Functional data containers

Presented to WPEC Subgroup 38

May 2016

Bret Beck

Lawrence Livermore National Laboratory

LLNL-PRES-???????

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Outline

- General comments
- 1d functional containers
- (n)d functional containers
- Math vs. physics functional container

Sorry, or should I be, that we did not update and distribute the specification on data containers, but I will note changes in this talk. In general, the main difference is in the naming (added dimension to names and removed ‘dimension’ attribute). This was driven by feedback from SG38 and I agree that it is an improvement.

Caveat

- Some of the FUDGE example are not fully functioning

General comments

- Want a set of functional data containers that store 1d-, 2d-, 3d-, …, (n)d-functions
- The more general the better
  - For example, do not limit to what is currently needed
  - We should limit the use and not the containers
    - I will give an example later
- Should be able to add containers as needed
  - Should be consistent with current containers
- Want to be able to represent a real function

What is a (n)d function?

- 1d function — $f(x1)$
  - Examples: $\sigma(E)$, $Q(E)$
- 2d function — $f(x2, x1)$
  - Examples: $P(\mu|E)$, $P(E'|E)$
- 3d function — $f(x3, x2, x1)$
  - Example: $P(E',\mu|E)$
- (n)d function — $f(x\{n\}, x\{n-1\}, \ldots x3, x2, x1)$
  - Example for 4d: $P(E',\mu,\varphi|E)$

We currently need different containers for each dimension

Real vs. pseudo functions

- Mathematical definition of a function
  - Must be single valued
- ENDF allows for what I call pseudo functions as some functions in ENDF are multi-valued
  - I have seen up to 10-ish y values for a single x value
- While we need to support what is in ENDF (but only for two y values per x value) I believe some functional containers should only support real functions (i.e., should only be single value). And, given the following MCNP warning, I think I am not the only one:

warning. ***2 coincident energy grid points in 92235.01c***

1d functional containers

Supporting ENDF TAB1

- Definition of TAB1 in ENDF manual

0.6.3.7 TAB1 Records

These records are used for one-dimensional tabulated functions such as $y(x)$. The data needed to specify a one-dimensional tabulated function are the interpolation tables NBT(N) and INT(N) for each of the NR ranges, and the NP tabulated pairs of $x(n)$ and $y(n)$. The shorthand representation is:

[MAT,MF,MT/ C1, C2, L1, L2, NR, NP/x_{int}/$y(x)$]TAB1

NR, number of ranges. Allows for multiple ranges with different interpolations.

