



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

FUDGE: A program for performing nuclear data testing and sensitivity studies

Bret R. Beck

September 24, 2004

FUDGE: A Program for Performing Nuclear Data Testing and
Sensitivity Studies

Santa Fe, NM, United States

September 26, 2004 through October 1, 2004

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Nuclear Data Sensitivity Program: FUDGE

- FUDGE: acronym of “For Updating Data and Generating ENDL”
- Which data category should FUDGE allow users to modify?
 - Nuclear data categories:
 - Experimental/theoretical
 - Evaluated (e.g., ENDL99, ENDF/B-VI)
 - Processed data
 - Data put into a format useful for modeling codes
 - » Deterministic want grouped, Legendre expanded data (ndf at LLNL)
 - » Monte Carlo may want equal-probable binned data (MCAPM at LLNL)
 - FUDGE allows users to modify LLNL’s evaluated data and to process the modified data for use in modeling codes.

Nuclear Data Sensitivity Program: FUDGE - II

- LLNL's evaluated nuclear data format is called ENDL
 - Acronym for Evaluate Nuclear Data Library
 - ENDL format stores data as pointwise values
 - LLNL has evaluated nuclear databases containing data for neutron, proton, triton, deuteron, Helium-3, Helium and gamma as projectiles (also called incident particle).
- Most other databases use the ENDF format
 - ENDF uses a combination of pointwise and parametric form to represent data
 - Some other databases are: (ENDF/B-VI, JEFF, JENDL)
 - Other databases typically only have data for neutron as a projectile
- We have programs to convert ENDF formatted data into the ENDL format.

Nuclear Data Sensitivity Program: FUDGE - III

- FUDGE allows users to
 - Import ENDL formatted data.
 - Mix and match data from various databases.
 - Create their own pointwise data.
 - Print data.
 - Plot data.
 - Modify data.
 - Add, subtract, multiply and divide of data are supported.
 - Process modified data for use in modeling codes.
- Advantages of modifying evaluated data versus processed data.
 - Modeling codes do not have to change
 - Users only need to know one program (FUDGE) to modify data.
- FUDGE must know how to interact with processing codes.

FUDGE Design

- Requirements:
 - FUDGE must be user friendly.
 - FUDGE must be fast.
 - FUDGE must be cross platform.
 - FUDGE must be scriptable.
- Solution: We choose Python as interface language
 - Python is:
 - Object-oriented,
 - Interpreted,
 - Interactive programming language
 - Runs on all platforms
 - Comes standard on most UNIX operating systems.

Supported Data Types

- FUDGE has 4 data types. Each is a Python class.
 - 1d, 2d, 3d and 4d pointwise data classes
 - These classes are called `endl1dmath`, `endl2dmath`, `endl3dmath` and `endl4dmath`.
 - These classes are designated by the number of columns needed to represent the data.
 - Cross-section data is $\sigma(E)$ which is 2 column data (2d), one for projectile energy E and the other for $\sigma(E)$.
 - $P(E,\mu)$ is 3 column data (3d). One each for E , μ and $P(E,\mu)$.
 - $\mu = \cos(\theta)$: θ is outgoing particles angle
 - $P(E,\mu)$ is the outgoing particles probability versus μ and projectile energy E .
 - All data types have printing and plotting methods (see examples).
 - `endl2dmath` class supports addition, subtraction, multiplication and division on numbers and other `endl2dmath` objects.

ENDL and FUDGE Hierarchy

- Level 1 (Top): Evaluation
 - ENDL99, ENDF/B-VI, JEFF, JENDL, etc
- Level 2: Projectile (a.k.a. incident particle)
 - Neutron, proton, gamma, etc
- Level 3: Target (a.k.a. isotope)
 - ${}^6\text{Li}$, ${}^{17}\text{O}$, ${}^{239}\text{Pu}$, etc
 - Labeled by “za” + 1000 * Z + A (e.g., ${}^{239}\text{Pu}$ is za094239)
- Level 4: Target Data
- Example
 - ENDL99
 - neutron
 - za094239
 - » Data (Cross-sections and spectra)

Starting FUDGE: A Simple Processing Example

The top class in FUDGE is the *endlProject* class that represents a database/incident particle hierarchy. For example, the Python expression

```
e = endlProject( database = "endl99", yi = "neutron" )
```

creates an *endlProject* and all default data will come from the neutron incident particle data in the "endl99" database. Data from a target can be read using *endlProject*'s *readZA* method. The following Python code reads in the cross-section data for ⁶Li from the endl99/neutron database. It then modifies the (n,t) cross-section, saves the modification and calls the processing codes to create a new deterministic transport data file.

```
from fudge import * # Get FUDGE routines
e = endlProject( database="endl99", yi="neutron" ) # Create a new endlProject
Li6 = e.readZA( 3006 ) # Read Li information
Li6.read( I = 0 ) # Read all Li cross-section data
xsec = Li6.findData( C = , I = 0 ) # Get reference to (n,t) cross-section
xsec_mod = 1.1 * xsec # Increase (n,t) cross-section by 10%
xsec.set( xsec_mod )
e.save() # Save modification
e.process( "ndf1", "ndf1.new" ) # Create a deterministic data file from modification
```

