`x4i`, The EXFOR interface

David Brown, March 4, 2011

Introduction

The `x4i` package is an interface to the EXFOR nuclear data library. It simplifies retrieval of EXFOR entries and can automatically parse them, allowing one to extract cross-section (and other) data in a simple, plot-able format. `x4i` also understands and can parse the entire reaction string, allowing one to build a strategy for processing the data.

EXFOR is a structured markup language for representing measured nuclear data. It is an old format, and is awkward to use for several reasons:

- It relies on data being in the correct columns in order to denote context. This is a legacy feature since EXFOR data used to be stored on FORTRAN punch cards.
- The data was often hand-entered so the format rules were not always rigorously obeyed (fortunately WPEC SubGroup 30 has remedied much of this ensuring that EXFOR data can be translated into C4 format, see ref. [1]).
- The mark-up language is surprisingly complex (see refs [2-5]).

Figures 1-5 illustrate the structure of the EXFOR format.

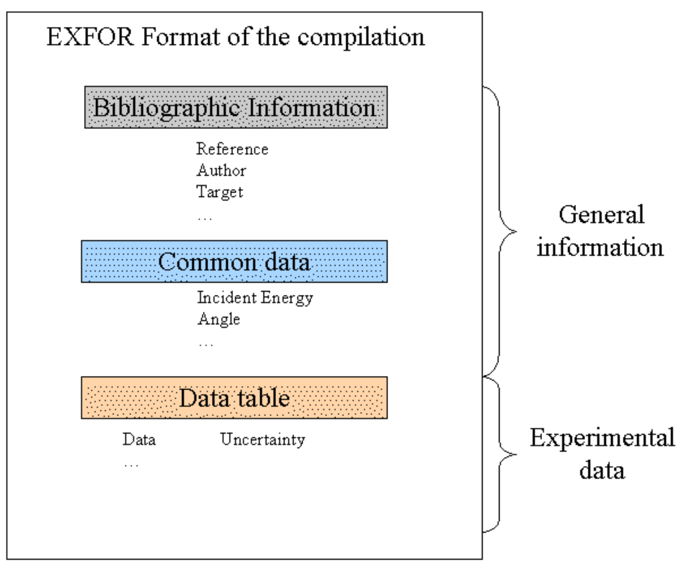
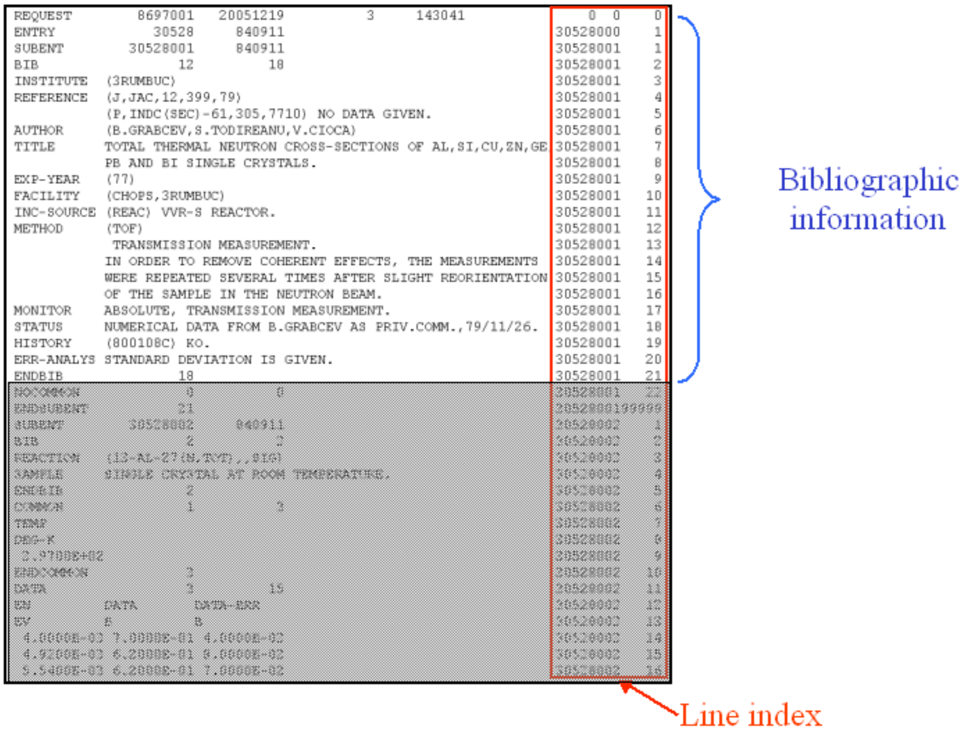
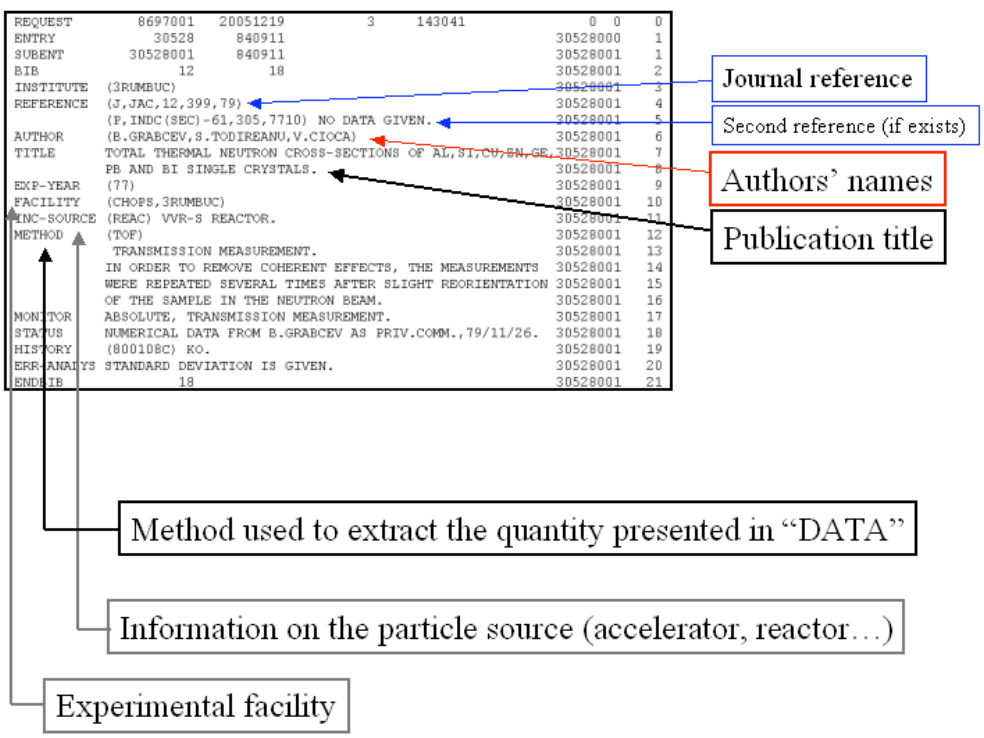


