LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-233400

Kiwi: An Evaluated Library of Uncertainties in Nuclear Data and Package for Nuclear Sensitivity Studies

J. Pruet

August 7, 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Version June 24, 2007

Kiwi: An evaluated library of uncertainties in nuclear data and package for nuclear sensitivity studies

J. Pruet

N Division, Lawrence Livermore National Laboratory, P. O.Box 808, Livermore, CA 94550

ABSTRACT

This report describes Kiwi, a program developed at Livermore to enable mature studies of the relation between imperfectly known nuclear physics and uncertainties in simulations of complicated systems. Kiwi includes a library of evaluated nuclear data uncertainties, tools for modifying data according to these uncertainties, and a simple interface for generating processed data used by transport codes. As well, Kiwi provides access to calculations of k eigenvalues for critical assemblies. This allows the user to check implications of data modifications against integral experiments for multiplying systems. Kiwi is written in python. The uncertainty library has the same format and directory structure as the native ENDL used at Livermore. Calculations for critical assemblies rely on deterministic and Monte Carlo codes developed by B division.

1. Introduction

At the national laboratories there has been over the last few years a growing interest in sensitivity analysis and uncertainty quantification. This relates in part to the adoption of QMU (quantification of margin uncertainties) as a tool for assessing stockpile reliability. Sensitivity studies are also emerging as a potentially powerful tool for guiding research investments aimed at improving predictive capability. The idea is that physics uncertainties found to have the biggest impact on calculations of performance can be targeted for directed study.

The computational nuclear physics group at Livermore developed Kiwi to facilitate nuclear sensitivity studies. As an added benefit, the process of looking carefully at nuclear uncertainties has helped us to identify existing data evaluations that need improvement. Kiwi includes:

1. A library of evaluated nuclear data uncertainties.

2. Tools for varying data according to these uncertainties and processing the varied data for use in transport codes.

3. A simple interface for checking varied data against critical assembly measurements.

A brief description of these pieces is given below. The appendix provides examples of how Kiwi is used. As a service to other programs at the Lab, members of the computational nuclear physics group will provide needed data files or scripts.

2. The Uncertainty Library

Table 1 lists reactions for which there are currently evaluated uncertainties in Kiwi. In addition to cross sections there estimates of uncertainties in the spectrum of prompt neutrons emitted following fission of ^{235}U and ^{239}Pu. We have an ongoing effort to add other important uncertainty evaluations.

For the most part, uncertainties were evaluated by considering the current ENDL evaluation and available experimental data. David Brown has developed a spline fitting code that provides a covariance matrix as well as point-wise uncertainty estimates. This was used in several evaluations. A description of this approach can be found in our evaluations for uncertainties in arsenic cross sections (Pruet, McNabb & Ormand 2005). A quite separate approach based on something akin to statistical bootstrapping was developed by Walid Younes. This approach is described in a technical report (Younes 2007) and was also used for several evaluations. In a few cases uncertainties were based on ENDF standards.

Uncertainties are stored in directory and file structures that closely mirror ENDL. The only difference is that whereas an ENDL file gives a reaction quantity (a cross section or spectrum), an uncertainty file gives the percent uncertainty in the reaction quantity. For example, a C=12, I=0 file in ENDL gives a list of [energy,cross-section] pairs for an n,2n reaction. The analogous C=12, I=0 file in the Kiwi database gives a list of [energy, percent uncertainty] pairs that describe the uncertainty in this same cross section. In general the user does not have to know many details of how data are stored.

3. Tools for Modifying Data

Kiwi provides different options for modifying cross sections and spectra. The user specifies which option to use through a parameter called "modType". For a given modification type the variation applied to data is determined by a second parameter called "modParam". Below we describe the different options for varying data and discuss how modParam is set for each option.

- modType = s1; modParam = f (a float)

  This simply scales a cross section by f

  $$\sigma(E) \to \sigma(E) \times f. \tag{1}$$

- modType = s2; modParam = f (a float)

  This adds of a multiple of the uncertainty in a cross section to the cross section

  $$\sigma(E) \to \sigma(E) + u(E) \times f, \tag{2}$$

  where $u(E)$ is the uncertainty in the cross section described by Kiwi's uncertainty library. As an example, if $f = 1$, then the whole cross section is increased by one standard deviation.

