANG":150.0}
```

Type of data currently in cell: Valid JSON
104 characters

Apply

Plot

Columns	X	Y1	Y2	Axis Type
Row...	☐	☐	☐	Numeric
En	☑	☐	☐	Numeric
Ang	☐	☐	☐	Numeric
Dat	☐	☑	☐	Numeric



Line type: None Point shape: Disc

UTF-8

Simplified data renormalization in SQLite DB Browser

Table x4pro_c5dat has column Fcm0 containing factor for renormalization by values of monitor (m1/m0)

DB Browser for SQLite - x4sqlite1.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL | Edit Database Cell

SQL 1

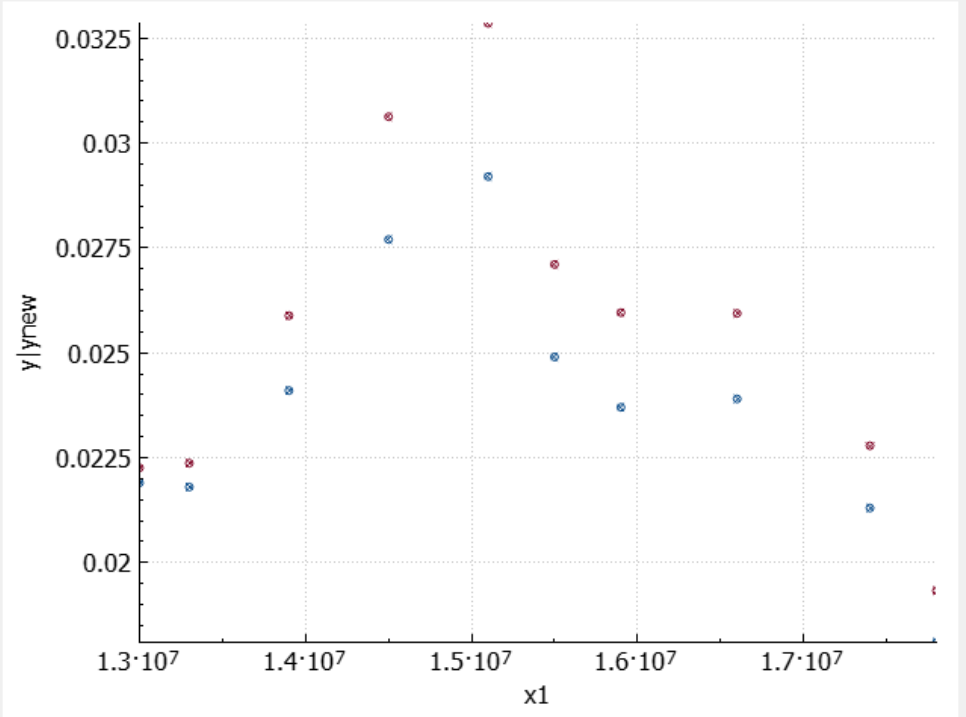
```
1  select x1,dx1,y,dy,
2  Fcm0,(y*Fcm0) as ynew,(dy*Fcm0) as dynew
3  from x4pro_c5dat
4  where DatasetID='30581004'
```

	x1	dx1	y	dy	Fcm0	ynew	dynew
1	13000000.0	50000.0	0.0219	0.0021	1.01635	0.022258065	0.002134335
2	13300000.0	50000.0	0.0218	0.0014	1.02609	0.022368762	0.001436526
3	13900000.0	100000.0	0.0241	0.0015	1.07402	0.025883882	0.00161103
4	14500000.0	100000.0	0.0277	0.0016	1.10564	0.030626228	0.001769024
5	15100000.0	100000.0	0.0292	0.0019	1.12509	0.032852628	0.002137671
6	15500000.0	100000.0	0.0249	0.0018	1.08857	0.027105393	0.001959426
7	15900000.0	100000.0	0.0237	0.0022	1.09522	0.025956714	0.002409484
8	16600000.0	50000.0	0.0239	0.0026	1.0853	0.02593867	0.00282178
9	17400000.0	100000.0	0.0213	0.0026	1.06965	0.022783545	0.00278109
10	17800000.0	50000.0	0.0181	0.0024	1.06795	0.019329895	0.00256308

```
Execution finished without errors.
Result: 10 rows returned in 9ms
At line 1:
select x1,dx1,y,dy,Fcm0,(y*Fcm0) as ynew,(dy*Fcm0) as dynew from x4pro_c
where DatasetID='30581004'
```

Plot

Columns	X	Y1	Y2	Axis Type
Row...	☐	☐	☐	Numeric
x1	☑	☐	☐	Numeric
dx1	☐	☐	☐	Numeric
y	☐	☑	☐	Numeric
dy	☐	☐	☐	Numeric
Fcm0	☐	☐	☐	Numeric
ynew	☐	☑	☐	Numeric



Line type: None | Point shape: CrossCircle

Part IV

Access to X4Pro data from Python and Fortran via SQL commands

Python examples

Hello, World!

