M3DC1 SPR-Based Adapt

Outline

- New APIs
- High-level SPR-Adapt API
- New Input Variables
- Workflow in 2D
- Workflow in 3D
- Handling of multi-plane fields during 3D adapt
- 2D Pellet Example
- 3D Pellet Example

New APIs

List of Updated Functionality [Fortran Side]

1 - updated subroutine “`create_field(f,prefix)`” and “`scorec_vector_create(f,n,prefix)`” to take an (optional) `prefix` to be able to name the fields on the C++ side.
2 - added subroutines “`mark_field_for_solutiontransfer(f)`” and “`scorec_vector_mark_for_solutiontransfer(f)`” to mark a given field for solution transfer. The marking of fields is done following the same logic used to write fields to output [see subroutine “`mark_fields(equilibrium)`”].
3 - added subroutine “`adapt_by_spr(fid,idx,t,ar,maxsize,refinelevel,coarsenlevel)`” for adaptation using spr-based error estimation. See slides below for the definition of each parameter.

New APIs

List of Updated Functionality [C++ Side]

1 - added the API “`int m3dc1_scorec_verbosity(int* l)`” to be able to print out more debug information during mesh adapt. On the fortran side this is called if the variable “`iprint`” is bigger than 1.
2 - updated “`m3dc1_field`” class to have a new private variable “`bool transfer`” specifying if the field should be transferred during adaptation.
3- added the API “`int m3dc1_spr_then_adapt (...)`” which handles spr-based adaptation for both 2D and 3D cases. [See the following slides for details of how this is used]

High-level SPR-Adapt API

int m3dc1_spr_then_adapt (...)

This high-level API is responsible for performing spr-based adaptivity for both 2D and 3D cases. The input variables are as follow:

- `FieldID* field_id  - the aggregate field used for spr`
- `int* index         - the index of the specific field in field_id`
- `int* ts            - time step`
- `double* ar         - error weight [should be smaller than 1.]`
- `double* max_size   - maximum edge length allowed`
- `int* refine_level  - maximum refinement allowed`
- `int* coarsen_level - maximum coarsening allowed [use -1 to disable]`

New Input Variables

Input variables to enable adaptivity in C1input files

- **ispradapt** (default 0): 1 to turn on spr-adapt and 0 to turn it off
- **isprntime** (default 10): an integer bigger than 0 to call spr-adapt every isprntime time steps
- **isprweight** (default 0.1): a real number smaller than 1 specifying the weight of error to calculate the size-field (smaller weights results in more aggressive adaptation).
- **isprmaxsize** (default 0.05): user-specified maximum allowed edge length
- **isprrefinelevel** (default 1): maximum level of refinement allowed during each call to spr-adapt [e.g. 3 means the lengths of the edges will not be reduced by a factor of more than 8(= 2^3)]
- **isprcoarsenlevel** (default -1): maximum level of coarsening allowed during each call to spr-adapt. use -1 to disable coarsening altogether.

Workflow in 2D

int m3dc1_spr_then_adapt (...) in 2D

- extract the **“target”** field from the aggregate field
- compute the **“gradient of target field”** at integration points using the implementation of reduced quintic basis functions on the C++ side
- compute the **“size field”** using the gradient field computed above and spr procedures in PUMI
- add fields that are marked for solution transfer to the **“solution transfer object”** for mesh adapt
- configure mesh adapt with the **“solution transfer object”** and the **“size field”**
- call **2D mesh adapt**