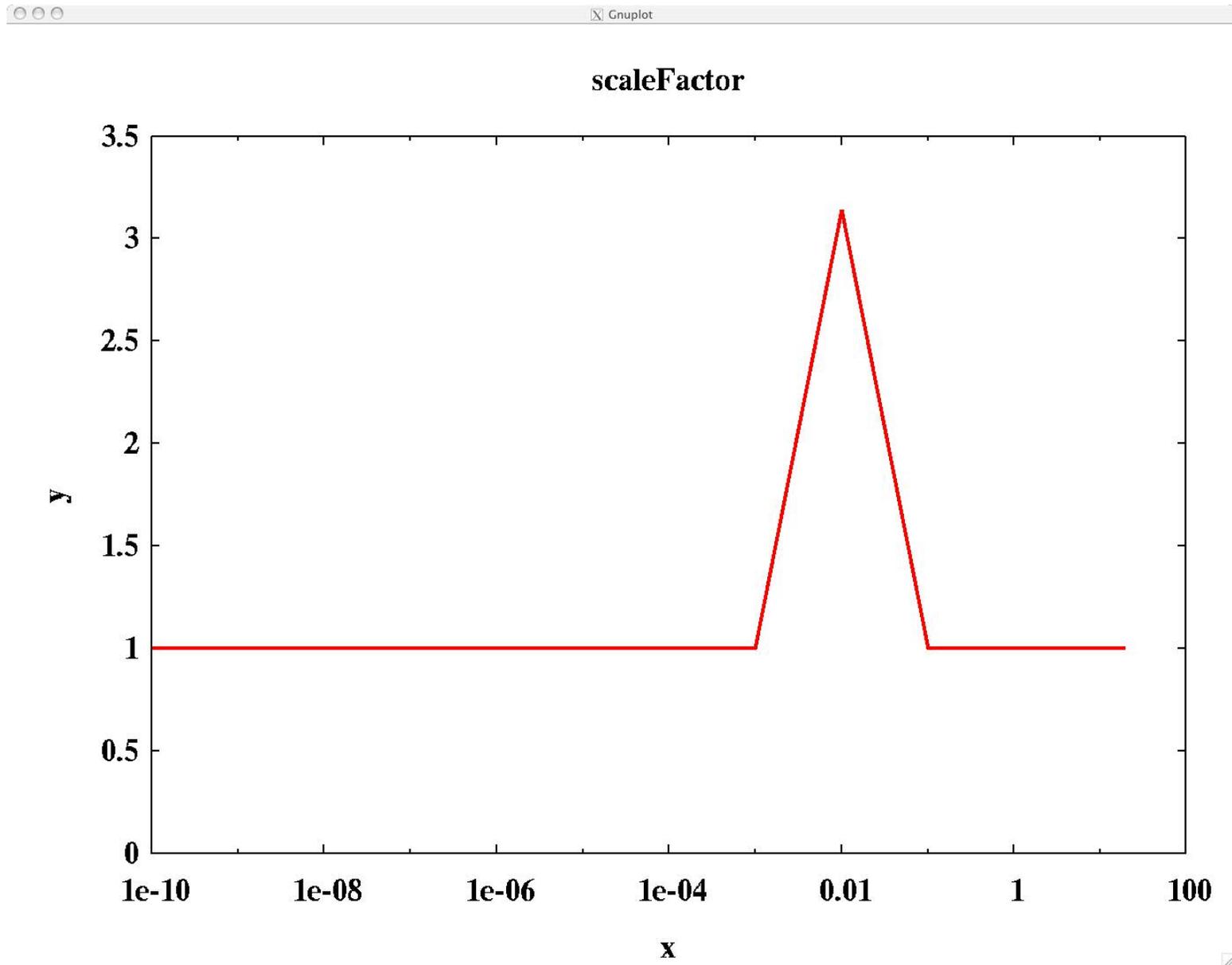
In the last line the processing routines are called. The first argument is the name of an exist processed data file that is to be used as a template. Parameters needed by the processing codes are obtained from this file. The second argument is the name of the processing file created using the modified data.

