Webnucleo Technical Report: User-Supplied Rate Functions to libnucnet

Bradley S. Meyer

January 10, 2010

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__rateFunction 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 a rate function has been defined, a user may set that function as the rate function for a particular reaction. To do so, the user calls the Libnucnet__Reac API routine Libnucnet__Reaction__setUserRateFunction(). For example, suppose the rate function **my_rate_function** has been written. The user then sets this as the rate function for reaction **p_reaction** by calling Libnucnet__Reaction__setUserRateFunction():

```
Libnucnet__Reaction__setUserRateFunction(
  p_reaction,
  "my rate function",
  (Libnucnet__Reaction__rateFunction) my_rate_function
);
```

The string **"my rate function"** is a key that allows the user to access the rate function for other purposes, such as updating the extra data.

If the user creates a Libnucnet__Reac, either directly or as part of a Libnucnet__Net or Libnucnet structure, it is possible to register the function with the Libnucnet__Reac structure. 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__rateFunction) my_rate_function
);
```

to register the function with the reaction collection Libnucnet__Reac *p_my_reactions. The user then calls

```
Libnucnet__Reac__setUserRateFunctions( p_my_reactions );
```

to set the function for each reaction according to the appropriate key, assuming the key has been set for each reaction. This is done with the Libnucnet__Reaction__setUserRateFunction(), as described above, or, more likely, via the **key** attribute to the **user_rate** tag in the reaction data xml parsed to create the reaction collection structure. It should be noted that the API function Libnucnet__Net__computeRates() calls Libnucnet__Reac__setUserRateFunctions() itself, as needed, so the user only needs to call this if he or she computes the rates for a reaction directly.

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. To set these data, the user calls Libnucnet__Reac__updateDataForUserRateFunction(), which has the syntax:

```
void
Libnucnet__Reac__updateDataForUserRateFunction(
  Libnucnet__Reac *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 reaction collection **p_my_reactions**, one would call

```
Libnucnet__Reac__updateDataForUserRateFunction(
  p_my_reactions,
  "my rate function",
  &d_density
);
```

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