Figure 1. Structure of an EXFOR entry. The bibliographic information is always contained in the first subentry of an entry and given the index '001'. Common data is data common to all data block in all subentries and is found in the '001' subentry. Each dataset is given its own subentry, beginning with subentry '002.'



```
REQUEST         8697001   20051219          3      143041                      0  0      0
ENTRY             30528     840911                                         30528000      1
SUBENT         30528001     840911                                         30528001      1
BIB                  12         18                                         30528001      2
INSTITUTE   (3RUMBUC)                                                      30528001      3
REFERENCE   (J,JAC,12,399,79)                                              30528001      4
            (P,INDC(SEC)-61,305,7710) NO DATA GIVEN.                       30528001      5
AUTHOR      (B.GRABCEV,S.TODIREANU,V.CIOCA)                                30528001      6
TITLE       TOTAL THERMAL NEUTRON CROSS-SECTIONS OF AL,SI,CU,ZN,GE,        30528001      7
            PB AND BI SINGLE CRYSTALS.                                     30528001      8
EXP-YEAR    (77)                                                           30528001      9
FACILITY    (CHOPS,3RUMBUC)                                                30528001     10
INC-SOURCE  (REAC) VVR-S REACTOR.                                          30528001     11
METHOD      (TOF)                                                          30528001     12
             TRANSMISSION MEASUREMENT.                                     30528001     13
            IN ORDER TO REMOVE COHERENT EFFECTS, THE MEASUREMENTS          30528001     14
            WERE REPEATED SEVERAL TIMES AFTER SLIGHT REORIENTATION         30528001     15
            OF THE SAMPLE IN THE NEUTRON BEAM.                             30528001     16
MONITOR     ABSOLUTE, TRANSMISSION MEASUREMENT.                            30528001     17
STATUS      NUMERICAL DATA FROM B.GRABCEV AS PRIV.COMM.,79/11/26.          30528001     18
HISTORY     (800108C) KO.                                                  30528001     19
ERR-ANALYS  STANDARD DEVIATION IS GIVEN.                                   30528001     20
ENDBIB               18                                                    30528001     21
NOCOMMON              0          0                                         30528001     22
ENDSUBENT            21                                                    3052800199999
SUBENT         30528002     840911                                         30528002      1
BIB                   2          2                                         30528002      2
REACTION    (13-AL-27(N,TOT),,SIG)                                         30528002      3
SAMPLE      SINGLE CRYSTAL AT ROOM TEMPERATURE.                            30528002      4
ENDBIB                2                                                    30528002      5
COMMON                1          3                                         30528002      6
TEMP                                                                       30528002      7
DEG-K                                                                      30528002      8
 2.9700E+02                                                                30528002      9
ENDCOMMON             3                                                    30528002     10
DATA                  3         15                                         30528002     11
EN          DATA       DATA-ERR                                            30528002     12
EV          B          B                                                   30528002     13
 4.0000E-03 7.0000E-01 4.0000E-02                                          30528002     14
 4.9200E-03 6.2000E-01 8.0000E-02                                          30528002     15
 5.5400E-03 6.2000E-01 7.0000E-02                                          30528002     16
```