- modType = pieceMod; modParam = $[[e_1, f_1], [e_2, f_2], [e_3, f_3], ...]$

  This is similar to the s2 modification, except here modParam describes a variation that depends on energy. This variation is defined as a piecewise linearly-interpolable function that has the value f_j at $E = e_j$. The cross section variation is given by

  $$\sigma(E) \to \sigma(E) \times f(E), \tag{3}$$

  which is almost of the same form as the variation described by eq. 2.

- modType = covLength; modParam = [energyLength,energyList,variationList]

  This allows data variations that assume correlated uncertainties. The covariance matrix under this option takes the form

  $$C_{ij} = C(E_i, E_j) = \langle u(E_i)u(E_j)\rangle = u(E_i)\times u(E_j)\times \exp(-|E_1 - E_2|/\text{energyLength}). \tag{4}$$

  Here the different E_i's are defined in energyList and energyLength is an effective correlation length. Note that the variation described by modType=s2 is equivalent to assuming an infinite correlation length.

A new cross section section is calculated by first computing the Cholesky decomposition of the covariance matrix,

$$C = LL^T \tag{5}$$

and then setting

$$\sigma(E) \rightarrow \sigma(E) + L \times f \tag{6}$$

where f_i is the ith element of variationList. When f_i is drawn from a distribution with unit variance and zero mean the sampled cross section has a covariance matrix given by C above.

- modType = covLength4d; modParam = [energyLength,energyList,variationList]

  This is similar to the covLength option discussed above. The difference is that here the data to be varied is a function of two input parameters rather than one. An example of this is the neutron spectrum emitted following fission. This spectrum depends on the energy of the incident neutron and is given as a function of the outgoing neutron energy. By contrast, a cross section only depends on the incident neutron energy.

  At present it is assumed that variations in the spectrum are the same for all incident energies. Put another way, though the correlation in outgoing neutron energy is accounted for, we assume a perfect correlation in ingoing neutron energy. Different uncertainties can be specified for different ingoing neutron energies (though the current spectrum uncertainty evaluations do not depend on incident neutron energy).

4. Critical Assembly Calculations

Experimental studies of critical assemblies provide powerful integral constraints on some nuclear data. Kiwi has a simple interface for calculating k eigenvalues. Calculations can be done either with the deterministic code Amtran (Amtran Development Team 2005) or the Monte Carlo code Mercury (Mercury Development Team 2007). The user does not have to know anything about critical assemblies or how these codes work.

Table 2 lists the assemblies that can be calculated deterministically through Kiwi. At present only the bare uranium Godiva and bare plutonium Jezebel assemblies can be calculated through Kiwi using Monte Carlo. The appendix gives an example k eigenvalue calculation.