Figure in ENDF manual

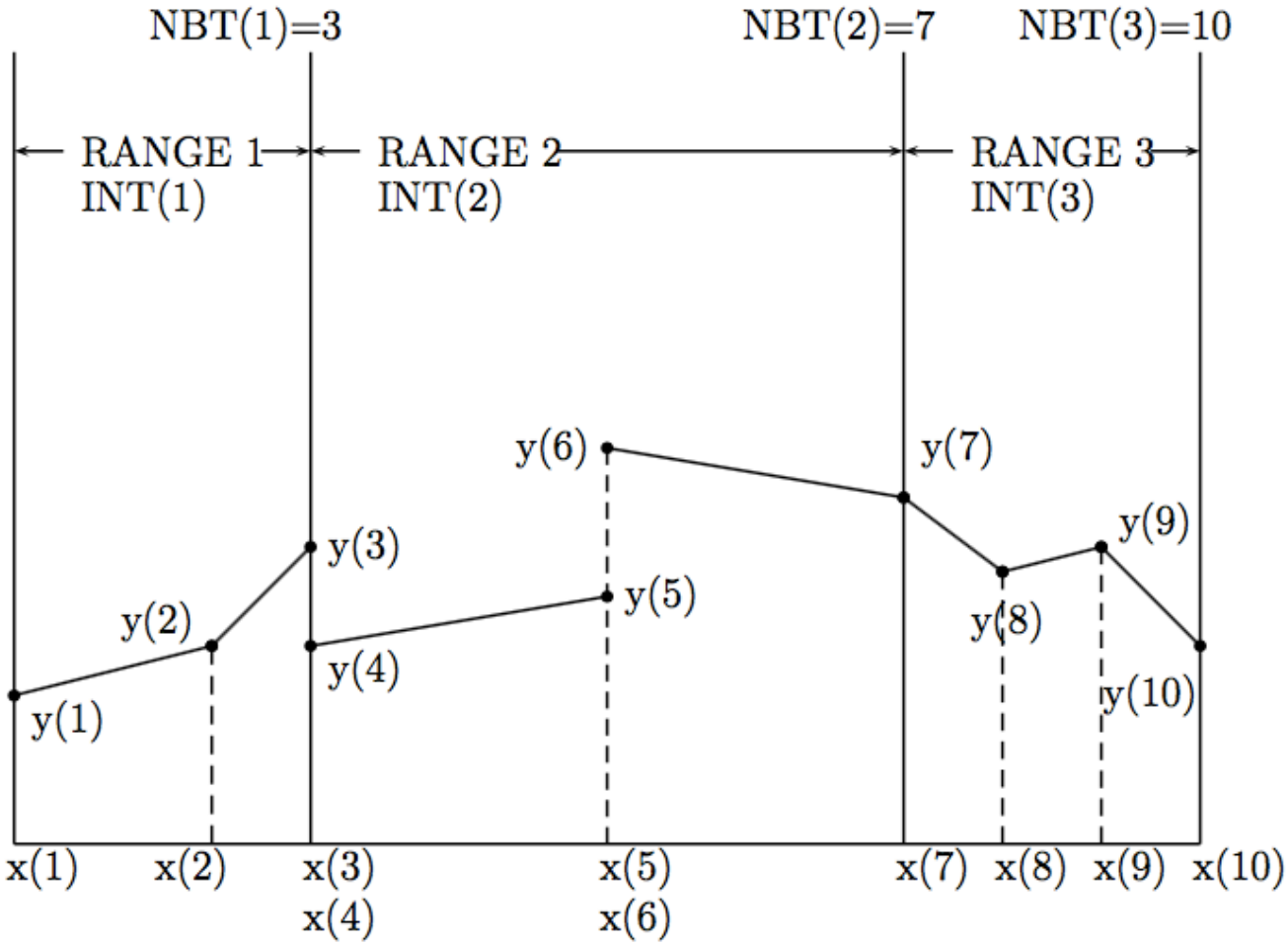

Figure 1: Interpolation of a tabulated one-dimensional function for a case with NP=10, NR=3.

ENDF TAB1

- Simple example with schematic XML container

```
 3.006000+3 5.963400+0          0          0          0          0 325 3 81
-1.473700+6-1.550000+7          0         32          1          5 325 3 81
          5          2          0          0          0          0 325 3 81
 1.809900+7 0.000000+0 1.850000+7 3.446747-3 1.900000+7 4.865911-3 325 3 81
 1.950000+7 5.764370-3 2.000000+7 6.387954-3                       325 3 81
```

```
<XYs1d interpolation="lin-lin">
   1.809900+7 0.000000+0 1.850000+7 3.446747-3 1.900000+7 4.865911-3
   1.950000+7 5.764370-3 2.000000+7 6.387954-3</XYs1d>
```

- Example with 3 ranges (I will call regions later)

```
 3.006000+3 5.963400+0          0          0          0          0 325 3 81
-1.473700+6-1.550000+7          0         32          1          5 325 3 81
          2          2          4          5          5          2 325 3 81
 1.809900+7 0.000000+0 1.850000+7 3.446747-3 1.900000+7 4.865911-3 325 3 81
 1.950000+7 5.764370-3 2.000000+7 6.387954-3                       325 3 81
```

```
<XYs1d interpolation="2 2 4 5 5 2">
   1.809900+7 0.000000+0 1.850000+7 3.446747-3 1.900000+7 4.865911-3
   1.950000+7 5.764370-3 2.000000+7 6.387954-3</XYs1d>
```

Could store like ENDF, not!

Example with 3 ranges cont.