Figure 2. A close-up on the first subentry showing the bibliographic data.



```
REQUEST         8697001   20051219          3      143041                      0  0      0
ENTRY             30528     840911                                         30528000      1
SUBENT         30528001     840911                                         30528001      1
BIB                  12         18                                         30528001      2
INSTITUTE   (3RUMBUC)                                                      30528001      3
REFERENCE   (J,JAC,12,399,79)                                              30528001      4
            (P,INDC(SEC)-61,305,7710) NO DATA GIVEN.                       30528001      5
AUTHOR      (B.GRABCEV,S.TODIREANU,V.CIOCA)                                30528001      6
TITLE       TOTAL THERMAL NEUTRON CROSS-SECTIONS OF AL,SI,CU,ZN,GE,        30528001      7
            PB AND BI SINGLE CRYSTALS.                                     30528001      8
EXP-YEAR    (77)                                                           30528001      9
FACILITY    (CHOPS,3RUMBUC)                                                30528001     10
INC-SOURCE  (REAC) VVR-S REACTOR.                                          30528001     11
METHOD      (TOF)                                                          30528001     12
             TRANSMISSION MEASUREMENT.                                     30528001     13
            IN ORDER TO REMOVE COHERENT EFFECTS, THE MEASUREMENTS          30528001     14
            WERE REPEATED SEVERAL TIMES AFTER SLIGHT REORIENTATION         30528001     15
            OF THE SAMPLE IN THE NEUTRON BEAM.                             30528001     16
MONITOR     ABSOLUTE, TRANSMISSION MEASUREMENT.                            30528001     17
STATUS      NUMERICAL DATA FROM B.GRABCEV AS PRIV.COMM.,79/11/26.          30528001     18
HISTORY     (800108C) KO.                                                  30528001     19
ERR-ANALYS  STANDARD DEVIATION IS GIVEN.                                   30528001     20
ENDBIB               18                                                    30528001     21
```


Figure 3. An even closer look at the details of the bibliographic data. Here one can see how the authors’ names, institutions and publication information are specified.

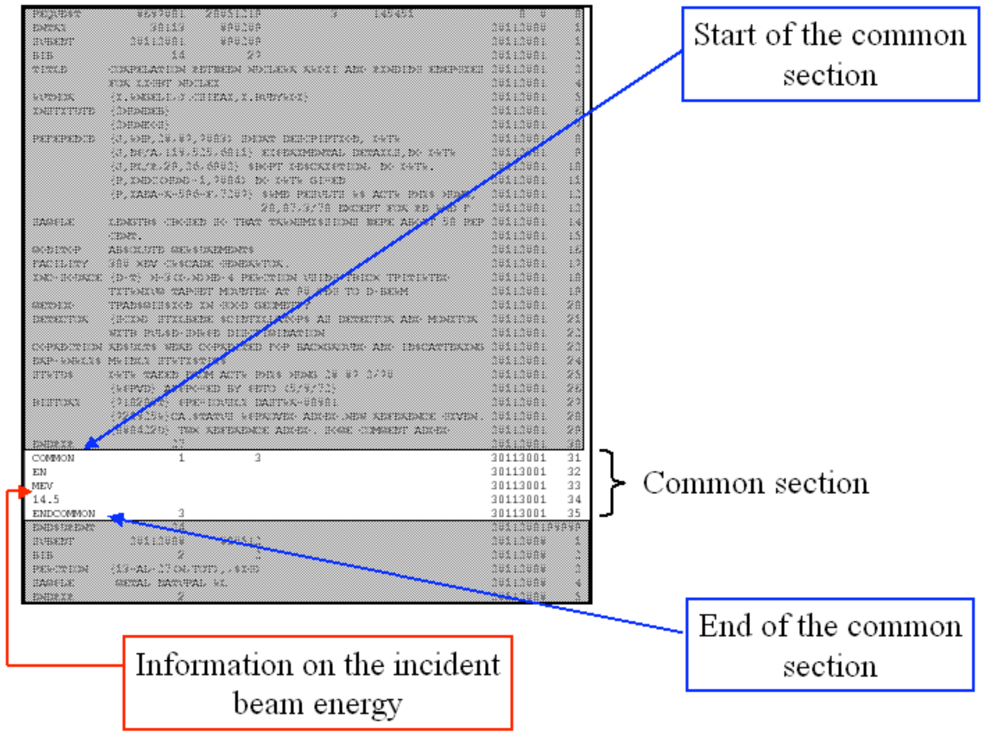


Figure 4. A sample COMMON data section. In this case, the beam energy for all data in subsequent subentries is given.