2d Computations with FUDGE: Example

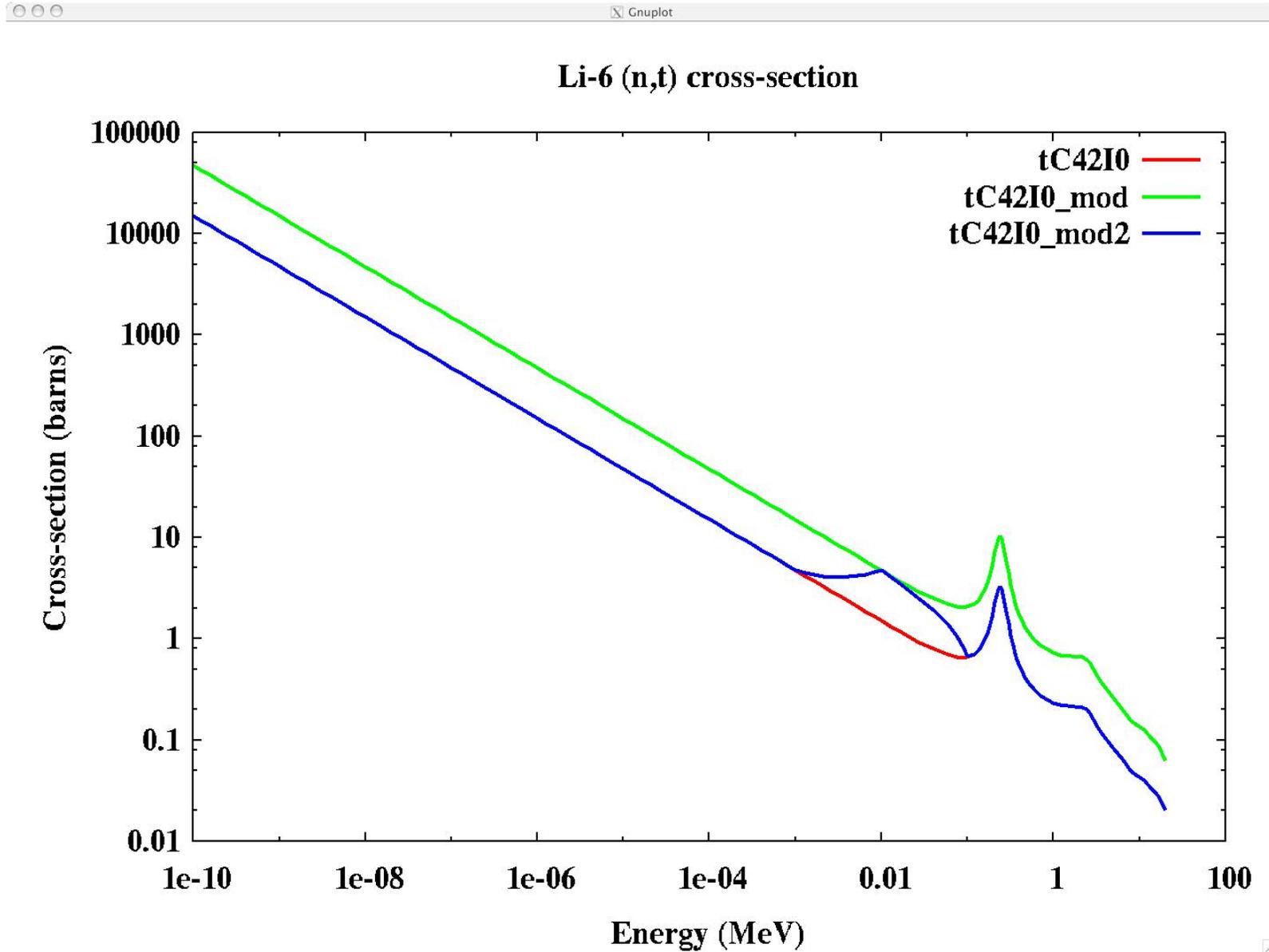
FUDGE can do addition, subtraction, multiple and division with endl2dmath objects. In this example I will first scale the $6\text{Li}(n,t)$ cross-section by 3.14. Then, I will scale it by another endl2dmath object labeled scaleFactor. A plot of scaleFactor is shown in on the page labeled “2d Computation example: Plot of scaleFactor”. A plot of the original cross-section and the two modified cross-sections are shown on the page labeled “2d Computation example: Li (n,t) Plot”.

```
from fudge import *
e = endlProject( database="endl99", yi="neutron" )
Li6 = e.readZA( 3006 )
Li6.read( I = 0 )
xsec = Li6.findData( C = , I = 0 )
xsec_mod = 3.14 * xsec
scaleFactor = endl2dmath( [ [ 1e-10, 1 ],
                             [ 1e-3, 1 ],
                             [ 1e-2, 3.14 ],
                             [ 1e-1, 1 ],
                             [ 20, 1 ] ] )
xsec_mod2 = scaleFactor * xsec
scaleFactor.plot()
qmultiPlot( [xsec , xsec_mod, xsec_mod2 ], legends = [ " xsec ", " xsec_mod", " xsec_mod2" ],
            title="Li-6 (n,t) cross-section", xylog = 3 )
```