```
print('Hello, world') #line comment
```

How to run it

```
$ python p1.py
Hello, world
```

How to run it

```
$ python3 p1.py
Hello, world
```

Connect to SQLite database, execute SQL statement. Note: sqlite3 comes with python3

```
import sqlite3
try:
    conn=sqlite3.connect("x4sqlite1.db",uri=True)
    conn.row_factory=sqlite3.Row  #in order to access data by column names
except Error:
    print(Error)
cursor=conn.cursor()
sql="select Entry,YearRef1,Author1 from ENTRY where Entry like 'F%'"
try:
    cursor.execute(sql)
    rows=cursor.fetchall()
except Error:
    print(Error)
cursor.close()
conn.close()
ii=0
for row in rows:
    Entry=row[0]
    Entry=row['Entry']   #access data by column name
    YearRef1=row[1];  Author1=row[2] #several instructions in one line separated by <;>
    ii+=1
    print('\t'+str(ii)+') '+str(Entry)+' '+str(YearRef1)+' '+Author1)
```

```
$ python db0.py
          1) F0001 1976 Gould
          2) F0002 1968 Ludecke
          3) F0003 1953 Sawyer
          4) F0004 1975 Liskien
          5) F0005 1979 Shinozuka
          . . . . .
```

Fortran examples

Hello, World! File: f0.f

```
      write (*,*) 'Hello, world'
      end
```

How to run it

```
$ gfortran -o f0 f0.f
$ f0
Hello, world
```

File f1.f : connect to SQLite database, execute SQL statement, call subroutine reading results

```
      program f1
      character*512  dbname
      character*512  outfile/'sql1tmp.dat'/
      character*4096 sql
      write (*,*) 'Program: f1.f (ver.2022-12-12)'
      write (*,*) 'by V.Zerkin, IAEA-NDS, 2021-2022'
      dbname='../../x4sqlite1.db'
      ierr=ix4lite_open(trim(dbname)//char(0))
      !--- char(0) added for compatibility with C
      if (ierr.ne.0) then
          write (*,*) 'Can''t open database'
          stop
      endif
      write (*,*) 'Open database: ',trim(dbname),' ierr=',ierr
      write (*,*)
      sql='select Entry,YearRef1,Author1'
     1  //' from ENTRY where Entry like ''F%'''
      write (*,*) 'SQL command:'
      write (*,*) trim(sql)
      sql=trim(sql)//char(0)
      outfile=trim(outfile)//char(0)
      ipnt=ix4lite_exec(sql,outfile)
      if (ierr.lt.0) then
          write (*,*) 'SQL error ierr=',ipnt
      else
          write (*,*) 'SQL executed OK:',ipnt,' rows'
      endif
      call read_data(outfile,ipnt)
      call ix4lite_close();
      write(*,'(/a)') 'Program completed successfully'
      stop
      end
```

Traditional Fortran-SQL problem: dealing with data types, pointers, C-style data and programming.

My approach for X4Pro: use simple C subroutines writing result of SQL to temporary text file and read result in Fortran code using formatted input. No pointers, almost no C.

Temporary file with result of SQL

```
#### 1
$Entry       F0001
$YearRef1    1976
$Author1     Gould
//
#### 2
$Entry       F0002
$YearRef1    1968
$Author1     Ludecke
//
#### 3
$Entry       F0003
$YearRef1    1953
$Author1     Sawyer
//
. . . . . .
#### 1427
$Entry       F1464
$YearRef1    1990
$Author1     Ignatenko
//
//EOF
```

Fortran examples

Continue file f1.f : Subroutine reading results from temporary file

```
      subroutine read_data(infile,ipnt)
      character(len=*) infile
      character*132 nam,line
      character*5   Entry
      character*4   sYear
      character*12  Author1
      write (*,*) 'Read data points:',ipnt
      nin=100
      open(unit=nin,file=infile,status='old',err=9999)
      iipnt=0
      do
          read(nin,'(a12,a)',end=8888) nam,line
          if (nam.eq.'//EOF')       exit
          if (nam.eq.'$Entry')      Entry=line
          if (nam.eq.'$YearRef1')   read(line,'(i11)') iYear
          if (nam.eq.'$YearRef1')   sYear=line
          if (nam.eq.'$Author1')    Author1=line
          if (nam(1:4).eq.'####') then !--- Begin of new Row
             iipnt=iipnt+1
             Entry=''
             iYear=0
             sYear='----'
             Author1=''
             cycle
          endif
          if (nam.eq.'//') then !--- End of Row
             write(*,*) iipnt,Entry,' ',sYear,',',Author1
          endif
      enddo
8888  close(nin)
9999  return
      end
```

Temporary file with result of SQL

```
#### 1
$Entry       F0001
$YearRef1    1976
$Author1     Gould
//
#### 2
$Entry       F0002
$YearRef1    1968
$Author1     Ludecke
//
#### 3
$Entry       F0003
$YearRef1    1953
$Author1     Sawyer
//
. . . . . .
#### 1427
$Entry       F1464
$YearRef1    1990
$Author1     Ignatenko
//
//EOF
```

How to compile it. Note: sqlite3.c is provided by SQLite developers team.

```
          #Windows:
$ gfortran f1.f sql1sub.c sqlite3.c -o f1
          #Linux, MacOS:
$ gfortran f1.f sql1sub.c sqlite3.c -o f1.exe -pthread -ldl
```

How to run it

```
$ f1
$ ./f1
 Program: f1.f (ver.2022-12-12)
 by V.Zerkin, IAEA-NDS, 2021-2022
 Open database: ../../x4sqlite1.db ierr=0

 SQL command:
 select Entry,YearRef1,Author1
 from ENTRY where Entry like 'F%'
 SQL executed OK:         1427  rows
 Read data points:         1427
           1 F0001 1976,Gould
           2 F0002 1968,Ludecke
           3 F0003 1953,Sawyer
```

Part V

Structure of programs in X4Pro

Location of database and modules

Database, Windows packages

```
\---x4pro1-trial
    |   COPYRIGHT.TXT
    |   LICENSE.TXT
    |   readme.txt
    |   run-x4pro.bat
    |   VERSION.TXT
    |   win-env.bat
    |   x4sqlite1.db
    +---win-bash-gfortran
    +---win-python3
    \---x4pro9
```

Common modules, shell scripts