- ENDF

```
 3.006000+3 5.963400+0          0          0          0          0 325 3 81
-1.473700+6-1.550000+7          0         32          1          5 325 3 81
          2          2          4          5          5          2 325 3 81
 1.809900+7 0.000000+0 1.850000+7 3.446747-3 1.900000+7 4.865911-3 325 3 81
 1.950000+7 5.764370-3 2.000000+7 6.387954-3                       325 3 81
```

```
<regions1d>
   <XYs1d interpolation="lin-lin"> 1.809900+7 0.000000+0 1.850000+7 3.446747-3 </XYs1d>
   <XYs1d interpolation="log-log"> 1.850000+7 3.446747-3 1.900000+7 4.865911-3
         1.950000+7 5.764370-3 </XYs1d>
   <XYs1d interpolation="lin-lin"> 1.950000+7 5.764370-3 2.000000+7 6.387954-3 </XYs1d></regions1d>
```

- <XYs1d> always are a real function.
- The <regions1d> is a pseudo function
- In FUDGE this can be converted to a single <XYs1d> by linearizing the middle region and blurring the boundaries

List of current 1d functional containers

- <XYs1d>
  - Contains a pointwise, single valued representation of a 1d function. This is, list of $[x1_i, f_i]$ pairs with $x1_i < x1_{i+1}$.
- <constant1d>
  - A 1d function that is constant between $X1_{Min}$ and $X1_{Max}$
- <polynomial1d>
  - Contains coefficients representing $f(x1) = \sum_{l=lMin}^{lMax} C_l \, (x1 - x1_{offset})^l$
- <Legendre1d>
  - Contains a list of Legendre coefficients
- <gridded1d>
  - Efficient for storing multi-group data
- <regions1d>
  - Contains 2 or more of the above 1d functional containers

The power of the <regions1d> container

- Prior example show how it allows for discontinuities (i.e., pseudo functions for compatibility with ENDF) and change in interpolation

- Also allows for mixing and matching of functions

```
<regions1d>
   <polynomial1d domainMin="0" domainMax="0.1"> 1 .2 0.05 </polynomial1d>
   <XYs1d interpolation="log-log"> 0.1 1.0205 0.2 2 . . . 10 3</XYs1d>
   polynomial1d domainMin="10" domainMax="2e7" domainOffset="10"> 3 .3 </polynomial1d></regions1d>
```

Higher dimensional functional containers

Higher dimensional functional containers

- ENDF TAB2

0.6.3.8 TAB2 Records

The next record type is the TAB2 record, which is used to control the tabulation of a two-dimensional function $y(x, z)$. It specifies how many values of z are to be given and how to interpolate between the successive values of z. Tabulated values of $y_l(x)$ at each value of z_l are given in TAB1 or LIST records following the TAB2 record, with the appropriate value of z in the field designated as C2. The shorthand notation for TAB2 is

[MAT,MF,MT/ C1, C2, L1, L2, NR, NZ/ Z_{int}]TAB2,

- Note, function defined as y(x,z)
- Allows for ranges (NR) alonf z-axis
  - That is, a change in interpolation
- As with TAB1, can be multi-valued.
- Stores a list of y(x) which in ENDF can be TAB1's or Legendre coefficients (actually LIST).

Defined 2d functional containers

- <XYs2d>
  - Contains a list of $[x2_i, f_i(x1)]$ pairs with $x2_i < x2_{i+1}$.
  - $f_i(x1)$ can be any mix/match of 1d functional containers
- <gridded2d>
  - May be useful for storing isotropic multi-grouped data
- <regions2d>
  - Contains 2 or more of the above 2d functional containers

Note, <XYs2d> is only guaranteed to a real function along X2 as some of the $f_i(x1)$'s can be <regions1d> containers. <XYs2d> with <regions2d> act like TAB2 in 2d.