2d Computation example: Plot of scaleFactor



2d Computation example: Li (n,t) Plot

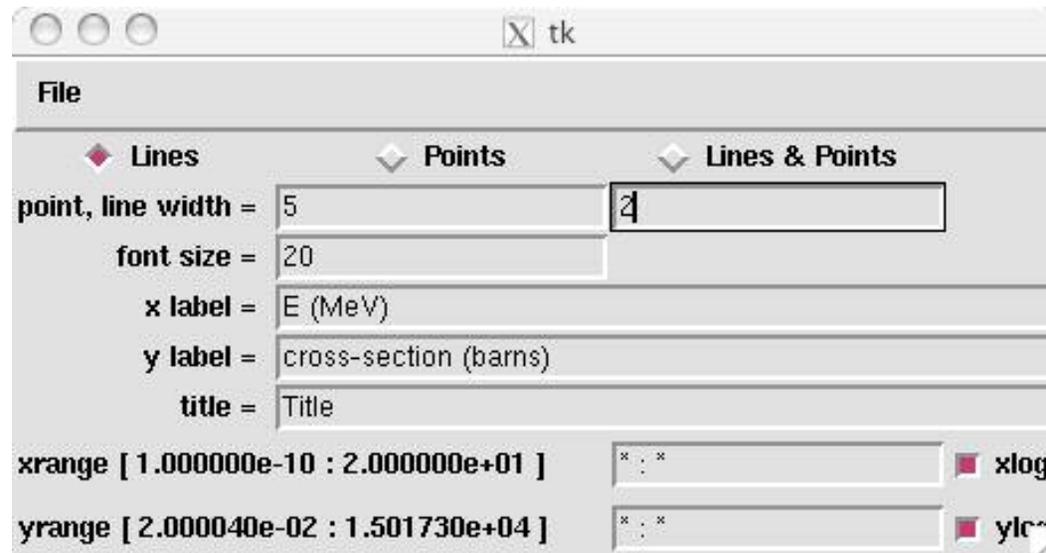


2d Plot Example and Dialog Window

```
from fudge import *
readOnly()

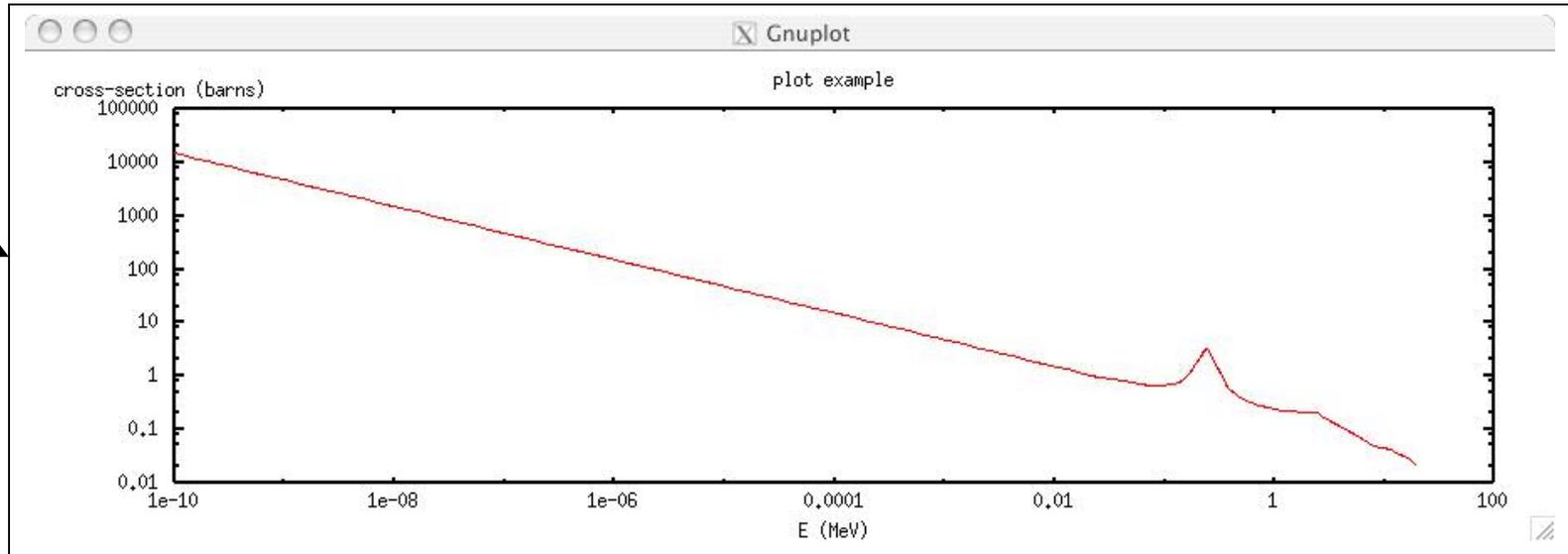
e = endlProject( database = "endl99", yi = "neutron" )
Li6 = e.readZA( 3006 )
Li6.read()
tC42I0 = Li6.findData( C = 42, I = 0 )
tC42I0.plot()
tC42I0.qplot()

b6 = endlProject( database = "endfb6_alpha16", yi = "neutron" )
Li6_b6 = b6.readZA( 3006 )
Li6_b6.read( I = 0 )
tC42I0_b6 = Li6_b6.findData( C = 42, I = 0 )
```

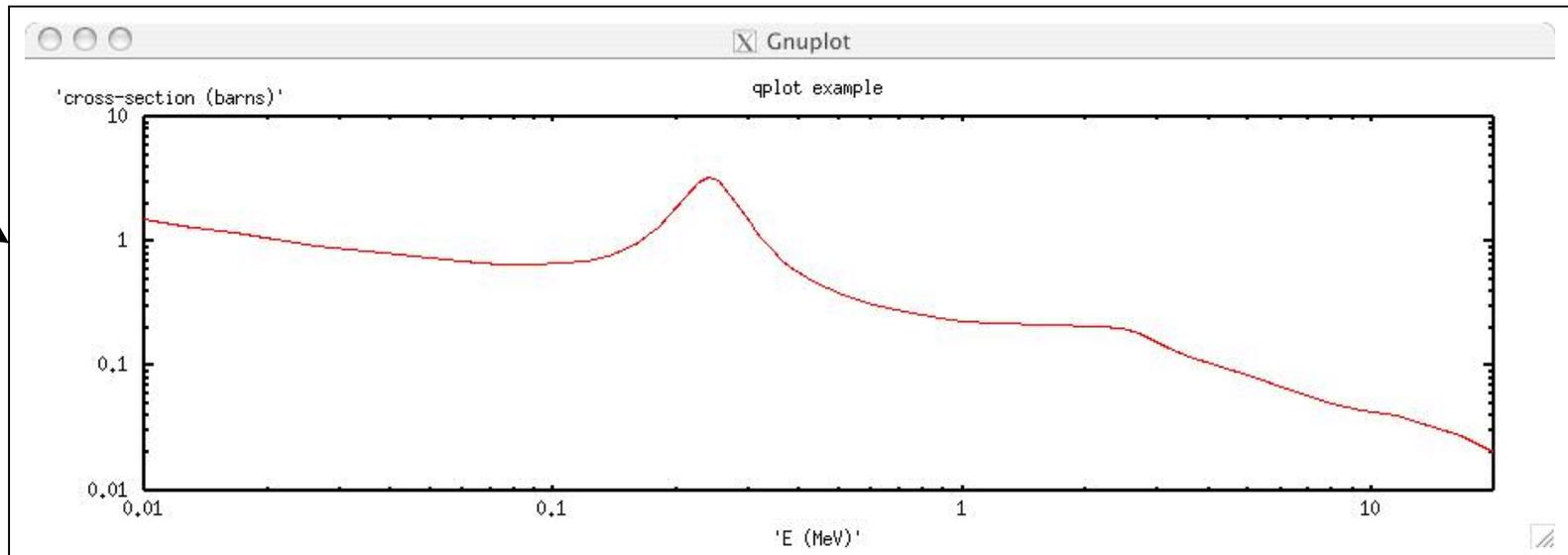


2d plot examples (plot vs qplot)

