Bay-Area Radiation Transport (BART), a Research-purpose Parallel Transport Code Framework

Weixiong Zheng[1], Joshua Rehak[1], Rachel Slaybaugh[1]

[1]Nuclear Engineering, University of California, Berkeley

Introductions

optionally give it a frame name

optioanlly give it a block name

- Fill this section up.

Equations and Discretizations

optionally give it a frame name

optioanlly give it a block name

- Fill this section up.

Finite Elements and Linear Algebra

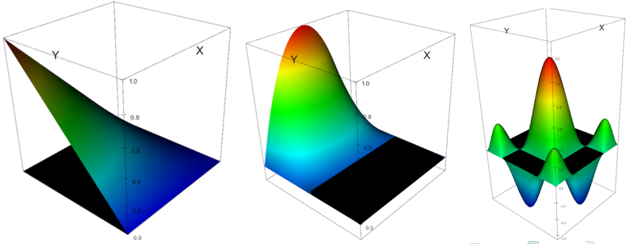
Finite elements in BART

Finite elements in general dimensions

- deal.II supports finite elements in general dimensions by templates
  - BART developers only need to call generic trial functions when implementing weak forms for 1/2/3 D
  - Specs of trial functions are hidden under the hood by deal.II for different dimensions.
- BART supports DFEM, CFEM, FV and RTk.
  - For high-order-low-order (HOLO), BART can assign individual finite elements to different equations.
  - All you need to do is to tell BART in input file:

```
set ho spatial discretization = cfem
set nda spatial discretization = dfem
```

- Polynomial orders can be changed in input file as well (see the following demos for Q1(left), Q2(middle) and Q4(right) trial functions)



Linear algebra in BART

Sparse matrix-vector product based computations

- Current implementation of BART assembles global matrices and utilize sparse matrix-vector product in linear algebraic solvers.
    - Easy implementation.
    - High computational efficiency with (bi/tri-) linear elements.

BART is interfaced with PETSc

- Most PETSc solvers/preconditioners wrapped in deal.II are used in PreconditionerSolver class of BART
    - Direct solver: parallel direct solver MUMPS
    - Iterative solvers and preconditioners
- Performance remedy: as solving will happen multiple times due to source/power iterations, we initialize the preconditioning/factorization only once then preconditioning/factorization matrices will be stored for reuse.

Meshing Capability

Homogenized and pin-resolved meshing capability in parallel

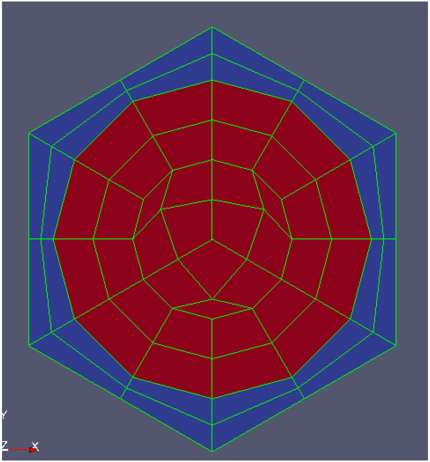
BART was initially implemented for homogenized mesh

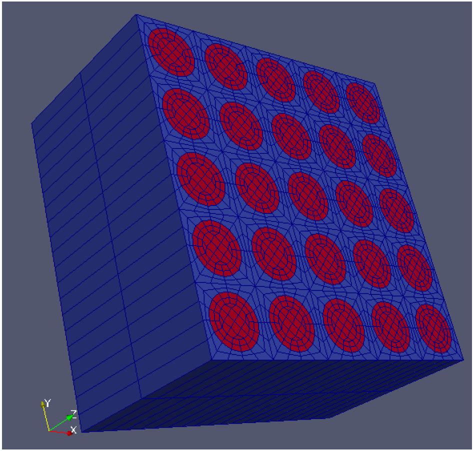
- Hyper-rectangle meshing based on `deal.II`:
  - Lines in 1D, rectangles in 2D and regular cuboid in 3D
  - Material ID assigned to coarsest mesh and stored in cell objects tractable when refining

Pin-resolved meshing

- Recent development enables the use of pin-resolved mesh
  - Rectangular (prism) pin is supported; hexagonal (prism) pin is under development
  - **Goodness:** meshing does **NOT** depend on `Cubit` or `gmsh`. `BART` realizes wrapper functions based on `deal.II` to draw complex geometries.
- We compose different pin models and replicate based on pin types in 2D.
- 3D meshes is realized by extruding 2D mesh.

Pin-resolved mesh demos

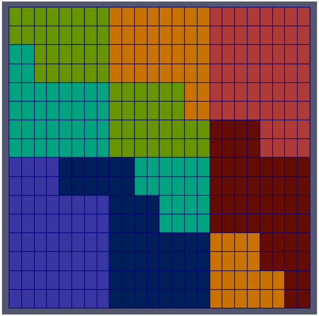


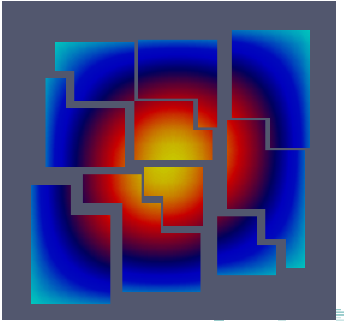


Meshing in Parallel

Distributed triangulation

- Triangulation (meshing) needs to support parallelism for parallel computations.
- deal.II supports two ways of triangulation in parallel
  - Shared (ParMETIS based): every processor has a copy of the global triangulation.
  - Distributed (p4est based): every processor only knows cells living on itself and a layer of neighboring cells from other processors on the local triangulation boundary
- BART supports distributed meshing from deal.II.
- 1D meshing is serial as deal.II has no parallel support





Unit Testing, documentation Continuous Integration and Code Coverage

Unit testing and documentation

We document and test everything possible

- We rewrote BART twice:
  - First time, we restructured BART and documented everything with doxygen.
  - Second time, we added unit testing.
- Philosophy: everything be documented and every function/class be tested if possible.
  - Documentation leads to better understandability of code in the future development.
  - Unit testing ensures new codes do not affect correctness of existing code.

G(oogle)Test and CTest are both used

- We want unit testing to be efficient and compatible with MPI
  - GTest is super efficient but hard to obtain compatibility with MPI.
  - CTest is slow but compatible with MPI.
- Not all the testings require MPI
  - We use GTest for all serial testing
  - We leave all MPI related testings to CTest.

Conclusions

optionally give it a frame name

optioanlly give it a block name

- Fill this section up.