5. Appendix: A Kiwi example

Kiwi is a fairly simple program. There are only a handful of commands the user will generally need. In the following we give an example of how Kiwi is used to modify data, generate a processed ndf file for use in a deterministic calculation, and check this file against critical assemblies. Kiwi is written in python and relies on the data modification package FUDGE developed by Bret Beck.

```
#A simple Kiwi example
import sys
sys.path.append(‘‘/usr/gapps/data/nuclear/evaluated/xProj/Utils’’) #path to kiwi
from xProj import *

x = xProj() #x is now a ‘‘project’’ that will be used to vary data

endlString = ‘‘I=0,C=10’’  #elastic scattering cross section
ZA = 94239                 #for Pu-239
modType = ‘‘s1’’           #simple scale factor change to the cross section
modParam = 1.5             #by a factor of 1.5
x.modCS(ZA,modType,modParam,endlString) #modify the data

endlString = ‘‘I=0,C=15’’  #fission cross section
ZA = 94239                 #for Pu-239
modType = ‘‘s1’’           #add a multiple of the uncertainty to the cross section
modParam = 1.0             #1 standard deviation increase
x.modCS(ZA,modType,modParam,endlString)

endlString = ‘‘I=0,C=15’’  #fission cross section
ZA = 92235                 #for U-235
modType = ‘‘covLength’’    #assume correlated uncertainties
covarianceLength = 1.0     #1 MeV correlation length
eList = [1.0,2.0,5.0,15.0] #nodes for the cross section change
varlist = [1.0,1.0,-1.0,0.0] #change at each node (see text)
modParam=[covarianceLength,eList,varList]
x.modCS(ZA,modType,modParam,endlString)

x.recordChanges(‘‘~/modifiedData’’) #this records the changed data to a file
x.ProcNDF(old=’’~/oldNDFFile’’,’’~/ndf1’’) #makes a new ndf file based on oldNDFFile
```

```
#now we want to check a critical assembly using the new data
#for Godiva and Jezebel the interface is simple
kJezebel = x.checkStaticCrit(‘‘jezebel’’,’’ndf’’)
kGodiva = x.checkStaticCrit(‘‘godiva’’,’’ndf’’)

#for other assemblies the interface is a little clunkier

deckname = ‘‘pumetfast002.inp’’ #deckname from Table 2
a = Assembly_Amtran.Amtran_Assembly(ndfFile = ‘‘~/ndf1’’,deckName = deckname)
a.setIdgroup(4) #this idgroup must be consistent with the created ndf file
a.runProblem()
k3 = a.keff
a.clean() #remove files from amtran run
```

We are now working to make a uniform syntax for all the different critical assembly calculations.

REFERENCES

The AMTRAN Users Guide (2005) can be found in /usr/gapps/amtran/doc/amtran.htm on the LC computing machines.

The Mercury Users Guide (2007) can be found in /usr/gapps/Mercury/doc/ on the LC computing machines, UCRL-TM-204296.

Pruet, J., McNabb, D.P. & Ormand, E. (2005), “Cross Section Evaluations for Arsenic Isotopes”, UCRL-TR-210452

Younes, W. (2007), “A Systematic Procedure for Assigning Uncertainties to Data Evaluations”, UCRL-TR-228283

This preprint was prepared with the AAS LaTeX macros v5.0.

Table 1. Reactions in the Kiwi Uncertainty Library

target isotope	reactions
^{2}H	(n,el); (n,2n)
^{6}Li	(n,tot); (n,t); (n,nd)
^{9}Be	(n,el)
^{74}As	(n,2n)
^{75}As	(n,2n)
^{235}U	(n,tot); (n,γ); (n,2n); (n,f)[a]
^{238}U	(n,tot); (n,γ); (n,2n); (n,f)[a]
^{239}Pu	(n,f)[a]

[a]For fission the library includes descriptions of uncertainties in both the cross section and the spectrum of prompt post-fission neutrons.

Table 2. Critical Assemblies Calculated Deterministically in Kiwi

assembly name	fissile material	reflector	input deck name
Jezebel	Pu (92% 239)	none	`jezebel.inp`
Godiva	U (94% 235)	none	`godiva.inp`
U233-MET-FAST-001	U (98 % 233)	none	`u233metfast001.inp`
U233-MET-FAST-002	U (98 % 233)	HEU (93% 235)	`u233metfast002.inp`
U233-MET-FAST-004	U (98 % 233)	tungsten	`u233metfast004.inp`
HEU-MET-FAST-003l	U (94% 235)	nickel	`heumetfast003.inp`
IEU-MET-FAST-005	U (36% 235)	steel	`ieumetfast005.inp`
HEU-MET-FAST-019	U (87% 235)	graphite	`heumetfast019.inp`
HEU-MET-FAST-020	U (87% 235)	poly	`heumetfast020.inp`
HEU-MET-FAST-028	U (93% 235)	uranium (99% 238)	`heumetfast028.inp`
PU-MET-FAST-002	Pu (74% 239, 14% 240)	none	`pumetfast002.inp`
PU-MET-FAST-008a	Pu (91% 239)	thorium	`pumetfast008a.inp`
PU-MET-FAST-011	Pu (95% 239)	water	`pumetfast011.inp`
PU-MET-FAST-018	Pu (92 % 239)	Be	`pumetfast018.inp`