Workflow in 3D

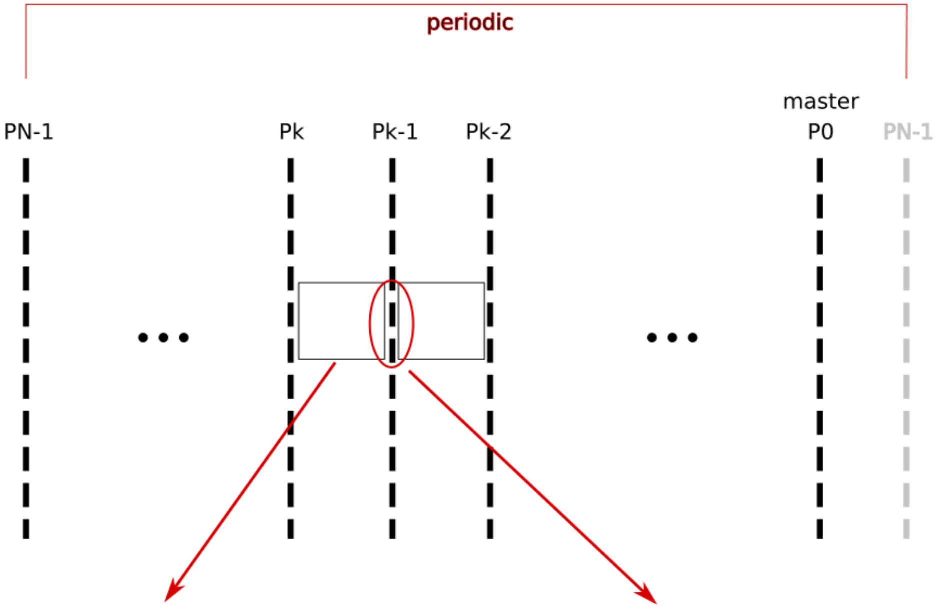
int m3dc1_spr_then_adapt (...) in 3D with N planes

- extract the **“target”** field from the aggregate field
- if marked for solution transfer, **transfer fields from the non-master planes to the master plane**
- remove all the entities that are not on the master plane
- compute N **“gradient of target fields”** for each of the planes at integration points
- compute the N **“size fields”** using the gradient fields above and spr procedures in PUMI
- at each vertex on the master plane set the size field to the minimum of the N size fields computed for each plane
- add all the fields to the **“solution transfer object”** for mesh adapt
- configure mesh adapt with the **“solution transfer object”** and the **“size field”**
- call **2D mesh adapt**
- reconstruct the 3D mesh
- **transfer fields from the master plane to the non-master planes** in the reconstructed 3D mesh

Handling of Multi-Plane Fields During 3D Adapt

Transfer of fields to/from the master plane

- Transfer of fields to/from the master plane happen in an iterative manner using a combination of remote entities in the parallel mesh and the individual wedge elements that connect the planes
- To transfer a field from the **k**th plane to the master plane the steps shown on the right picture are repeated **k** times.
- Transfer of fields from the master plane to the other planes is done by first transferring everything to the last plane (PN-1) and then repeating the above procedure.



1 - Within a process:

use downward adjacency to transfer data from Pk to Pk-1 [on the left side]

2 - Between processes:

use remote copy information to transfer from the left side of Pk-1 to the right side