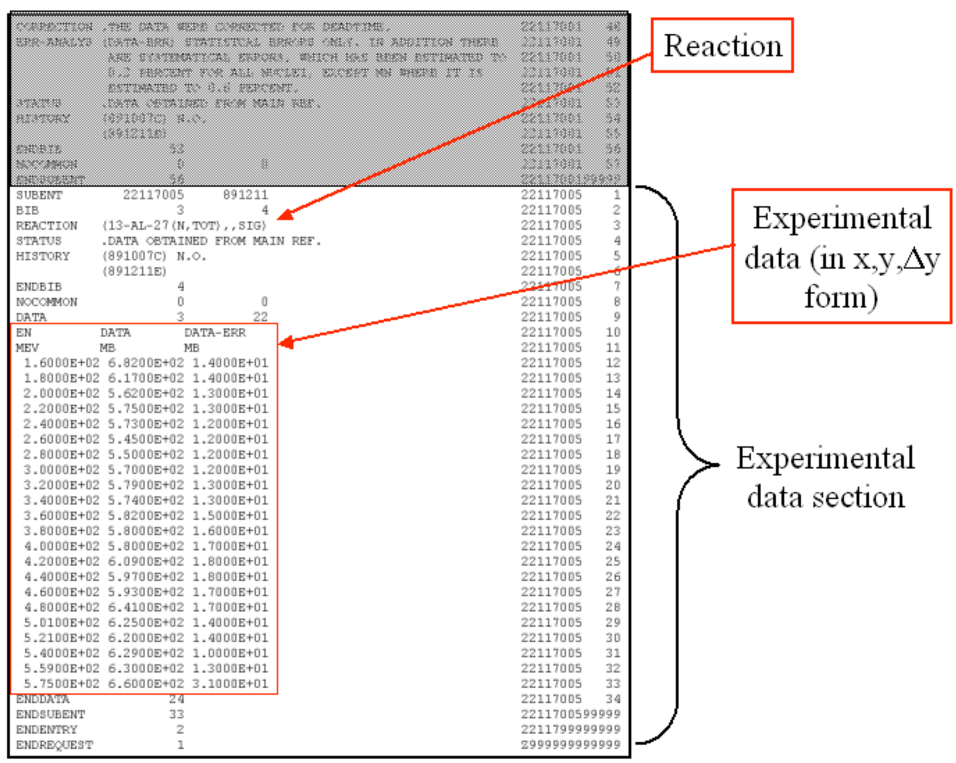


Figure 5. A sample DATA section. DATA sections contain the actual data from a measurement. This is combined with the data in COMMON data to produce an instance of the `X4DataSet` class detailed later in this report.

Installation

Installation of `x4i` is straightforward.

From the subversion repository:

- Checkout the code:
```
host$ svn co svn+ssh://username@ocfmachine.llnl.gov/usr/gapps/CNP_src/
all/live_repos/svnRepos/x4i/trunk/x4i
```
You must be a member of the `ndg` group on LLNL's OCF facility.
- Unpack the EXFOR data contained in the repository:
```
host$ python x4i/setupEXFORdb.py -u
```
- Put `x4i` in your `PYTHONPATH`.
- That's it!

From a tarball:

- Unpack the code:
```
host$ tar xzf xvi-1.0.tar.gz
```
- Put `x4i` in your `PYTHONPATH`
- That's it!

Basic usage

Now we describe how to use `x4i`. We begin by explaining how to query the EXFOR database and how to retrieve data. All retrievals and queries are handled by the classes in the `exfor_manager` module. The class `X4DBManagerDefault` defaults to the `X4DBManagerPlainFS` class and this is the class supported out of the box by `x4i`. Here is an example of its use:

```
host$ python
Python 2.6.4 (r264:75706, Dec 22 2009, 09:45:51)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from x4i import exfor_manager, exfor_entry
>>> db = exfor_manager.X4DBManagerDefault()
>>> help( db )
Help on instance of X4DBManagerPlainFS in module x4i.exfor_manager:

class X4DBManagerPlainFS(X4DBManager)
 |  Exfor data base manager for data stored on local filesystem in
directory hierarchy.
 |
 |  Methods defined here:
 |
 |  __init__(self, **kw)
 |
 |  query(self, author=None, reaction=None, target=None, projectile=None,
quantity=None, product=None, MF=None, MT=None, C=None, S=None, I=None,
SUBENT=None, ENTRY=None)
 |
 |  retrieve(self, author=None, reaction=None, target=None,
projectile=None, quantity=None, product=None, MF=None, MT=None, C=None,
S=None, I=None, SUBENT=None, ENTRY=None)
```

```
|
. . .
```

