Streamlining the main loop

Increasing readability & maintainability

Step by step merging « roadmap »

- **develop PB** **[tag to have a comparison point before dealing with 1)]**
- **develop_openmp** [changes in develop merged - json] AMU/HLST

  -> should be merged in develop once independently tested
- **Forks/iter/species_scaling(_DR3)** IO/ISFIN

  1) import thousands of changes in blanks, hyphens, subroutines names to develop [adopt same rules as defined by IO to avoid such situation ?]

  2) Move the (few) SOLPS specific statements to the proper interface routines (e.g. avoid using interface modules in other parts of the code)

  3) import changes to non-linear iterative mode, Arhenius rates, radiation tallies … to develop & in the manual

  4) Review the implementation of new tallies introduced to allow species resolved rescaling (enforce particle conservation so as not to loose particles in EIRENE, by –slightly- rescaling densities)

  5) proceed with streamlining along the lines discussed during the meeting

Step by step merging « roadmap »

- ~~**develop PB [tag to have a comparison point before dealing with 1)]**~~
- ~~**develop_openmp** [changes in develop merged - json] AMU/HLST~~

  ~~-> should be merged in develop once independently tested~~
- ~~**Forks/iter/species_scaling(_DR3)** IO/ISFIN~~

  ~~1) import thousands of changes in blanks, hyphens, subroutines names to develop [adopt same rules as defined by IO to avoid such situation ?]~~

  ~~2) Move the (few) SOLPS specific statements to the proper interface routines (e.g. avoid using interface modules in other parts of the code)~~

  ~~3) import changes to non-linear iterative mode, Arhenius rates, radiation tallies … to develop & in the manual~~

  ~~4) Review the implementation of new tallies introduced to allow species resolved rescaling (enforce particle conservation so as not to loose particles in EIRENE, by –slightly- rescaling densities)~~

  5) proceed with streamlining along the lines discussed during the meeting

Core segregation: principle

- **What do we mean ?**

Ideally : all branching dealt with in the starter phase, then compact core just does what is strictly needed using generic routines

EIRENE is very far from this model

- **Find a pragmatic way forward in that direction so as to** :

- improve performance without loosing/damaging functionnalities (user's immediate perspective)

- part of the streamlining of the code ->lower the entry barrier to the code to facilitate further maintenance.

-

Core segregation: concrete first steps

- **Moving things that can be precalculated to the starter**

  Move pre-calculation additional surfaces to starter phase to speed up calculation (when relevant). Implemented for triangles but needs checks. Octree usable.

- **Reduce branching (esp. Geometry)**

  introduce cell & face structure types, unifying the treatment of unstructured and structured grids

  (quite some work !)

  ++ readability

  ++ takes us closer of the concepts used for IMAS gdd

  ++ less special cases to worry about

  To be noted : Toroidal faces in 2D cases need to behave as periodicity surfaces (check on phi)

  surfaces are not necessarily planes (circular, elliptic grid)

  some observations :

  Branching identified as an issue on BlueGene (strongly architecture dependent)

  geometry branching difficult to eliminate (LEVGEO but not only) and MR's exercise with pragmas showed no visible effect on performance – on the case(s) tested

Code streamlining aspects

Here : make the core MC calculation more readable as a first step

Things that make entry potential high : cryptic variable names, side effects in procedures, gotos spaghetti routines

- introduce particle type, pass as argument (locate, folneut …). Avoid as much as possible side effects in function/subroutine calls

  Would open the way for unit tests

- rename key variables if names not explicit enough (T*-> Time2*). First step could be progressively describing variables in modules

  MODCOL could also be made more explicit

- Make the modules core, starter + (post-processor) specific,

  (Use the openmp work as a guide / private vs shared variables)

- Refactor folneut.f by identifying individual actions, turn them into procedures and use a do while loop enclosing a select case construct (or possibly also recursive calls)

Timeline and assignments ? (2021)

- ~~Dealing with folneut : mid december 1 one day common work (YM, PB, JG, WD) -> Leuven~~
- Cell & faces structure
- ~~Plan meeting at IO (YM+XB)~~

Today :

- presentation of the proposed way to deal with folneut
- Way forward on geometry ?