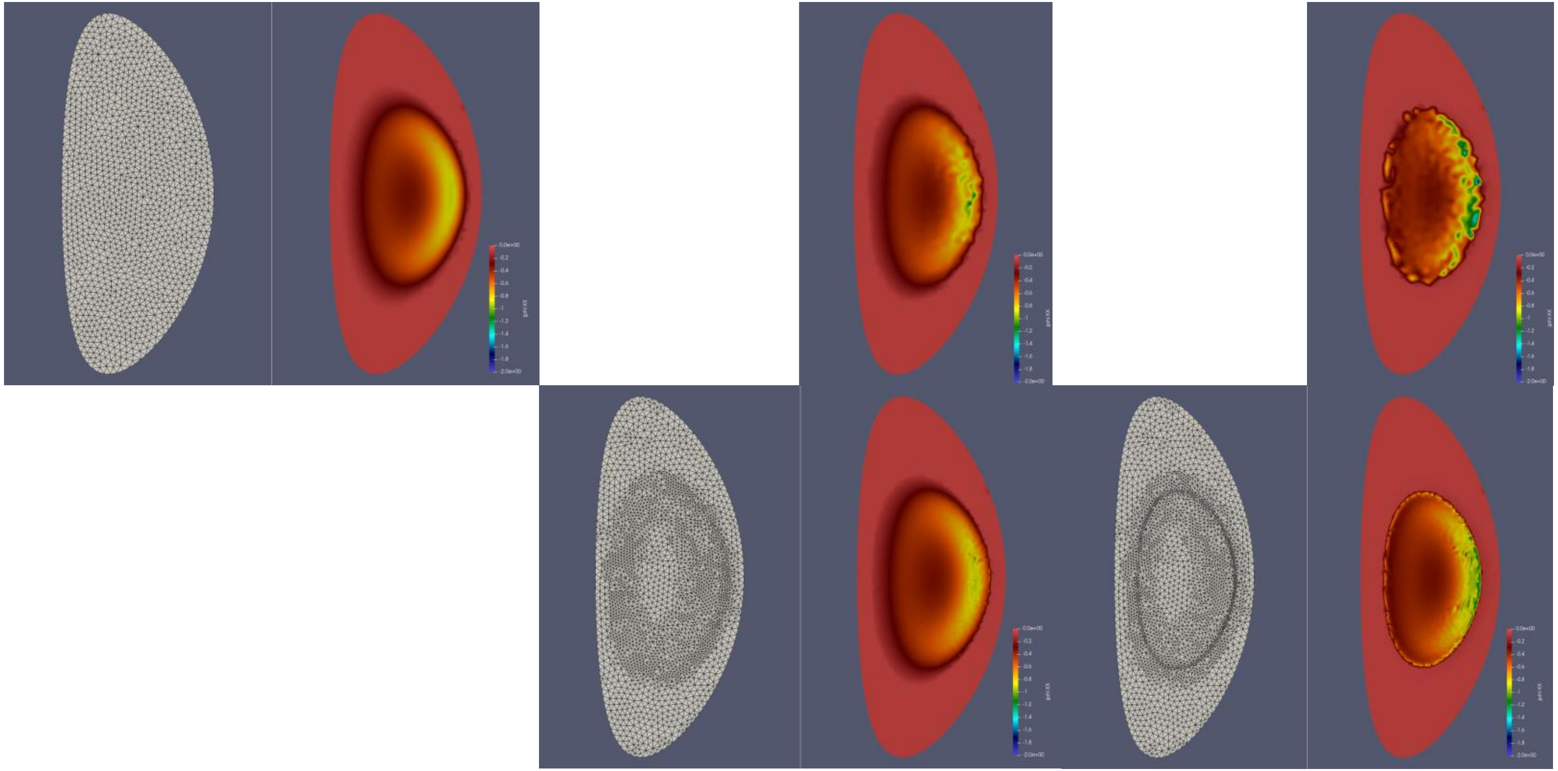
2D Pellet Example

- field used for spr error estimation $\mathrm{grad}(\psi)$
- spr and adapt are called every 10 time steps
- case is run for 500+ time steps
- mesh/field (jphi) are shown at time steps 10, 110, 210, 310, 410, 510
- result are shown for solve on the initial mesh and adaptive solve runs for comparison