plot

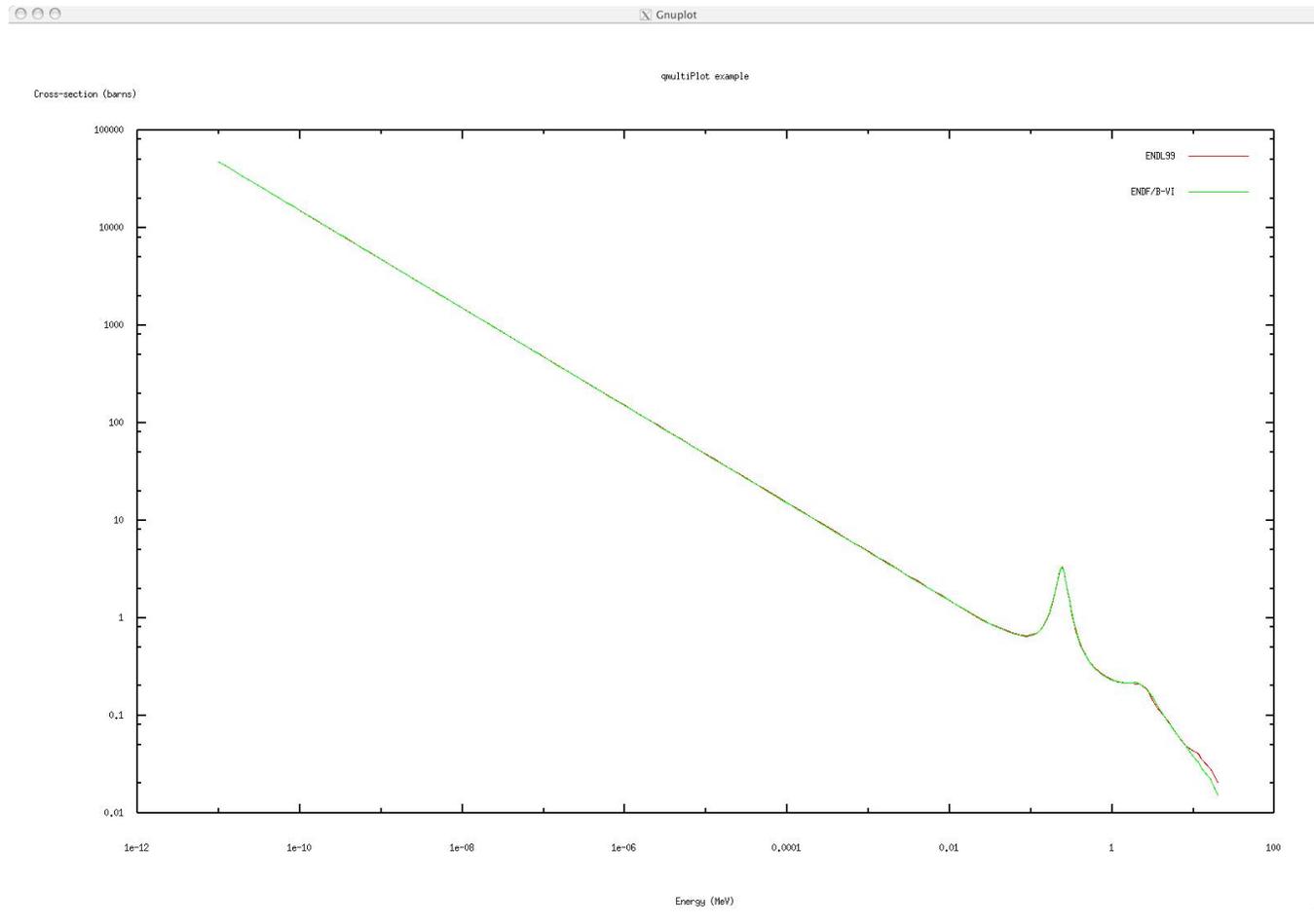


qplot



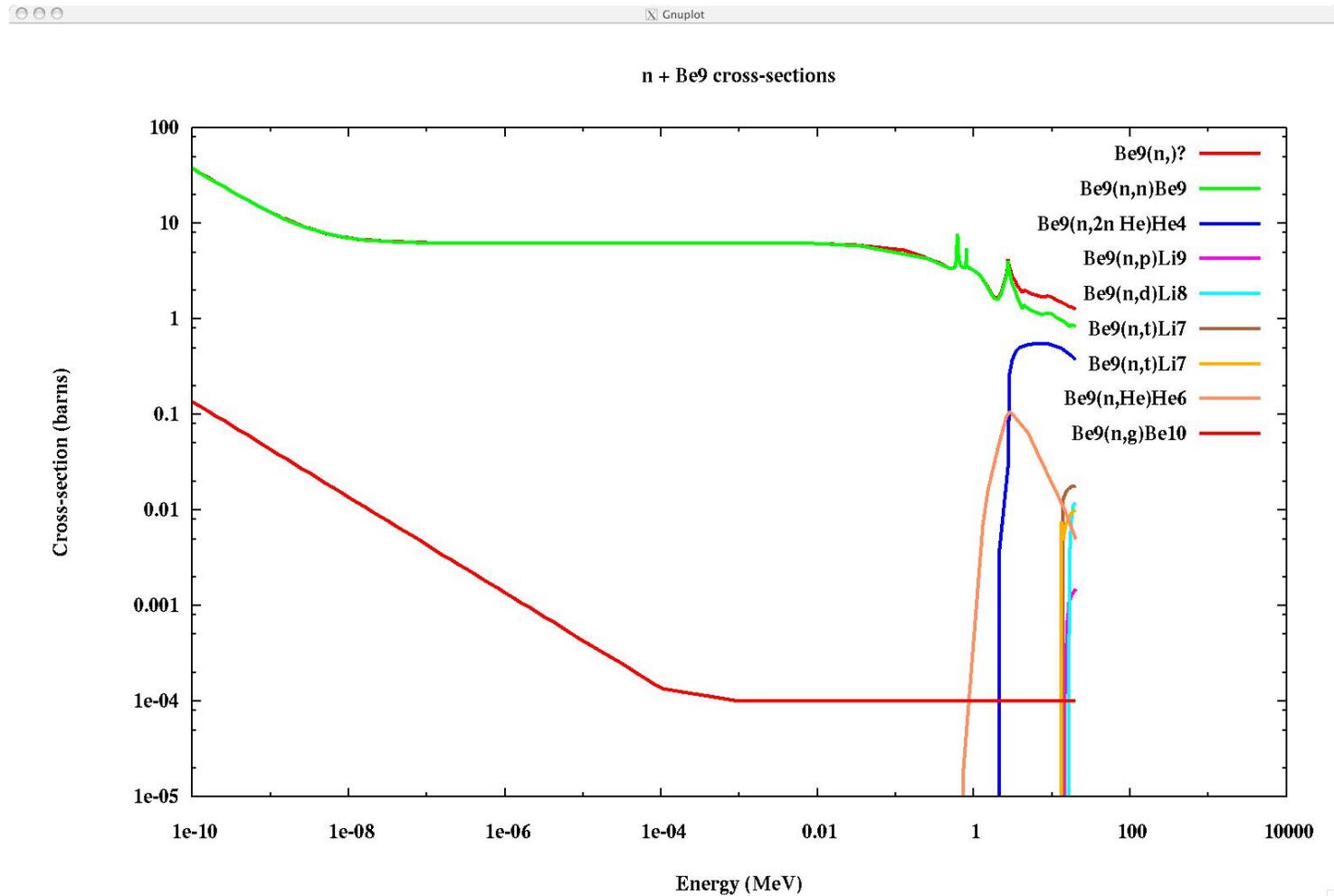
2d qmultiPlot example

```
tC42I0 = Li6.findData( C = 42, I = 0 )  
tC42I0_b6 = Li6_b6.findData( C = 42, I = 0 )  
qmultiPlot( [ tC42I0, tC42I0_b6 ], legends = [ "ENDL99", "ENDF/B-VI" ], xylog = 3, title = "qmultiPlot example" )
```



