ITM AMNS Interface

D.P. Coster

[1] Max-Planck-Institut für Plasmaphysik, EURATOM Association, D-85748 Garching bei München, Germany

September 10, 2008

Outline

Introduction

With this interface I aim at clearly separating the AMNS data and the use of the data. The only communication between the two should be by 9 calls, which break into the following:

- initialization (2)
- finalization (2)
- querying parameters (2)
- setting parameters (2)
- getting data (1)

For the first 4, there is one call that applies to the entire AMNS system, and one call that applies to a particular table. In all cases a "handle" is used to identify which instance is being addressed. An optional argument can be specified for each call to get the error status — if not provided, an error will cause the code to stop.

ITM_AMNS_SETUP

```
subroutine ITM_AMNS_SETUP(handle, version, error_status)
  optional version, error_status
  type(amns_handle_type), intent(out) :: handle
  type(amns_version_type), intent(in) :: version
  type(amns_error_type), intent(out) :: error_status
```

- Initializes the whole package

ITM_AMNS_SETUP_TABLE

```
subroutine ITM_AMNS_SETUP_TABLE( &
    handle,reaction_type,reactant, &
    handle_rx,error_status)
  optional error_status
  type(amns_handle_type), intent(in) :: handle
  type(amns_reaction_type), intent(in) :: reaction_type
  type(amns_reactants_type), intent(in) :: reactant
  type(amns_handle_rx_type), intent(out) :: handle_rx
  type(amns_error_type), intent(out) :: error_status
```

- Initializes the AMNS package for a particular reaction
- reaction specified by
  - `reaction_type`
  - `reactant`

ITM_AMNS_FINISH

```
subroutine ITM_AMNS_FINISH(handle, error_status)
  optional error_status
  type(amns_handle_type), intent(inout) :: handle
  type(amns_error_type), intent(out) :: error_status
```

- Terminates the use of the AMNS package
  - frees up allocated memory

ITM_AMNS_FINISH_TABLE

```
subroutine ITM_AMNS_FINISH_TABLE(handle_rx, error_status)
  optional error_status
  type(amns_handle_rx_type), intent(inout) :: handle_rx
  type(amns_error_type), intent(out) :: error_status
```

- Terminates the use of the table associated with a particular reaction
  - frees up allocated memory

ITM_AMNS_QUERY

```
subroutine ITM_AMNS_QUERY(handle,query,answer,error_status)
  optional error_status
  type(amns_handle_type), intent(in) :: handle
  type(amns_query_type), intent(in) :: query
  type(amns_answer_type), intent(out) :: answer
  type(amns_error_type), intent(out) :: error_status
```

- provides a mechanism for querying the AMNS package

  version

ITM_AMNS_QUERY_TABLE

```
subroutine ITM_AMNS_QUERY_TABLE( &
    handle_rx,query,answer, &
    error_status)
  optional error_status
  type(amns_handle_rx_type), intent(in) :: handle_rx
  type(amns_query_type), intent(in) :: query
  type(amns_answer_type), intent(out) :: answer
  type(amns_error_type), intent(out) :: error_status
```

- provides a mechanism for querying for information about a particular reaction

source	data source
no_of_reactants	number of involved reactants
index	reaction index (used at the moment for choosing spectroscopic line)

ITM_AMNS_SET

```
subroutine ITM_AMNS_SET(handle,set,error_status)
  optional error_status
  type(amns_handle_type), intent(in) :: handle
  type(amns_set_type), intent(in) :: set
  type(amns_error_type), intent(out) :: error_status
```

- used for setting global parameters
    - none implemented yet
    - could implement a global debugging flag

ITM_AMNS_SET_TABLE

```
subroutine ITM_AMNS_SET_TABLE(handle_rx,set,error_status)
  optional error_status
  type(amns_handle_rx_type), intent(in) :: handle_rx
  type(amns_set_type), intent(in) :: set
  type(amns_error_type), intent(out) :: error_status
```

- used for setting parameters associated with a particular table

nowarn	Don't complain about extrapolation

ITM_AMNS_RX

```
subroutine ITM_AMNS_RX_1(handle_rx,out,arg1,arg2,arg3,error_
  optional arg2,arg3,error_status
  type(amns_handle_rx_type), intent(inout) :: handle_rx
  real (kind=R8), intent(out) :: out(:)
  real (kind=R8), intent(in) :: arg1(:),arg2(:),arg3(:)
  type(amns_error_type), intent(out) :: error_status
```

- Used for getting the rate on a grid "out" of the same dimensions as "arg1", "arg2", ...