Examples for 2d containers for P(μ|E)

- LLT=1 can be stored as a list of <Legendre1d> in a <XYs2d> container as:

```
<XYs2d>
    <Legendre1d value="1e-5"> . . . </Legendre1d>
    <Legendre1d value="2e-5"> . . . </Legendre1d>
    <Legendre1d value="3e-5"> . . . </Legendre1d>
    . . .
    <Legendre1d value="2e7"> . . . </Legendre1d></XYs2d>
```

- LLT=2 can be stored as a list of <XYs1d> in a <XYs2d> container as:

```
<XYs2d>
    <XYs1d value="1e-5"> . . . </XYs1d>
    <XYs1d value="2e-5"> . . . </XYs1d>
    <XYs1d value="3e-5"> . . . </XYs1d>
    . . .
    <XYs1d value="2e7"> . . . </XYs1d></XYs2d>
```

Examples cont.

- LLT=3 can be stored as a list of <Legendre1d> in a <XYs2d> container followed by a list of <XYs1d> in a <XYs2d> container where the <XYs2d>'s are stored in a <regions2d> container as:

```
<regions2d>
    <XYs2d>
        <Legendre1d value="1e-5"> . . . </Legendre1d>
        <Legendre1d value="2e-5"> . . . </Legendre1d>
        <Legendre1d value="3e-5"> . . . </Legendre1d>
        . . .
        <Legendre1d value="2e1"> . . . </Legendre1d></XYs2d>
    <XYs2d>
        <XYs1d value="2e1"> . . . </XYs1d>
        <XYs1d value="2e2"> . . . </XYs1d>
        <XYs1d value="2e3"> . . . </XYs1d>
        . . .
        <XYs1d value="2e7"> . . . </XYs1d></XYs2d></regions2d>
```

Change in functional without region2d

- LLT=3 can be stored as a list of <Legendre1d> in a <XYs2d> container followed by a list of <XYs1d> in a <XYs2d> container where the <XYs2d>'s are stored in a <regions2d> container as:

```
<XYs2d>
        <Legendre1d value="1e-5"> . . . </Legendre1d>
        <Legendre1d value="2e-5"> . . . </Legendre1d>
        <Legendre1d value="3e-5"> . . . </Legendre1d>
        . . .
        <Legendre1d value="1e1"> . . . </Legendre1d>
        <XYs1d value="2e1"> . . . </XYs1d>
        <XYs1d value="2e2"> . . . </XYs1d>
        <XYs1d value="2e3"> . . . </XYs1d>
        . . .
        <XYs1d value="2e7"> . . . </XYs1d></XYs2d>
```

I believe that the <XYs2d> container should allow this as we do not want to restrict what people in the future may want, but we can limit its use currently as a community.

Defined 3d functional containers

- <XYs3d>
  - Contains a list of [$x3_i$, $f_i(x2,x1)$] pairs with $x3_i < x3_{i+1}$.
  - $f_i(x2,x1)$ can be any mix/match of 2d functional containers
- <gridded3d>
  - Stores 3d multi-grouped data compactly
- <regions3d>
  - Contains 2 or more of the above 3d functional containers

Note, <XYs3d> is only guaranteed to a real function along X3 as some of the $f_i(x2,x1)$'s can be <regions2d> containers or contain <regions1d> containers. <XYs3d> with <regions3d> act like TAB2 in 3d.

Defined {n}d functional containers

- <XYs{n}d>
  - Contains a list of [x{n}$_i$, f_i(x{n-1}, . . ., x2,x1)] pairs with x{n}$_i$ < x{n}$_{i+1}$.
  - f_i(x{n-1}, . . ., x2,x1) can be any mix/match of {n-1}d functional containers
- <regions{n}d>
  - Contains 2 or more of the above {n}d functional containers

Note, <XYs{n}d> is only guaranteed to a real function along X{n}.

Real functional containers

- If a functional container is not a regions contain and it does not contain a regions container at any dimension, then it is guaranteed to be a real function.

- If a container is a regions container or contains regions container at any dimension, it may or may not be a real function.

- This is why I have stated that regions contains must contain 2 or more containers of there dimension. If it only contains one, remove the regions container as it is not needed.