2d qmultiPlot example - cont.

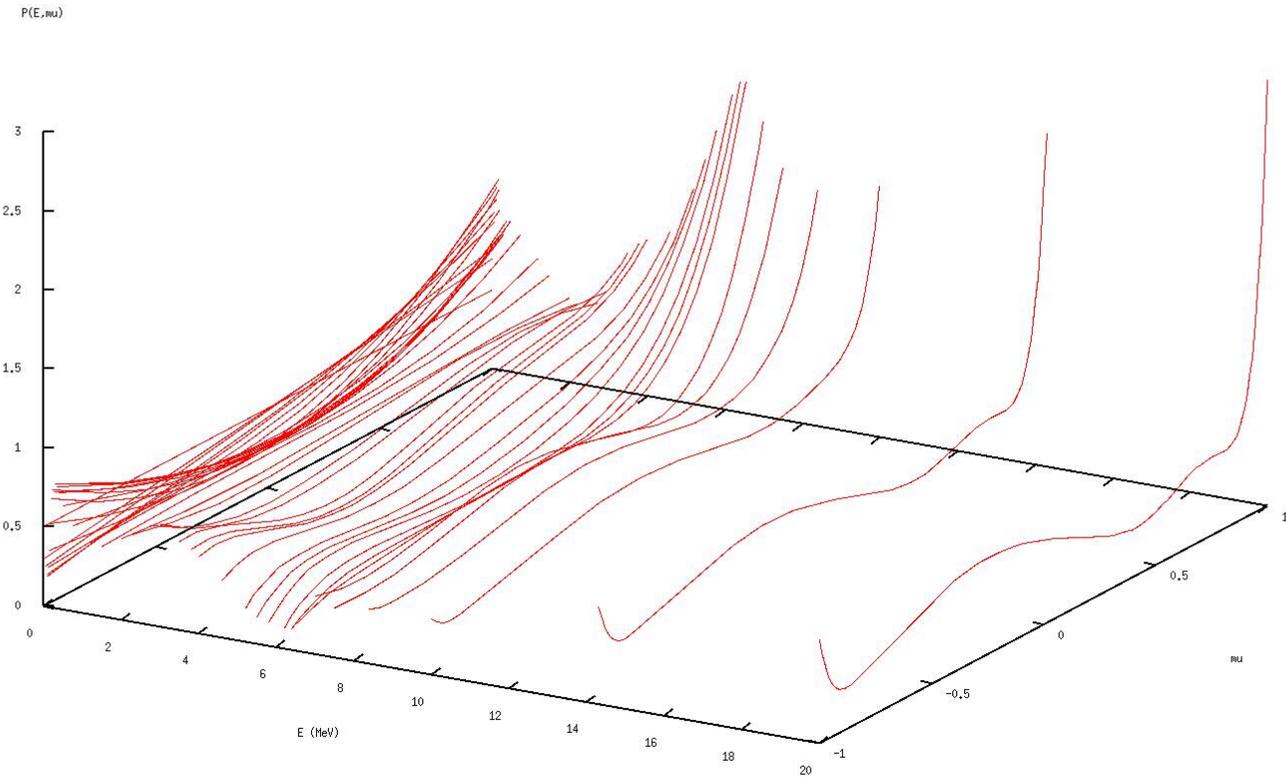
```
Be9AllxSec = Be9.findDatas( I = 0 )
l = []
for i in Be9AllxSec : l.append( reactionEquations( 1, 4009, i.C )[1] )
qmultiPlot( Be9AllxSec, xylog = 3, legends = 1, title = "n + Be9 cross-sections", xMax = 1000, \
            lineWidth = 4, fontSize = 36 )
```