ITM_AMNS_RX, continued

- There is a generic interface `ITM_AMNS_RX` and particular routines `ITM_AMNS_RX_1`, `ITM_AMNS_RX_2` and `ITM_AMNS_RX_3`
  - handles 1d, 2d and 3d codes
  - (increasing this is easy!)
- "out" can depend on 1, 2 or 3 physics parameters
  - for 1, only "arg1" should be specified
  - for 2, "arg1" & "arg2" should be specified
  - for 3, "arg1", "arg3" & "arg3" should be specified
  - (increasing this to 4d is easy, beyond would require additional work)

Implementation of the AMNS package

- set of f95 types
- set of f95 modules
- uses the GRID interpolation package of Silvio Gori (IPP-Garching)

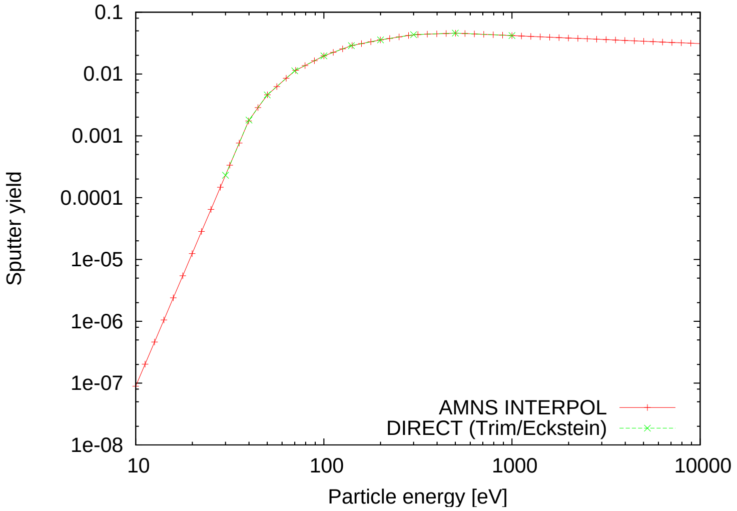
Implementation of the GRID interpolation package

- written by Silvio Gori
- set of f95 types
- set of f95 modules
- implements linear interpolation on 1d, 2d, 3d and 4d grids

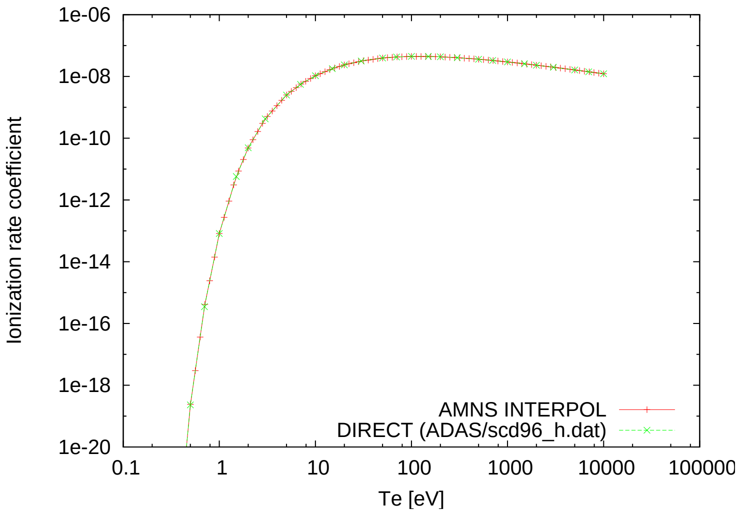
Using the package

```
  call ITM_AMNS_SETUP(amns)
  query%string='version'
  call ITM_AMNS_QUERY(amns,query,answer)
...
  call ITM_AMNS_SETUP_TABLE(amns, lr_rx, species_lr, amns_lr)
  query%string='source'
  call ITM_AMNS_QUERY_TABLE(amns_lr,query,answer)
...
  set%string='nowarn'
  call ITM_AMNS_SET_TABLE(amns_lr,set)
  call ITM_AMNS_RX(amns_lr,rate(:,:,0),ne,te)
...
  call ITM_AMNS_FINISH_TABLE(amns_lr))
  call ITM_AMNS_FINISH(amns)
```