Once the database manager is initialized, we can run a query:

```
>>> db.query(author='Panitkin')
{u'40177': [u'40177001', u'40177002', u'40177003'], u'40121':
[u'40121001', u'40121002'], u'40431': [u'40431001', u'40431002'],
u'41335': [u'41335001', u'41335002']}
```

All queries return a Python `dict`. The keys of the dictionary are the EXFOR entry number (the 'u' preceding the entry simple tells us that the key is encoded in Unicode). The values of the dictionary are a list of subentry numbers of the EXFOR entry whose contents match the query search criteria. If a particular subentry matches the search criteria, the corresponding documentation subentry (the '001' subentry) is also returned. The complete list of search criteria are given in Table 1. A partial list of searchable observables is given in Table 2.

Retrievals also can be made using the database manager:

```
>>> db.retrieve(author='Panitkin')
{u'40177': ["SUBENT        40177001   20021225   20030502   20050926
0000\nBIB                 12         44\nINSTITUTE  (4RUSFEI)\nREFERENCE
(J,AE,33,825,197210)     Table of Data and Graph are\n
Given\nAUTHOR     (YU.G.PANITKIN,V.A.TOLSTIKOV)\nTITLE       Radiative
. . .
```

The search result from a retrieval is identical to that of the queries except that the subentry number is replaced by a string containing the entire text of the subentry.

Translating data retrieved using `x4i` is also simple:

```
>>> x = db.retrieve(target='PU-
239',reaction='N,2N',quantity='SIG',author='Lougheed' )
>>> x.keys()
[u'13883']
>>> y = exfor_entry.X4Entry( x['13883'] )
```

Here we've run a retrieval to get some cross-section data and then inserted the entire entry 13883 into the constructor for the `X4Entry` class of the `exfor_entry` module. The X4Entry class instance (and its components) handle all of the parsing of the EXFOR entry.

In the following section, we will detail some of the things one can do with an `X4Entry` instance. For now, we'll just illustrate how to extract the cross-section data in a format we can plot:

```
>>> dss = y.getSimplifiedDataSets()
>>> dss.keys()
[('13883', '13883002', ' ')]
>>> print dss[('13883', '13883002', ' ')]
#  Authors:   R.W.Lougheed, W.Webster, M.N.Namboodiri, D.R.Nethaway,
K.J.Moody, J.H.Landrum, R.W.Hoff, R.J.Dupzyk, J.H.Mcquaid, R.Gunnink,
E.D.Watkins
#  Title:     239Pu And 241Am(N,2N) Cross-Section Measurements Near E(N)
= 14 Mev
#  Year:      2002
#  Institute: Lawrence Livermore National Laboratory, Livermore, CA
#  Reference: Radiochimica Acta 90, 833 (2002)
#  Reaction:  Cross section for 239Pu(n,2n)238Pu
#        Energy        Data          d(Data)
```

```
#         MeV           barns          barns
         13.8           0.228          0.006384
         14.0           0.219          0.007884
         14.8           0.214          0.002996
>>> open('plotfile.dat',mode='w').writelines(str(dss[('13883','13883002',
' ')]))
```

What we've done here is extract all the datasets in our `X4Entry` instance using the `getSimplifiedDataSets()` member function. The results are stored in another Python `dict`, this time keyed off with a Python `tuple` with the following structure: `(entry #, subentry #, pointer)`. In this case, there is no pointer so that spot is taken by a string comprising a single space character. In other cases, the pointer may be number either referring to additional data. We will explain this further in the next sections.

Search Criteria	**Details**	**Implemented in version 1.0**
`author`	Only one author may be specified and only the family name should be given. Proper capitalization must be used.	Yes
`reaction`	Enter in form "projectile,products," e.g. `N,2N` or `N,F` or `D,3N+P`. Wildcards may be used, e.g. `*,2N`.	Yes
`target`	Enter in form "SYM-Z," e.g. `HE-3`. The symbol should be in upper case.	Yes
`projectile`	The standard ENDL set are supported, namely: `N`, `P`, `D`, `T`, `A`, `G`, `HE-3`. Additionally, the projectile may be any nucleus of form "SYM-Z" (provided such heavy-ion data exists in EXFOR).	Yes
`quantity`	This defines the observable, e.g. cross-section is `SIG`. Table 2 lists the supported quantities.	Yes
`product`	Residual nucleus (if any) of a particular reaction. Enter in form "SYM-Z," e.g. `HE-3`. The symbol should be in upper case.	Yes, partially
`MF`	The ENDF quantity, e.g. MF=3 is cross-section data.	No
`MT`	The ENDF reaction, e.g. MT=18 is fission.	No
`C`	The ENDL reaction, e.g. C=12 is (n,2n).	No
`S`	The ENDL reaction modifier, e.g. S=1 denotes discrete level excitations.	No
`I`	The ENDL quantity, e.g. I=1 denote angular probability distributions, $P(E\|\mu)$.	No
`SUBENT`	The EXFOR Subentry number. It is 8 characters long and the last 3 digits specify the subentry within the EXFOR entry corresponding to the first 5 characters.	Yes
`ENTRY`	The EXFOR Entry number. It is 5 characters long.	Yes