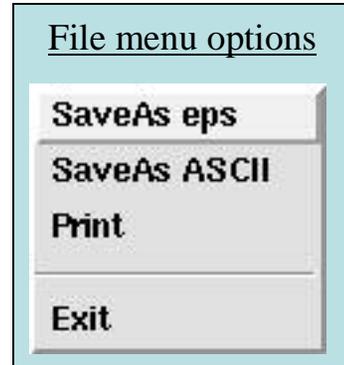
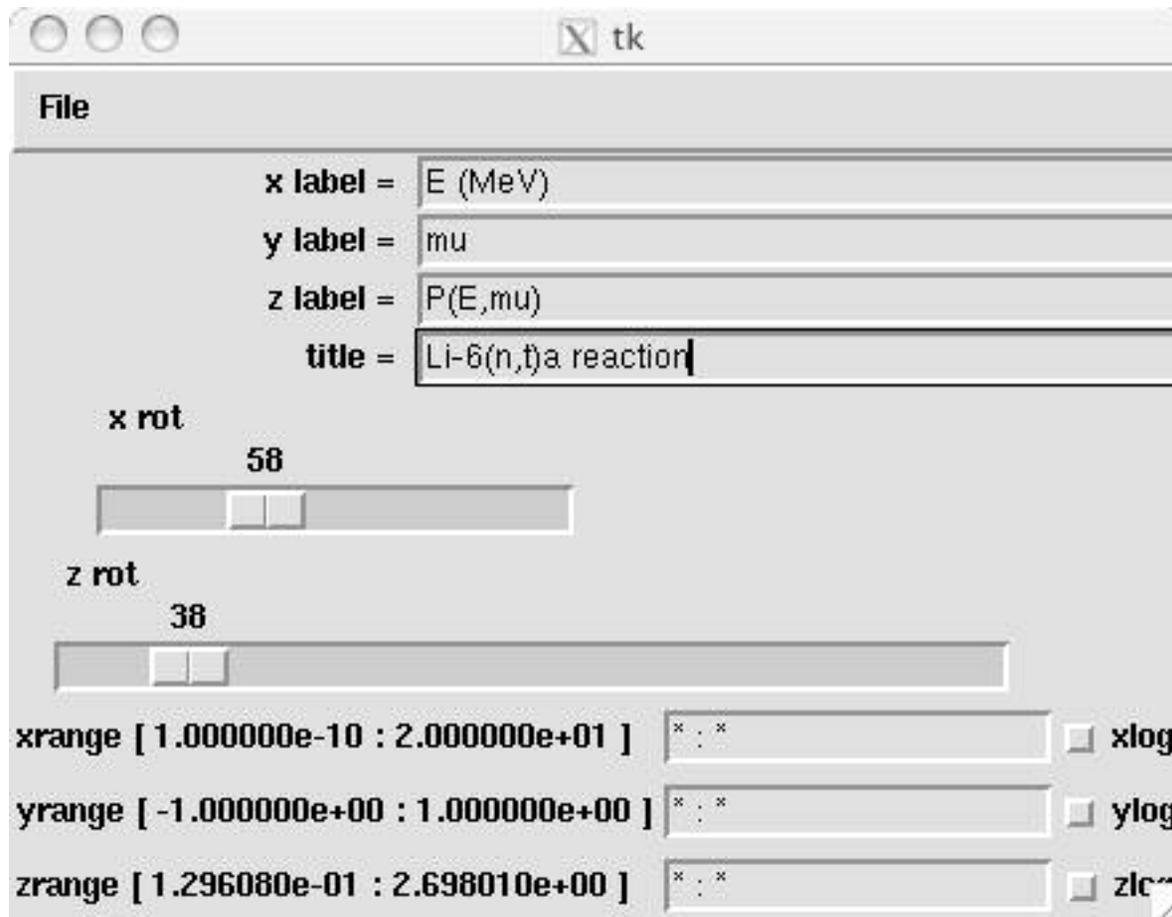
3d plot example