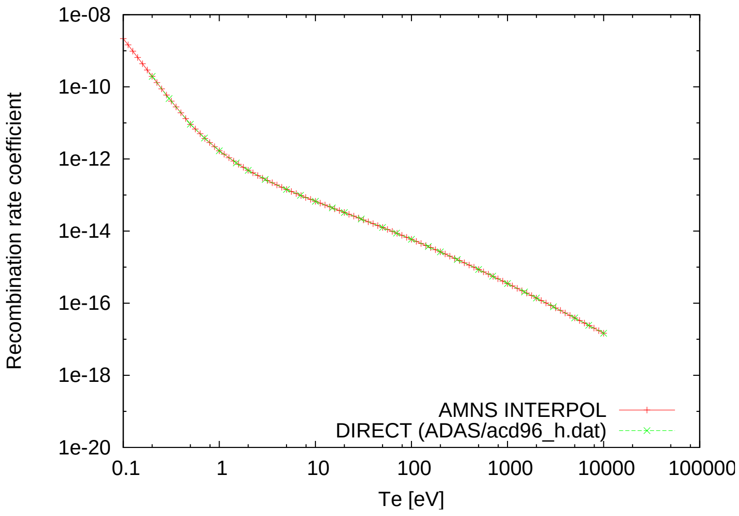
Sputtering yield based on TRIM



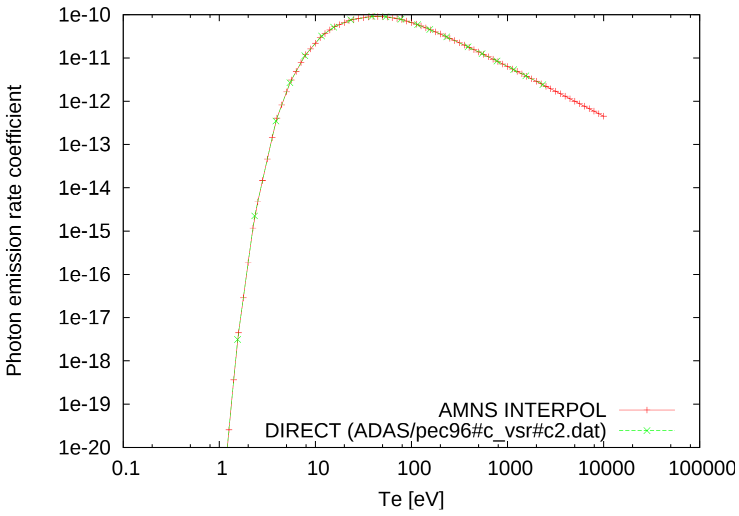
Ionization rate based on ADAS (adf11)



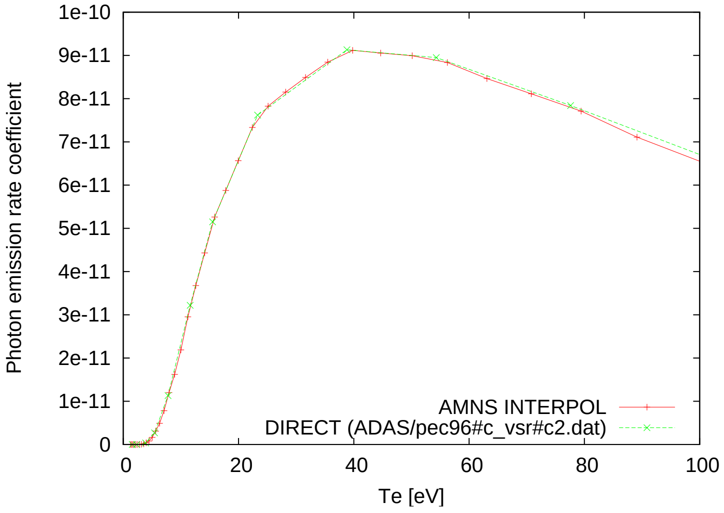
Recombination rate based on ADAS (adf11)



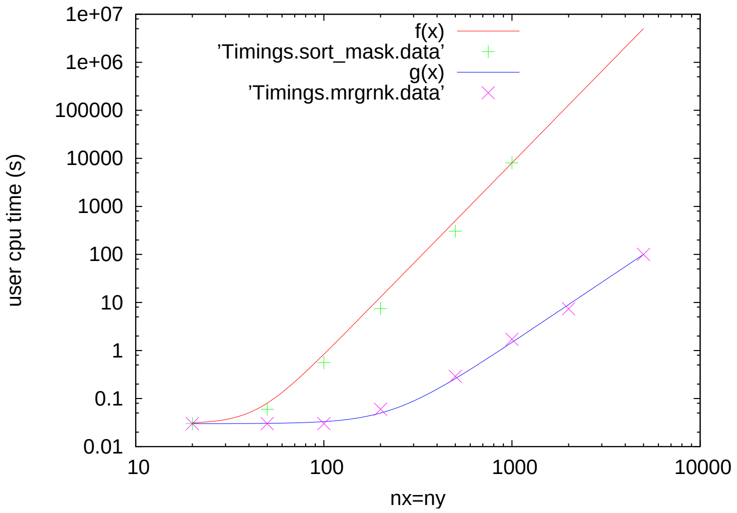
Line radiation rate based on ADAS (adf15)



Line radiation rate based on ADAS (adf15), zoom



Performance improved by replacing the sort component



(ob

from http://www.fortran-2000.com/rank/mrgrnk.f90).

Discussion

- Need to decide if this is the route forward
- If yes
  - Need to clean up the code
    - implement more error checking
    - document GRID package
    - ...
  - Need to convert AMNS data into a standard table format
    - currently done in the demo code
    - probably want to do this outside of the interface
  - Make the data available to the interface through some standardized method
    - using the UAL?
  - V&V the interface
  - implement in various codes