Meshes/Solutions

Initial mesh and solutions on it

Adapted mesh and solutions on it



Time = 10

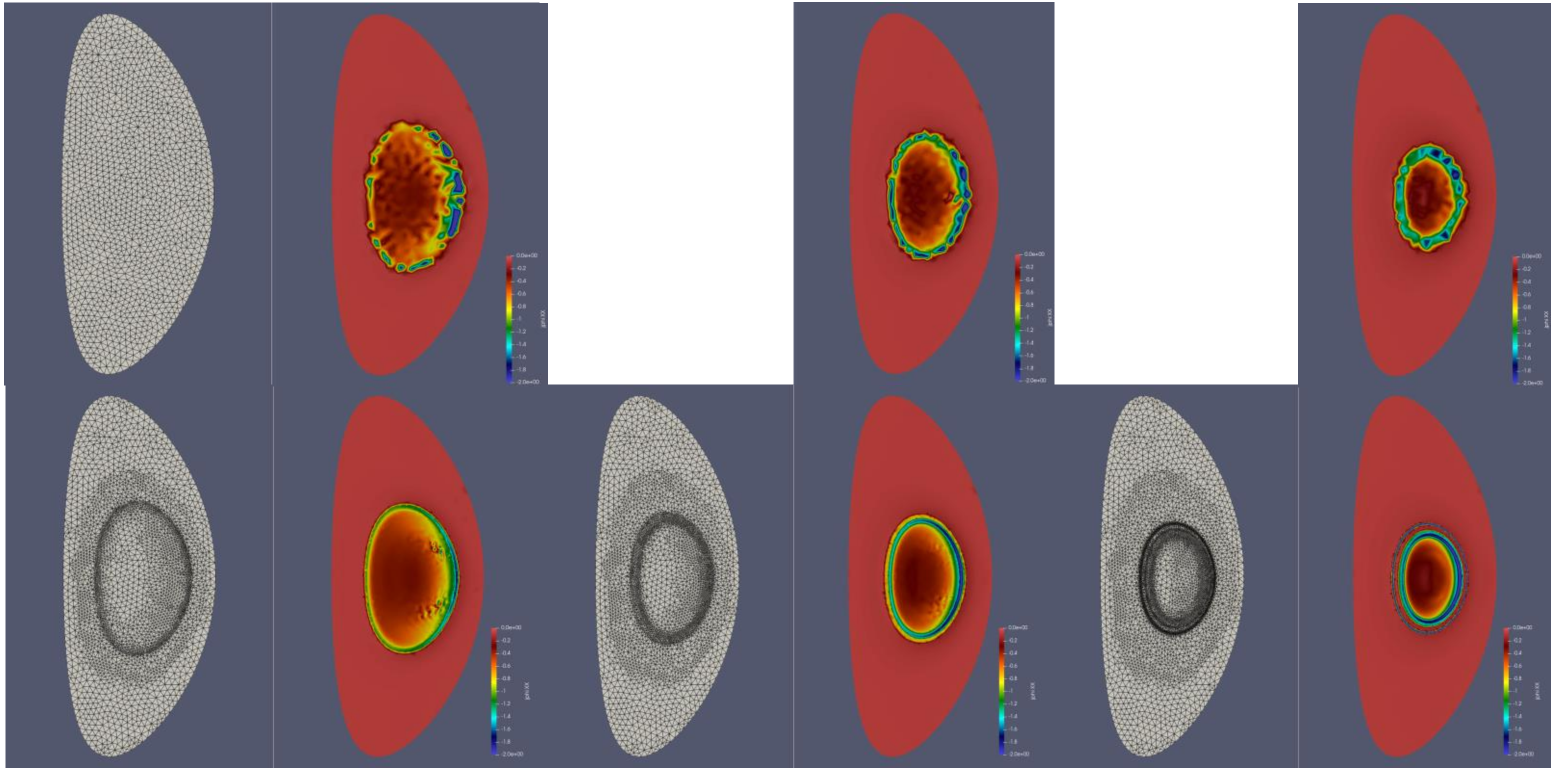
Time = 110

Time = 210

Meshes/Solutions

Initial mesh and solutions on it

Adapted mesh and solutions on it



Time = 310

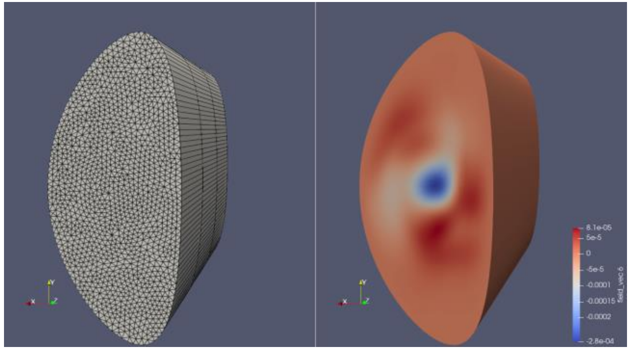
Time = 410

Time = 510

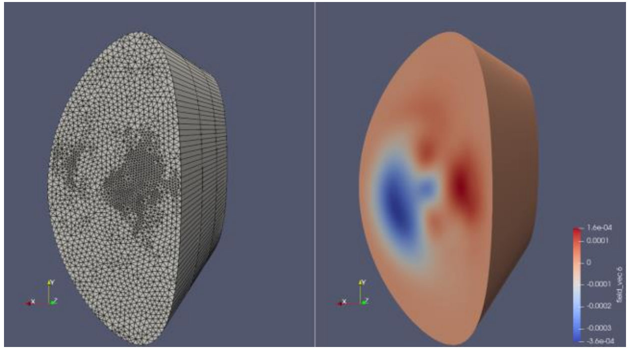
3D Pellet Example

- field used for spr error estimation $\mathrm{grad}(\psi)$