Math vs. physics functional container

What is math and what is physics

- Is P(μ) as math or physics function?
- Is P(E,μ) as math or physics function?
- Is P(E',μ) as math or physics function?
- Is P(E',$\mu|E$) as math or physics function?

Physics containers in FUDGE

- Is P(μ) as math or physics function?
- Is P(μ|E) as math or physics function?
- Is P(E',μ) as math or physics function?
- Is P(E',μ|E) as math or physics function?
- In FUDGE they are all physics functions.
  - And, P(μ) are contained in P(μ)|E) and P(E',μ) containers. Also, P(E',μ) are contained in P(E',μ|E) containers

Math vs. physics

- A math function does not know what it represent. That is, in 1d it is just a the function y(x).

- In physics, the function represents a physical quantity which generally has labels and units
  - Examples, σ(E) in barn, Q(E) in eV, P(E',μ|E) in 1/eV with E in eV and E' in eV.

- FUDGE does not distinguish between math and physics as all functional containers are physics containers. That is, all have an <axes> member.

Why does FUDGE only have Physics containers?

- So that a P(μ) in a P(μ|E) knows it is a physics function (i.e., knows what it is) and can stand on its own.

```
<XYs2d>
    <Legendre1d value="1e-5"> . . . </Legendre1d>
    <Legendre1d value="2e-5"> . . . </Legendre1d>
    <Legendre1d value="3e-5"> . . . </Legendre1d>
    . . .
    <Legendre1d value="2e7"> . . . </Legendre1d></XYs2d>
```

xData	angular.py	energy.py
XYs1d	XYs1d(xData.XYs1d)	XYs1d(xData.XYs1d)
Legendre1d	Legendre1d(xData.Legendre1d)	
region1d	regions1d(xData.regions1d)	regions1d(xData.regions1d)
XYs2d	XYs2d(xData.XYs2d)	XYs2d(xData.XYs2d)
regions2d	regions2d(xData.regions2d)	regions2d(xData.regions2d)

Example 1 in pseudo FUDGE

```
PofMuGivenE = angular.XYs2d( . . .  )
PofMu = angular.XYs1d( . . . )
PofMuGivenE.append( PofMu )

PofEpGivenE = energy.XYs2d( . . .  )
PofEp = energy.XYs1d( . . . )
PofEpGivenE.append( PofEp )

PofEpGivenE .append(PofMu )          # will cause a raise
```

Example 2 in pseudo FUDGE

```
PofEpGivenE = energy.XYs2d.readXML( ) # E and E' in MeV

PofEp = energy.XYs1d.readXML( )       # E and E' in eV

PofEpGivenE.add( PofEp )      # FUDGE will convert PofEp
                              # to MeV
```

When we use or compare part of a function, for example, the P(E'), I think we want it to be a physics function.

Example 3 in pseudo FUDGE

```
PofEpGivenE = energy.XYs2d.readXML( ) # E and E' in MeV
PofEpGivenE.plot( )          # label with units automatic
PofEpGivenE[2].plot( )       # label with units automatic

PofEp = energy.XYs1d.readXML( )
PofEp.plot( )                # label with units automatic
```

Example 4 in pseudo FUDGE

```
xSec1 = crossSection.readXML( . . . ) # E and E' in MeV

xSec2 = crossSection.readXML( . . . ) # E and E' in eV

total = xSec1 + xSec2                 # Automatic unit conversion
```

If FUDGE had a math and physics class, then we would need to have an _add__ method for both, and every other method in math that we also want in physics.

Math container with a physics container

- P(E',μ|E) # contains axes
  - M(E',μ|E)
    - P(E',μ) # contains axes?
      - M(E',μ)
        - P(μ) # contains axes?
          - M(μ)

- P(E',μ|E) # contains axes
  - P(E',μ) # contains axes or inherits from parent
    - P(μ) # contains axes or inherits from parent

Pros and cons from and infrastructure view

- Pros
  - Have to implement axes anyway
  - Do not have to implement many methods twice and keep them in sync
  - At every level, a functional knows what it is
- Cons
  - I cannot think of one as axes class has to be implemented and a member of something