Gnuplot



3d plot dialog window



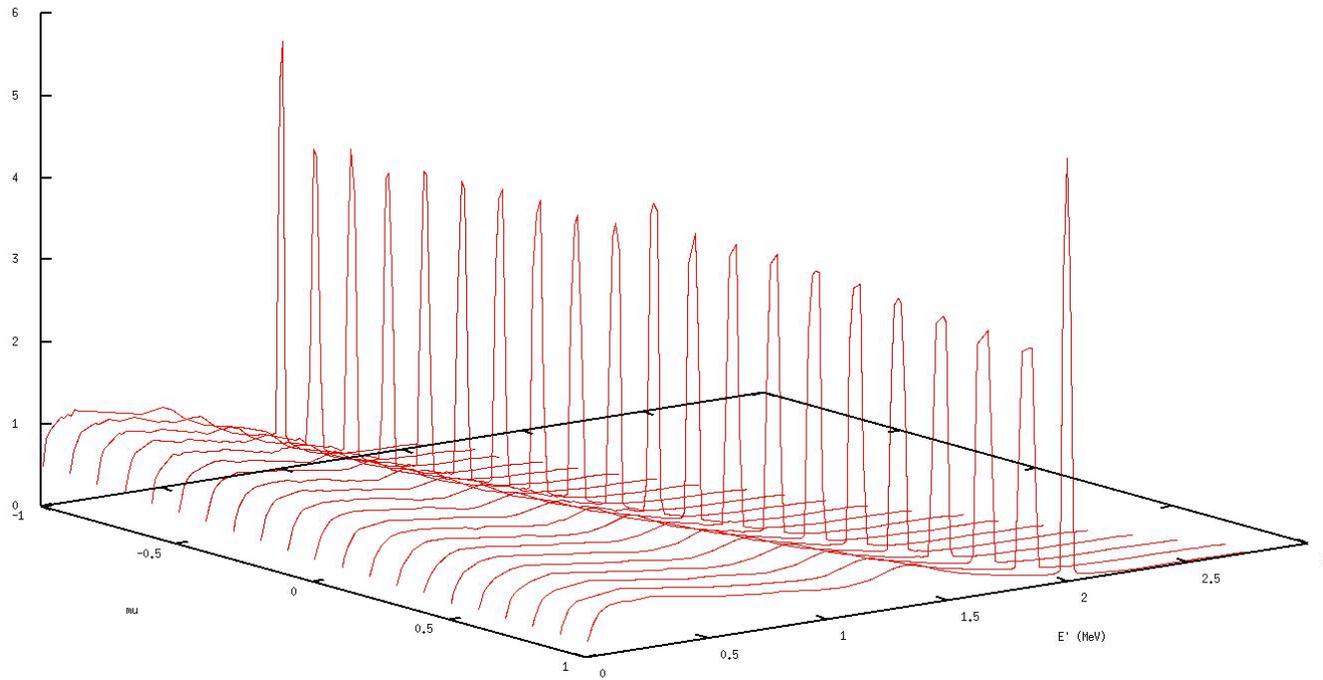
4d plot example



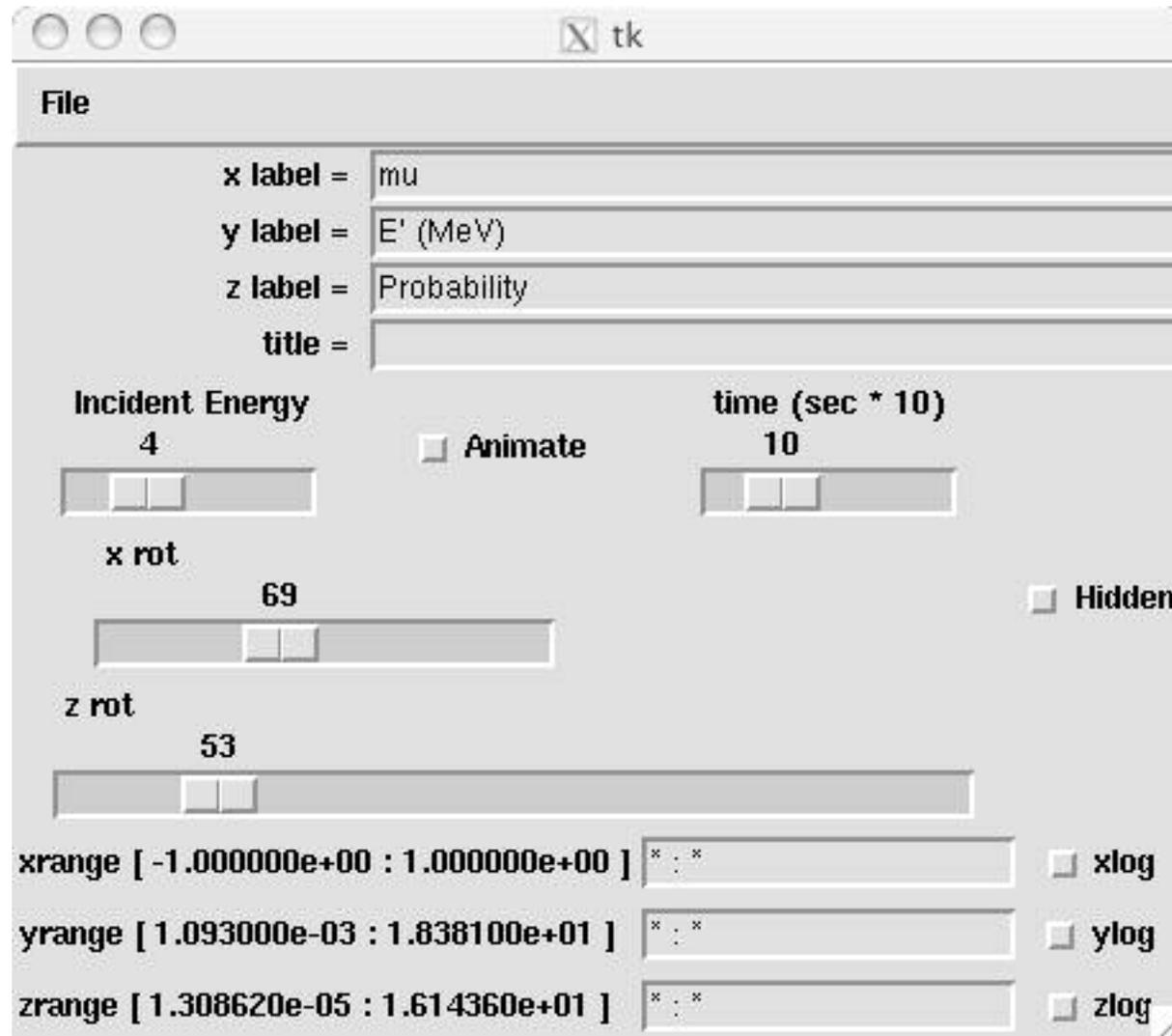
Gnuplot

E (MeV) = 4.500000e+00

Probability



4d plot dialog window



Future Improvements to FUDGE

- Use Numerical Python arrays to store data instead of the Python lists
 - Numerical Python arrays require less memory and are faster to search.
- Update FUDGE to handle uncertainty values.
 - ENDL format currently does not allow for uncertainty values
 - We are redesigning the ENDL format which will include support for uncertainty values.
 - End users want uncertainty values for guidance.
- Add more 3d and 4d math functionality.
- Add a Graphical User Interface (GUI).