- spr and adapt are called every 5 time steps
- case is run for 30 time steps
- mesh/field (jphi) are shown at time steps 5, 10, 15, 20, 25, 30
- result are shown for solve on the initial mesh and adaptive solve runs for comparison

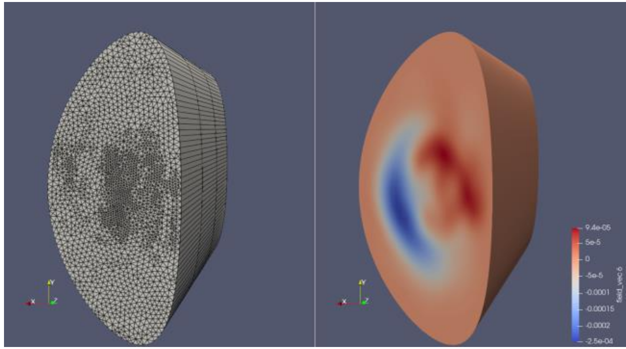
3D Adaptive result



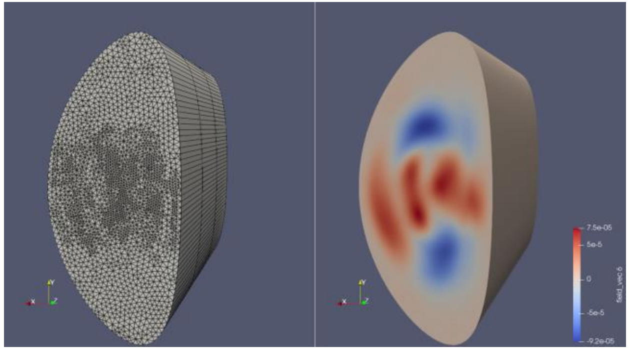
ntime=5



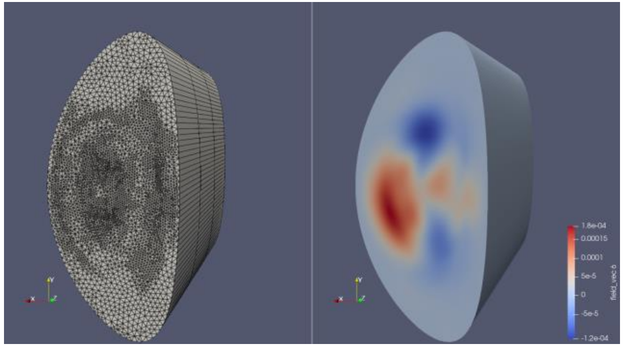
ntime=10



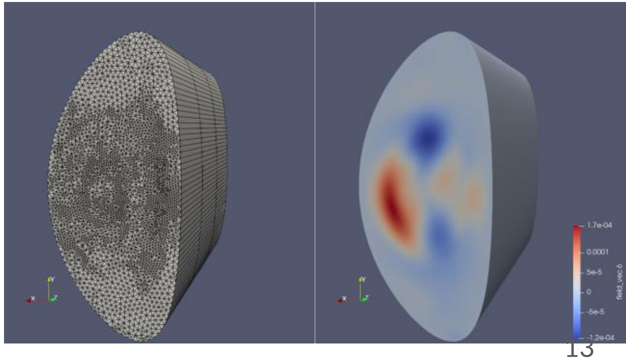
ntime=15



ntime=20



ntime=25



ntime=10