```
x4pro9
|   auto_corr.py
|   COPYRIGHT.TXT
|   dbConn-sqlite.py
|   dbConn.py
|   endf2plot-matpl.py
|   endf2plot-plotly.py
|   endf2plot.py
|   exfor2plot-matpl.py
|   exfor2plot-plotly.py
|   exfor2plot.py
|   expert_corr.py
|   index.html
|   init-lin.sh
|   LICENSE.TXT
|   quick-test.sh
|   quick-test3.sh
|   readme.txt
|   run-all.sh
|   run-all3.sh
|   rweb11.py
|   rweb12.py
|   set1.sh
|   set2.sh
|   times.log-0
|   tree.txt
|   VERSION.TXT
|   x4out.py
|   x4pro.txt1
+---part1-0-sig
. . . . . .
```

Examples by parts

```
+---part1-0-sig
+---part1-1-sig
+---part1-2-da
+---part1-3-dap
+---part1-4-de
+---part1-5-dae
+---part1-6-fy
+---part1-7-covar
+---part1-fortran
+---part2-1-sig1
+---part2-2-da1an
+---part2-3-da1ei
+---part2-4-de1eo
+---part2-5-dae1eo
+---part2-6-fy
+---part3-1-auto1
+---part3-2-user1
+---part3-3-expert1
+---part3-4-auto2
+---part3-5-ratio2sig
+---part3-6-daleg2sig
+---part3-7-legrs2da
+---part3-8-leg2r33
\---part4-0-couchdb
```

Switch output by setup scripts

Most of the scripts can be switched from using
Plotly to Matplotlib and back
*(replacing source files *.py)*

```
Windows:
1) Set Plotly
   x4pro9> bash set1.sh
2) Set Matplotlib
   x4pro9> bash set2.sh

Linux/MacOS:
1) Set Plotly
   $ ./set1.sh
2) Set Matplotlib
   $ ./set2.sh
```

Content of an example directory

*.py	*Python codes (main script and modules)*
runme.sh	*Run example(s) using python*
runme3.sh	*Run example(s) using python3*
*.tto.txt	*Terminal output (stored)*
*.tto	*Terminal output (last run by user)*
*.htm	*Html output (stored)*
*.html	*Html output (last run by user)*
*.json.txt	*JSON output (stored)*
*. json	*JSON output (last run by user)*
*.pdf, *.png	*Last PDF produced by Matplotlib*

Structure of program in Python

1) Import packages and common modules

```
import os
import sys
import datetime
sys.path.append('./')
sys.path.append('../')
import dbConn
from rx5de import *
from x4out import *
from exfor2plot import * #plot by plotly or matplotlib
```

2) Open database

3) Create SQL command

4) Execute SQL command

5) Process rows and create List of datasets, where dataset={label, x[], y[], dx[], dy[]}

6) prepareExforDataForPlot(datasets, plotting params)

7) Required Libs={LibName:LibColor,...}

8) webEndfDataForPlot: retrieve list of ENDF datasets, retrieve datasets

9) prepareEndfDataForPlot

10) myOfflinePlot(x4datasets, endfdatasets, params)

11) print('\nProgram successfully completed')

Simples example: part1-0-sig/sig0x.py

4) Execute SQL command:

```
select * from sig1 where (Target like 'Al-27') and (Reaction like 'n,a')
```

"From" refers to "View"

"View" is the result set of a stored query

```
CREATE VIEW sig1 AS
 select x4pro_c5dat.DatasetID
 ,x4pro_c5dat.idat as iPoint
 ,REACODE.fullCode
 ,REACODE.Pointer,ENTRY.Entry,REACODE.SubAcc as Subent
 ,ENTRY.YearRef1,ENTRY.nAuthors,ENTRY.Author1Ini,ENTRY.Author1
 ,REACSTR.Target, REACSTR.Reaction
 ,lower(REACSTR.Projectile) as Projectile
 ,REACSTR.sProd,REACSTR.sTarg
 ,REACODE.zaTarget1,REACODE.zaIncident1
 ,REACODE.outParticles,REACODE.MF,REACODE.MT
 ,x4pro_c5dat.x1  as En
 ,x4pro_c5dat.dx1 as dEn
 ,x4pro_c5dat.y   as Sig
 ,x4pro_c5dat.dy  as dSig
from x4pro_c5dat
 inner join REACODE on REACODE.ReacodeID=x4pro_c5dat.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.SF58 like ',SIG')
  and (REACSTR.SF8='')
  and ((REACSTR.SF9='') or (REACSTR.SF9='EXP'))
  and (REACODE.nReacstr=1)
order by
  REACODE.fullCode,ENTRY.YearRef1 desc,x4pro_c5dat.DatasetID
  ,En,x4pro_c5dat.idat
```

5) Process rows and create List of datasets, where dataset={label, x[], y[], dx[], dy[]}

```
def getDatasets4plot(rows,fx=1e-6,fy=1e3):
    lx=len(rows)
    datasets=[]
    ii=0; lastDatasetID=''; lastDataset={}
    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(YearRef1)+','+Author1)
        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(YearRef1)+' '+Author1+" x:"+str(xx)+" y:"+str(yy)+" dy:"+str(dyy)+" dx:"+str(dxx))
    return datasets
```

5) Process rows and create List of datasets, where dataset={label, x[], y[], dx[], dy[]}

```
datasets=getDatasets4plot(rows)
print('datasets:',len(datasets))
ldata=len(datasets)
if (ldata<=0):
    print("---No data found---")
    sys.exit(2)
#_________________Output EXFOR datasets_________________
with open(outhtml+'.json','w') as outfile:
    json.dump(datasets,outfile,indent=2)
```

__getDatasets from datapoints: 661

datasets: 92