Table 1. Valid search keys for queries and retrievals from the EXFOR database manager classes.

Quantity	Details	Variations on quantity supported	Simplified translation of data available
`DA`	Angular distribution $d\sigma(E)/d\mu$	`EVAL`	Yes
`DA/DE`	Double differential data $d\sigma(E)/d\mu dE'$		No, high priority
`DE`	Energy distribution $d\sigma(E)/dE'$	`EVAL`	Yes
`FY`	Fission yields		No
`NU`	Average number of neutrons emitted in fission event $\bar{\nu}(E)$	`EVAL`, `PR`	Yes
`NU/DE`	Fission neutron spectrum $d\bar{\nu}(E)/dE'$		No
`POL/DA`	Polarization		No, high priority
`POT`	Potential scattering parameter		No
`RI`	Resonance integral of cross-section		No
`SIG`	Cross-section $\sigma(E)$ or average cross-section in some variations of this observable.	`EVAL`, `MXW`, `SPA`, `FST`, `RTE`, `FIS`, `AV`	Yes

Table 2. A selection of supported quantities. The full list is given in EXFOR dictionary 30 (see ref. [3])

The `X4Entry` class

In this section, we provide a more detailed look into the `X4Entry` class and its use. A partial list of member functions is provided in Table 3.

Let us begin the discussion by picking up where we left off in the previous section's example. We return to the `X4Entry` in the Python variable '`y`':

```
>>> y = exfor_entry.X4Entry( x['13883'] )
>>> y.keys()
['13883001', '13883002']
>>> type( y[1] )
<class 'x4i.exfor_subentry.X4SubEntry'>
```

In this simple example, we have illustrated that `X4Entry`s are really Python `dict`s, with keys being the subentry accession number (in this case, abbreviated to '1') and values being instances of the `X4SubEntry` class. Note that the subentry accession numbers 1, '1', '001', 13883001, and '1388301' are all equivalent. Continuing:

```
>>> y['1'].keys()
['BIB']
>>> y['1']['BIB'].keys()
['STATUS', 'REFERENCE', 'FACILITY', 'INSTITUTE', 'TITLE', 'INC-SOURCE',
'AUTHOR', 'HISTORY']
>>> y['1']['BIB']['REFERENCE']
REFERENCE  (J,RCA,90,833,2002)
>>> str(y['1']['BIB']['REFERENCE'])
'Radiochimica Acta 90, 833 (2002)'
```

Clearly `X4Entry`s and `X4SubEntry`s are simply nested Python `dict`s whose keys and values correspond to the structure of the original EXFOR (sub)entry. This example illustrates one other point: the Python `str()` operator returns a "pretty" version of what it acts on. In this case, the reference field of the bibliography section of subentry #1.

Function	Arguments (other than `self`)	Description
`__str__()`		Enables Python `str()` function: the "pretty" string formatter. Recursively applies `str()` to all components of `self`.
`__repr__()`		Enables Python `repr()` function: the "representation" string formatter (strings returned by this function are nearly equivalent to the original EXFOR entry). Recursively applies `repr()` to all components in `self`.
`__getitem__( key )`	`key`	Enables element access with the `[ ]` operator (e.g. `[ key ]`)Return the `X4SubEntry` instance with subentry number `key`.
`deleted()`		Returns `True` if this entry has been deleted (a skeletal version of the entry remains in the EXFOR database though).
`getDataSets()`		Returns a Python `dict` containing all of the `X4DataSets` contained in `self`. The keys of the `dict` are a `tuple`: `(entry #, subentry #, pointer).`
`getSimplifiedDataSets()`	`makeAllColumns = False`	Similar to `getDataSets` except that data has been parsed (if possible), producing a simpler dataset that may be interpreted easier (and plotted!). See `X4DataSet` below.
`meta()`		Return an instance of the meta data derived from `self`.
`meta().citation()`		Returns a string containing the citation for the current entry. Suitable for publication.
`meta().legend()`		Returns a string containing information for the current entry. Suitable for use as a plot legend.
`meta().xmgraceHeader()`		Returns a string containing information for the current entry. Use this as the header for a dataset you are plotting in xmgrace.

