Webnucleo Technical Report: User-Supplied Rate Functions to libnucnet

Bradley S. Meyer

June 13, 2011

This technical report describes how to supply rate functions to libnucnet.

1 User-Supplied Rate Functions

The standard functions in libnucnet for calculating reaction rates are single rates (numbers independent of the temperature, density, or any other external parameter), rate tables (tables giving the rate vs. temperature, which are interpolated), or non-smoker fit functions. These standard rate functions are described in the libnucnet input technical report.

In some cases users may wish to provide their own rate functions that would be applied during reaction rate calculations. As of version 0.5, this is possible. To do so, a user writes a Libnucnet__Reaction__userRateFunction with the prototype

```
double
my_rate_function(
  Libnucnet__Reaction *p_reaction,
  double d_t9,
  void *p_user_data
);
```

In this function, **d_t9** is the temperature in 10^9 K and **p_user_data** is a pointer to a user-supplied data structure. The user's routine then returns the rate for the reaction. The user must supply a rate function for each different rate parameterization considered.

2 Setting the Rate Function for a Reaction

Once user rate functions are defined, they should be registered with the reaction collection. To do so, the user calls Libnucnet__Reac__registerRateFunction(). For example, with the **my_rate_function** above, one would call

```
Libnucnet__Reac__registerRateFunction(
  p_my_reactions,
  "my rate function",
  (Libnucnet__Reaction__userRateFunction) my_rate_function
);
```

to register the function with the reaction collection Libnucnet__Reac *p_my_reactions. In this example, the string "my rate function" is a key that allows libnucnet to retrieve and apply the appropriate function to a reaction.

A reaction needs to have an associated function key. If reaction data are read in from XML, this is done automatically as the XML is parsed via the **key** attribute to the **user_rate** tag. If a reaction is added separately, however, the user must set the key directly. The user does this by calling the API routine Libnucnet__Reaction__setUserRateFunctionKey with the prototype

```
void
Libnucnet__Reaction__setUserRateFunctionKey(
  Libnucnet__Reaction *p_reaction,
  const char *s_function_key
);
```

In this function, s_function_key is a function key to a registered user rate function.

3 Rate Data

If a reaction has a user-supplied rate function, the user can set the data for it by the Libnucnet__Reaction__updateUserRateFunctionProperty() API routine. Such data are typically reaction-rate fit parameters. They are properties, which are strings identified by a name and up to two optional tags. The user may do this directly or may set the data in the input xml file, as described in the Webnucleo technical report on input xml to libnucnet.

Once the data for a reaction are set, they may be retrieved via the Libnucnet__Reaction__getUserRateFunctionProperty() routine. Here the user supplies the reaction pointer and the strings for the name of the property and the tags, if present. The property is returned as a string. The properties may also be accessed by the API routine Libnucnet__Reaction__iterateUserRateFunctionProperty(). Here the user supplies a function with the prototype

```
void
my_iterate_function(
  const char *s_name,
  const char *s_tag1,
  const char *s_tag2,
  const char *s_value,
  void *p_data
);
```

Once this function is defined, the user then iterates over the properties for the rate function for the reaction by calling, for example,

```
Libnucnet__Reaction__iterateUserRateFunctionProperties(
  p_reaction,
  s_name,
  s_tag1,
  s_tag2,
  (Libnucnet__Reaction__user_rate_property_iterate_function)
    my_iterate_function,
  p_data
);
```

The iteration is over all properties that match s_name, s_tag1, and s_tag2. If any of these is NULL, the comparison is a match; thus, if all three are NULL, the routine will iterate over all properties of the rate function for the reaction.

Finally, the user may remove a property for a user rate function by calling Libnucnet__Reaction__removeUserRateFunctionProperty().

4 Extra Data

A reaction rate is typically computed from fit data and from data characterizing the physical conditions present at any point in the calculation. The former are what we have called "rate data". These are data that apply to a particular reaction. The latter are extra, or "user", data that might correspond to the density at a particular point in the calculation. These user data are those supplied to the user's rate function, that is, the data pointed to by **p_user_data**, as described in §1.

The extra data are typically associated with time-dependent quantities, which, in turn, are associated with zones. As of version 0.9, then, to set these data, the user calls Libnucnet__Zone__updateDataForUserRateFunction(), which has the syntax:

```
void
Libnucnet__Zone__updateDataForUserRateFunction(
  Libnucnet__Zone *self,
  const char *s_function_key,
  void *p_data
);
```

For example, to pass the density (stored as **d_density**) to a user rate function registered with the key **"my rate function"** in the zone **p_zone**, one would call

```
Libnucnet__Zone__updateDataForUserRateFunction(
  p_zone,
  "my rate function",
  &d_density
);
```

Once this is done, any reaction rate computed from the function with key "my rate function" would be computed in p_zone from the rate data for the particular reaction and the current temperature T_9 and the density d_density set in the function above.

As of version 0.10, the user can assign a deallocator for the extra data associated with a user-defined rate function. The user writes a routine with the prototype

```
void
Libnucnet__Reaction__user_rate_function_data_deallocator(
  void * p_data
);
```

where **p_data** is a pointer to the user-defined data structure. This routine must free all memory associated with **p_data**, including the structure itself. Once this routine is written, the user then associates it with a rate function by calling Libnucnet__Reac__setUserRateFunctionDataDeallocator(), with prototype

```
int
Libnucnet__Reac__setUserRateFunctionDataDeallocator(
  Libnucnet__Reac * p_reac,
  const char * s_function_key,
  Libnucnet__Reaction__user_rate_function_data_deallocator
    pf_deallocator
);
```

This routine associates the deallocator function **pf_deallocator** to the user rate function defined by the key **s_function_key** for the reaction collection **p_reac**. The routine returns 1 (true) if the association succeeded and 0 (false) if not. Once the deallocator is associated with a rate function, a call to Libnucnet__Zone__updateDataForUserRateFunction() will employ the user's deallocation function to deallocate the memory for the previously allocated extra data before adding the new data.

The current data in a zone for a user rate function can be retrieved with the API function Libnucnet__Zone__getDataForUserRateFunction with the prototype

```
void *
Libnucnet__Zone__getDataForUserRateFunction(
  Libnucnet__Zone * p_zone,
  const char * s_function_key
);
```

This retrieves the current extra data for the rate function identified by the rate function key s_function_key. Examples in the libnucnet distribution demonstrate how to apply extra data to a user function.