```
[
  {
    "DatasetID": "31842017",
    "Reacode": "13-AL-27(N,A)11-NA-24,,SIG",
    "x4lbl": "2022, J.Jarosik",
    "x":  [      17.5,      19.8,      27.5    ],
    "y":  [      63.0,      35.0,      12.6    ],
    "dy": [      6.0,       4.0,       1.7     ],
    "dx": [      1.0,       0.9,       0.7     ]
  },
  {
    "DatasetID": "31834002",
    "Reacode": "13-AL-27(N,A)11-NA-24,,SIG",
    "x4lbl": "2020, D.Kral",
    "x":  [      29.1    ],
    "y":  [      2.7     ],
    "dy": [      0.4     ],
    "dx": [      null    ]
  },
  {
    "DatasetID": "22312002",
    "Reacode": "13-AL-27(N,A)11-NA-24,,SIG",
    "x4lbl": "1993, Y.Ikeda",
    "x":  [      13.33,     13.57,     13.75,     13.98,     14.22,     14.42,     14.65,     14.91    ],
    "y":  [      126.2,     128.0,     122.0,     122.7,     119.0,     114.5,     112.5,     112.0    ],
    "dy": [      4.4,       4.7,       4.5,       4.4,       5.7,       4.0,       4.0,       3.9      ],
    "dx": [      null,      null,      null,      null,      null,      null,      null,      null     ]
  },
```


etc.

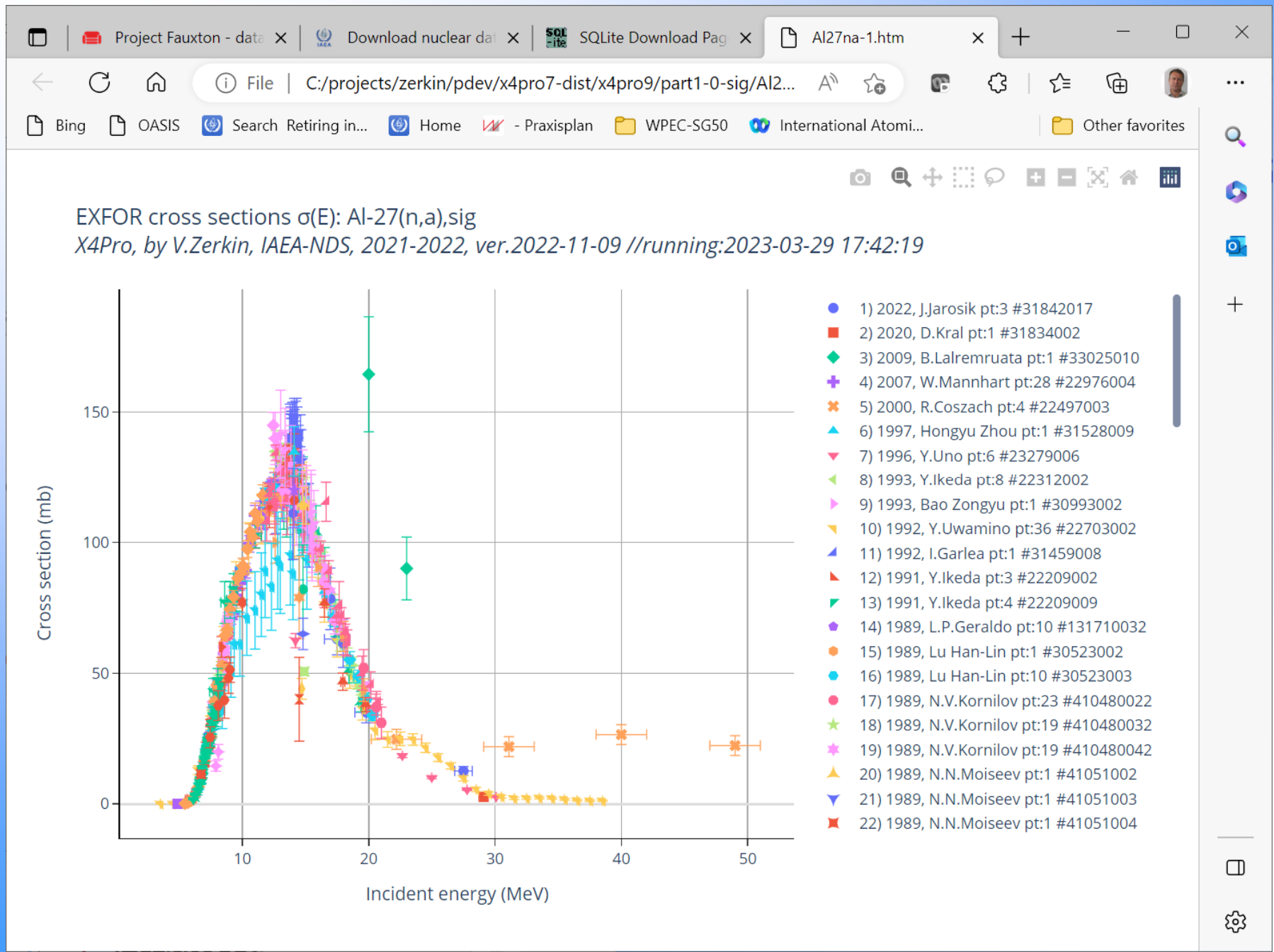
6) prepareExforDataForPlot(datasets, plotting params)
10) myOfflinePlot(x4datasets, endfdatasets, params)

```
#_____________Preparing EXFOR data for plot_____________
data1=[]; ii=0
for dataset in datasets:
    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'])))