Table 3. Member function reference for the `X4Entry` class and the `X4EntryMetaData` class. Functions in the `X4EntryMetaData` class are prefixed with the `meta()` call from the `X4Entry` class.

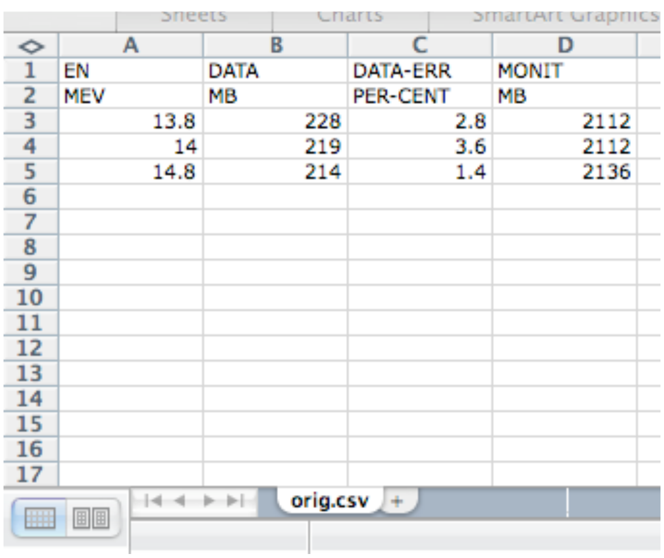
There are two other functions to elaborate, the `getSimplifiedDataSets()` and `getDataSets()` function. Both return `dicts` whose values are `X4DataSets`, either "plain" or "simplified." In the next section we will describe the `X4DataSet` and explain the difference between a "plain" `X4DataSet` and a "simplified" `X4DataSet`. Here we illustrate the use of either function:

```
>>> dss = y.getSimplifiedDataSets()
>>> dss.keys()
[('13883', '13883002', ' ')]
```

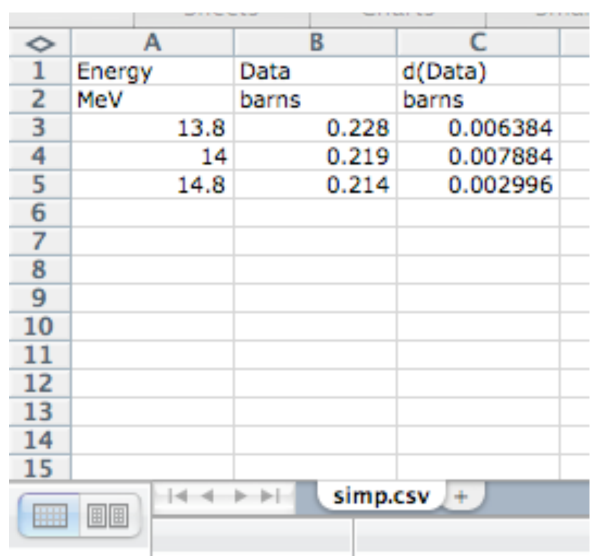
This function returns a Python `dict` whose keys are a `tuple`: `(entry #, subentry #, pointer)`. The EXFOR pointer here is a string consisting of a space. In many EXFOR subentries, the EXFOR compilers chose to store multiple datasets. To distinguish them (and to map the data to other fields in the EXFOR entry), the compilers gave the sets a distinct one character pointer. When `x4i` encounters such a case, the `dict` returned from `getSimplifiedDataSets()` will have one key per pointer.

The `X4DataSet` class

The `X4DataSet` class and its subclasses are probably the class most users of `x4i` will become familiar with first as instances of these classes contain the experimental data one wishes to plot and/or manipulate. In the previous section we introduced two functions in the `X4Entry` class that return `dicts` containing `X4DataSets`. We also introduced the concept of "plain" and "simplified" datasets. Figure 6 shows the csv output of the `X4DataSet`, retrieved in the previous section, in its "plain" form and its "simplified" form. As one can see, both contain the same data, but the "simplified" set is in consistent units and extraneous columns have been removed.



	A	B	C	D
1	EN	DATA	DATA-ERR	MONIT
2	MEV	MB	PER-CENT	MB
3	13.8	228	2.8	2112
4	14	219	3.6	2112
5	14.8	214	1.4	2136



	A	B	C
1	Energy	Data	d(Data)
2	MeV	barns	barns
3	13.8	0.228	0.006384
4	14	0.219	0.007884
5	14.8	0.214	0.002996

Figure 6. Difference between a "plain" X4DataSet (on the left) and a "simplified" X4DataSet (on the right). Note that the units for the simplified set are consistent between a data column and an uncertainty column. Also notice that cross sections are always given in barns and energies in MeV. Table 4 lists all the units supported in "simplified" `X4DataSets`.