#_____________Plot data from EXFOR_____________
plot1={}
plot1['data']=data1
xaxis=dict(title='Incident energy (MeV)',showline=True,linecolor='black',ticks='outside'
            ,showgrid=True,gridcolor='#aaaaaa',type=xtype)
yaxis={'title':'Cross section (mb)','showline':True,'linecolor':'black'#,'type':'log'
            ,'showgrid':True, 'gridcolor':'#aaaaaa','ticks':'outside','type':ytype
            ,'zeroline':True, 'zerolinecolor':'#dddddd'#, 'zerolinewidth':0.1
}
plot1['layout']=Layout(title='EXFOR cross sections \u03c3(E): '+plotTitle
            +'<br><i>X4Pro, by V.Zerkin, IAEA-NDS, 2021-2022, ver.2022-11-09 //running:'+ct+'</i>'
            ,xaxis=xaxis,yaxis=yaxis
            ,plot_bgcolor='white'
            ,legend=dict(traceorder="grouped")
)
plotly.offline.plot(plot1,filename=outhtml+'.html')
```

Final result: Al27na-1.json, Al27na-1.html

After exit, Python opens your browser "offline" with generated html file.



Part VI

Usage of X4Pro in Web Apps.

EE-View

Experimental-Evaluated data Viewer

What is EE-View

EE-View: experimental-evaluated data previewer presenting an additional Web interface to existing EXFOR-ENDF database system. EE-View works in a Web-browser using Html5/JavaScript and plotting package Plotly.js.

EE-View retrieves data from EXFOR/X4Pro and ENDF via AJAX using Web-API.

EE-View provides following functionality:

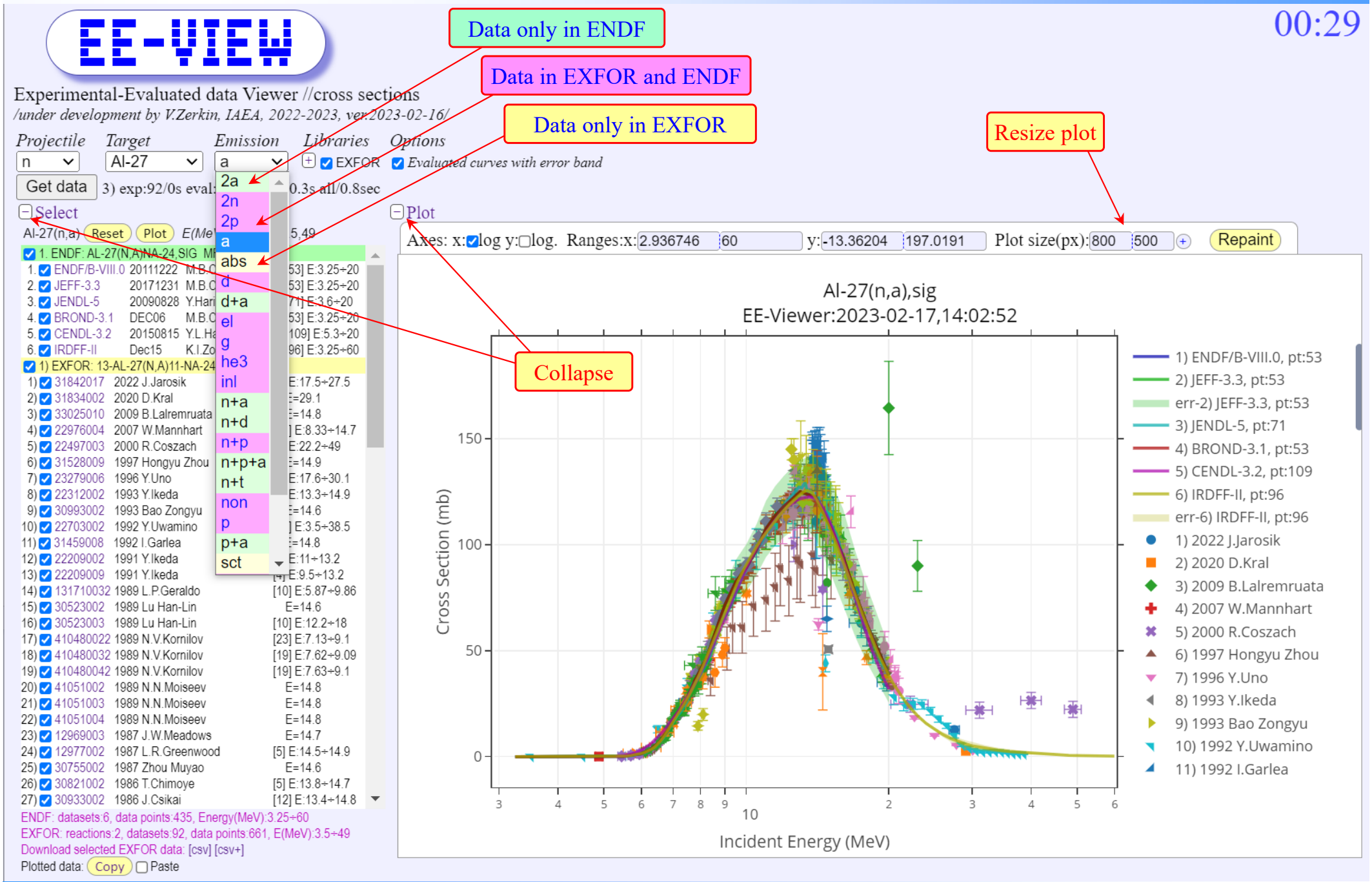
1. *Quick plot EXFOR and ENDF data by one click (few seconds)*
2. *Plot evaluated curves with error-band (MF33/MF34)*
3. *Coloured items in data selection menu indicate existing experimental and evaluated data*
4. *Selection datasets by reaction-codes and energy range*
5. *Copy/paste data to the plot*
6. *Export data to CSV format for uploading to Excel*
7. *Output plot to PNG and SVG using package Plotly.js*
8. *Implemented for cross sections and angular distributions*

EE-View Experimental-Evaluated data Viewer

1. Cross sections with drop-down choice of data: https://www-nds.iaea.org/exfor/eeview.htm