Column Label	Units	Comments
Data	barns, barns/ster, 1/MeV ptcls/fis, no-dim	Unit choice depends on nature of the observable. Maybe dimensionless if data is ratio data.
Energy	MeV	Incident energy
E'	MeV	Outgoing energy
Angle	degrees	

Table 4. Column names and units in simplified `X4DataSet`'s.

As one can see in Figure 6, one can think of an X4DataSet as a spreadsheet containing the dataset's values. Indeed, the Python `__getitem__` operator allows us to directly access elements in this spreadsheet:

```
>>> myset = ds[ ('13883', '13883002', ' ') ]
>>> myset[ 'LABELS', 0]
'EN'
>>> myset[ 'LABELS', 1 ]
'DATA'
>>> myset[ 'UNITS', 1 ]
'MB'
>>> myset[ 0, 1 ]
228.0
```

Of course, `X4DataSets` also come with meta data describing the set:

```
>>> myset.legend()
'(2002) R.W.Lougheed, W.Webster, et al.'
>>> myset.citation()
'R.W.Lougheed, W.Webster, et al., Radiochimica Acta 90, 833 (2002);  Data
taken from the EXFOR database, file EXFOR ?????.??? dated 2002, retrieved
from the IAEA Nuclear Data Services website.'
```

Next, we point out the two methods for exporting the data, the `csv()` and the `str()` functions. The `csv()` function exports the dataset to a comma separated value file, suitable for viewing in Microsoft Excel. The `str()` function returns a string that can be viewed in the xmgrace plotting package. The complete list of member functions for the `X4DataSet` class is given in Table 5.

Finally, we want to elaborate on the implementation of "simplified" `X4DataSets`. When the `getSimplifiedDataSets()` function is called from, it in turn calls the `getDataSets()` function to get all of the data in an `X4Entry`. Then, the `X4DataSet` function `getSimplified()` is called to attempt to convert the `X4DataSet` into its simpler form. Currently very few quantities in EXFOR can be converted to simpler forms. The list as of version 1.0 of `x4i` is given in Table 2.

Function	Arguments (other than `self`)	Description
`__str__()`		Enables Python `str()` function: the "pretty" string formatter.
`__repr__()`		Enables Python `repr()` function: the "representation" string formatter (strings returned by this function are nearly equivalent to the original EXFOR).
`__getitem__((i,j))`	`i,j`	Access the data element in row `i` column `j`. If `i` = '`LABELS`' or '`UNITS`', then the corresponding string heading the column is returned.
`citation()`		Returns a string containing the citation for the current entry. Suitable for publication.
`csv( f )`	f	Writes data in `self` to file `f` in CSV format. The CSV format stands for "Comma Separated Value" and may be read by MS Excel.
`getSimplified( )`	`makeAllColumns = False, failIfMissingErrors = False`	Returns an X4DataSet that has been "simplified." See the main text for what that entitles. If the optional argument `makeAllColumns` is `True`, every data column will be accompanied by an uncertainty column even if one is not present in the original data. If the optional argument `failIfMissingErrors` is `True`, an exception will be raised if there is no uncertainty column accompanying one or more data columns.
`legend()`		Returns a string containing information for the current entry. Suitable for use as a plot legend.

Table 5. Member function reference for the `X4DataSet` class and subclasses.

Changing/upgrading the source database

To update or change the source database, you will need a copy of the new database from the IAEA. It is available as a zipfile downloaded from the IAEA website: http://www-nds.iaea.org/x4toc4-master/. There are two sets of files there. Those with the name of the form `X4-releasedate.zip` are the ones usable by `x4i`.

To install the IAEA library, assuming that your zip file is named `X4-releasedate.zip`:

```
host$ python x4i/setupEXFORdb.py –iX4-releasedate.zip
```

Please read the help message (`python setupEXFORdb.py –h`) for more information.

Bibliography

[1] A. Koning, “WPEC Subgroup 30: Quality improvement of the EXFOR database Status report June 2009,” NEA report number NEA/NSC/WPEC/DOC(2009)416 (2009).
[2] O. Schwerer, “LEXFOR,” IAEA Nuclear Data Section report number IAEA-NDS-208, Vienna, Austria (2008).
[3] O. Schwerer, “EXFOR Exchange Formats Manual,” IAEA Nuclear Data Section report number IAEA-NDS-207, Vienna, Austria (2008).
[4] O. Schwerer, “EXFOR/CINDA Dictionary Manual,” IAEA Nuclear Data Section report number IAEA-NDS-213, Vienna, Austria (2008).
[5] O. Schwerer, “EXFOR Basics Manual,” IAEA Nuclear Data Section report number IAEA-NDS-206, Vienna, Austria (2008).