2. Cross sections with open choice of data: https://www-nds.iaea.org/exfor/eeview1.htm
3. Angular distributions: https://www-nds.iaea.org/exfor/eeview-da.htm

EE-View Experimental-Evaluated data Viewer

Cross sections



EE-View Experimental-Evaluated data Viewer

Cross sections

0.8sec

03:25

EE-VIEW

Experimental-Evaluated data Viewer //cross sections

/under development by V.Zerkin, IAEA, 2022-2023, ver.2023-02-16/

Get data | Al-27(n,a) 3) exp:92/0s eval:6/0.4s plot/0.4s all/0.8sec

Projectile:n

g ✓n p d t he3 a He-6 Li-6 Li-7

Target:Al-27

Ag-110M Ag-111 Ag-112 Ag-113 Ag-114 Ag-115 Ag-116 Ag-117 Ag-118M Al-26 Al-26M ✓Al-27 Al-CMP Al-OXI Am-240 Am-241 Am-242 Am-242M Am-243 Am-244 Am-244M Ar-0

Emission:a

2a 2n 2p ✓a abs d d+a el g he3 inl n+a n+d n+p n+p+a n+t non p p+a sct t

Select

Al-27(n,a) Reset Plot *E(MeV)min,max:* 8,18

1. ENDF: AL-27(N,A)NA-24,SIG MF:3 MT:107

1.	ENDF/B-VIII.0	20111222	M.B.Chadwick+	[53] E:3.25÷20
2.	JEFF-3.3	20171231	M.B.Chadwick+	[53] E:3.25÷20
3.	JENDL-5	20090828	Y.Harima+	[71] E:3.6÷20
4.	BROND-3.1	DEC06	M.B.Chadwick+	[53] E:3.25÷20
5.	CENDL-3.2	20150815	Y.L.Han	[109] E:5.3÷20
6.	IRDFF-II	Dec15	K.I.Zolotarev	[96] E:3.25÷60

1) EXFOR: 13-AL-27(N,A)11-NA-24,,SIG

1)	31842017	2022	J.Jarosik	[3] E:17.5÷27.5
2)	31834002	2020	D.Kral	E=29.1
3)	33025010	2009	B.Lalremruata	E=14.8
4)	22976004	2007	W.Mannhart	[28] E:8.33÷14.7
5)	22497003	2000	R.Coszach	[4] E:22.2÷49
6)	31528009	1997	Hongyu Zhou	E=14.9
7)	23279006	1996	Y.Uno	[6] E:17.6÷30.1
8)	22312002	1993	Y.Ikeda	[8] E:13.3÷14.9
9)	30993002	1993	Bao Zongyu	E=14.6
10)	22703002	1992	Y.Uwamino	[36] E:3.5÷38.5
11)	31459008	1992	I.Garlea	E=14.8
12)	22209002	1991	Y.Ikeda	[3] E:11÷13.2

Libraries

EXFOR

Options

Colors

ENDF data only

EXFOR data only

EXFOR ∪ ENDF

EXFOR ∩ ENDF

Statistics of usage: visits: 653, requests: 10

Created by V.Zerkin (v.zerkin@iaea.org), L

Database and Programming: EXFOR/X4P

Experimental Data Source: EXFOR, Netwo

Evaluated Data Source: CSEWG, WPEC, I

Libraries

EXFOR

ENDF/B-VIII.0 (USA,2018)

JEFF-3.3 (Europe,2017)

JENDL-5 (Japan,2021)

CENDL-3.2 (China,2020)

BROND-3.1 (Russia,2016)

IRDFF-II (IAEA,2019)

All other libraries

Options

Evaluated curves with error band

Colors

ENDF data only

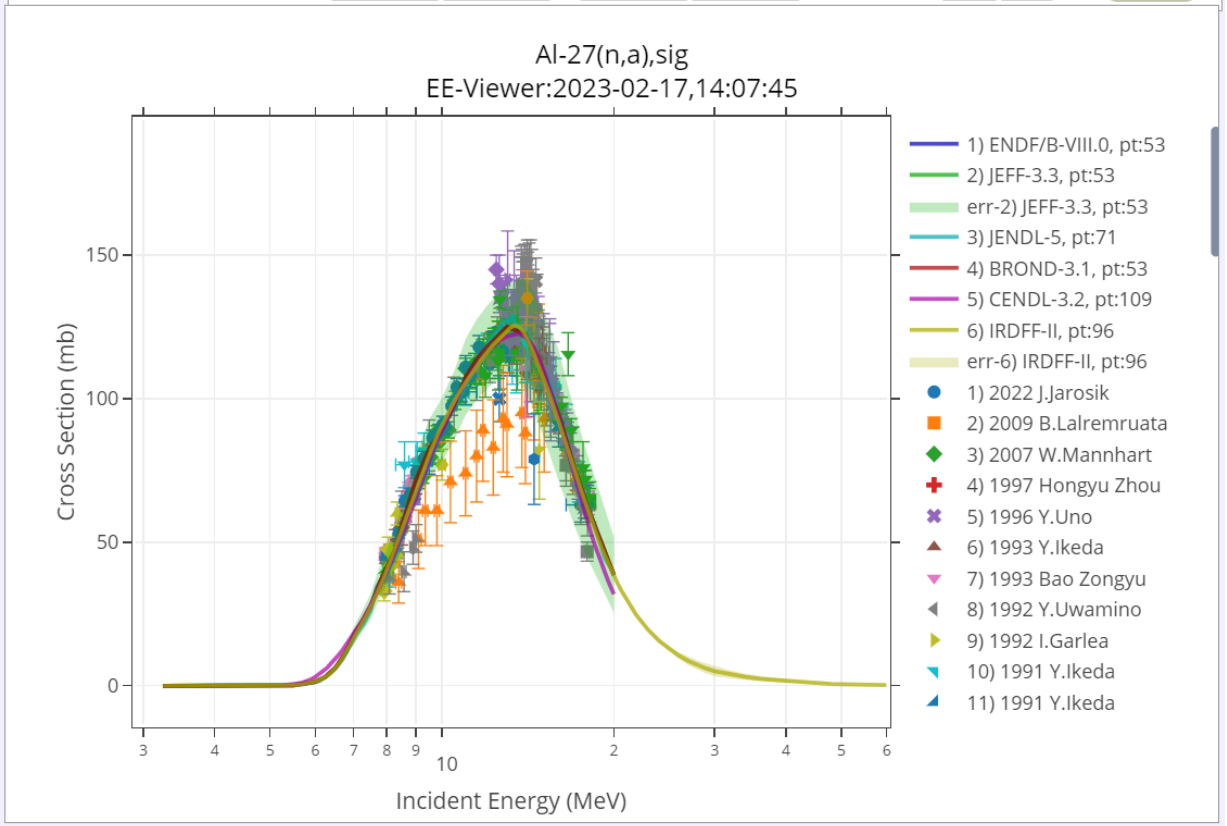
EXFOR data only

EXFOR ∪ ENDF

EXFOR ∩ ENDF

Plot

Axes: x:☑log y:☐log. Ranges:x:2.911482 60 y:-13.36204 197.0191 Plot size(px):750 500 Repaint



EE-View Experimental-Evaluated data Viewer

Angular distribution

2.7sec

04:18

EE-VIEW

Experimental-Evaluated data Viewer //angular distributions

/under development by V.Zerkin, IAEA, 2022-2023, ver.2023-02-16/

Projectile n | *Target* Mn-55 | *Emission* el | *Inc.Energy* 3.5 MeV | *Libraries* EXFOR | *Options* Evaluated curves with error band

Get data | 3) exp:80/0s eval:5/2.6s plot/0.1s all/2.7sec

Select

Mn-55(n,el) Reset Plot Ei=3.5MeV

1.	ENDF: MN-55(N,EL)MN-55-L0,DA MF:4 MT:2			
1.	ENDF/B-VIII.0	20111222	IAEA Evaluation ...	[137] Ei:3.5e6
2.	JEFF-3.3	20171231	IAEA Evaluation ...	[46] Ei:3.5e6
3.	JENDL-5	20210607	N.Iwamoto	[82] Ei:3.5e6
4.	BROND-3.1	20111222	IAEA Evaluation ...	[137] Ei:3.5e6
5.	CENDL-3.2	950817	B.S.Yu+	[74] Ei:3.5e6
	1) EXFOR: 25-MN-55(N,EL)25-MN-55,,DA			
1)	22155082	1992	A.Takahashi Ei:1.41e7	[16] An:15÷160
2)	21722038	1972	I.Fujita Ei:1.41e7	An=110
3)	20019082	1969	B.Holmqvist Ei:2.47e6	[19] An:19.9÷160
4)	20019082	1969	B.Holmqvist Ei:3.00e6	[20] An:19.9÷160
5)	20019082	1969	B.Holmqvist Ei:3.49e6	[20] An:19.9÷160
6)	20019082	1969	B.Holmqvist Ei:4.00e6	[20] An:19.9÷160
7)	20019082	1969	B.Holmqvist Ei:4.56e6	[20] An:19.9÷160
8)	20019082	1969	B.Holmqvist Ei:6.09e6	[19] An:19.9÷160
9)	20019082	1969	B.Holmqvist Ei:7.05e6	[19] An:19.9÷160
10)	20019082	1969	B.Holmqvist Ei:8.05e6	[19] An:19.9÷160
11)	11511008	1966	R.L.Becker Ei:3.20e6	[14] An:20÷150
12)	11519005	1966	S.A.Cox Ei:6.77e5	[8] An:20÷160
13)	11519005	1966	S.A.Cox Ei:6.86e5	[8] An:20÷160
14)	11519005	1966	S.A.Cox Ei:6.94e5	[8] An:20÷160
15)	11519005	1966	S.A.Cox Ei:7.03e5	[8] An:20÷160
16)	11519005	1966	S.A.Cox Ei:7.11e5	[8] An:20÷160
17)	11519005	1966	S.A.Cox Ei:7.19e5	[8] An:20÷160
18)	11519005	1966	S.A.Cox Ei:7.28e5	[8] An:20÷160
19)	11519005	1966	S.A.Cox Ei:7.36e5	[8] An:20÷160
20)	11519005	1966	S.A.Cox Ei:7.45e5	[8] An:20÷160
21)	11519005	1966	S.A.Cox Ei:7.53e5	[8] An:20÷160
22)	11519005	1966	S.A.Cox Ei:7.61e5	[8] An:20÷160
23)	11519005	1966	S.A.Cox Ei:7.70e5	[8] An:20÷160
24)	11519005	1966	S.A.Cox Ei:7.78e5	[8] An:20÷160
25)	11519005	1966	S.A.Cox Ei:7.87e5	[8] An:20÷160
26)	11519005	1966	S.A.Cox Ei:7.95e5	[8] An:20÷160
27)	11519005	1966	S.A.Cox Ei:8.03e5	[8] An:20÷160
28)	11519005	1966	S.A.Cox Ei:8.12e5	[8] An:20÷160

ENDF: datasets:5, data points:476, Energy(MeV):1.8e-4÷0.e0

EXFOR: reactions:2, datasets:80, data points:740, E(MeV):1.5e-5÷1.6e-4

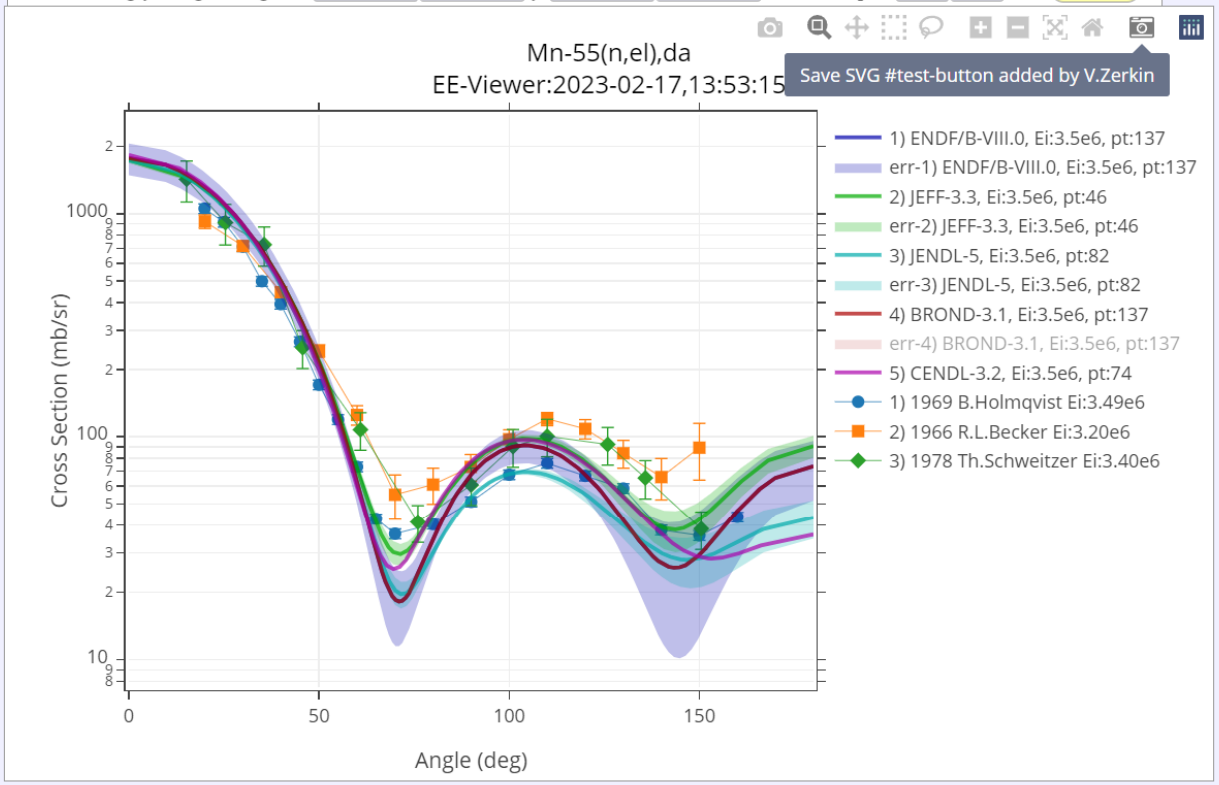
Download selected EXFOR data: [csv] [csv+]

Plotted data: Copy Paste

Plot

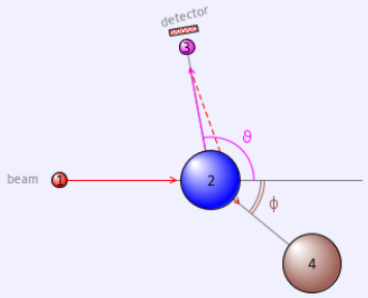
Axes: x:log y:log. Ranges:x: 0 180 y: 7.546586 2770.051 Plot size(px): 780 500 Repaint

Save SVG #test-button added by V.Zerkin



Point #9) x=100 y=97.4617 dy=9.93493 Dataset #2) 1966 R.L.Becker Ei:3.20e6

Point #9) Dataset #2) 1966 R.L.Becker Ei:3.20e6



25-MN-55(N,EL)25-MN-55,,DA Qvalue=0.0(keV)

1 Projectile:	n	M_1=1.008665
2 Target:	Mn-55	M_2=54.938046
3 Scattered:	n	M_3=1.008665
4 Recoil:	Mn-55	M_4=54.938046
	Laboratory System	
E_1=3200.0	E_3=3065.0	θ=100.0° σ(θ)=97.4617 ±10.2%
E_2=0.0	E_4=134.995	φ=39.5° σ(φ)=302.871
	Center of Mass System	
E_{cm}=3142.31		
E_1'=3085.65	E_3'=3085.65	θ'=101.0° σ'(θ')=98.1023
E_2'=56.6528	E_4'=56.6528	φ'=79.0° σ'(φ')=98.1023

Units: M:(amu); E:(keV); σ:(mb/sr)

Statistics of usage: visits: 652, requests: 998, since 01-Feb-2023

Created by V.Zerkin (v.zerkin@iaea.org), IAEA-NDS, 28-Dec-2022. Last updated:2023-02-16,12:01:53
Database and Programming: EXFOR/X4Pro/ENDF-Relational by V.Zerkin, IAEA-NDS, 1999-2023
Experimental Data Source: EXFOR, Network of Nuclear Reaction Data Centres (NRDC), 1970-2023
Evaluated Data Source: CSEWG, WPEC, IAEA-NDS, IPPE, CNDC, JAEA, NRG, CCFE, FZK

Thank you.

Citing of the materials of this presentation should be done with proper acknowledgement of the IAEA and author