# 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

# What is **BART**?

## A open-source research-purpose code

- We hold **BART** on Github with MIT license.
- We build **BART** to be a research-purpose transport code.
- We aim to provide a framework s.t. graduate students only need necessary amount of knowledge on **C++** and third-party libraries to implement new ideas for research

## A finite element code based on **deal.II**

- We build **BART** to be a finite element code based on **deal.II**
  - Finite element is wired-shape mesh friendly
  - **BART** computes in general dimension as **deal.II** does.
  - **BART** only call generic functions instead of dimension specified ones.
- Any specs of finite elements are wrapped by **deal.II** s.t. **BART** developers focus only on physical/symbolically mathematical aspects.

## A code in parallel

- We build **BART** to be a parallel code computing on distributed memory
  - Even small-size problems (¡10 million DoFs) can be overwhelming for local computers
- It is natural to enable parallelism as **deal.II** has nice wrappers.

# Principles for development

- Coding style: [Google Style](google.com)

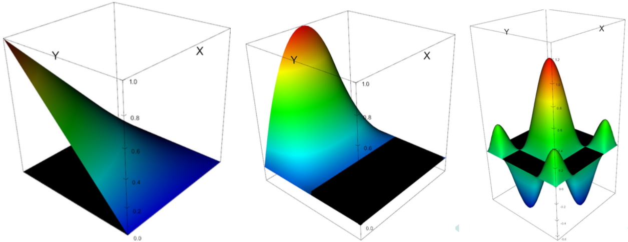
# Finite elements in **BART**

## 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)



# What can **BART** do?

## What approximations can **BART** have?

- Transport approximations that can be decoupled to individual equations
  - Discrete ordinates approximation
  - Diffusion equations
  - Canonical form of simplified spherical harmonics (SPN)
- PN is feasible through extension of current framework.

## Solve small-to-median-sized problems

- A transport code doing all real-world large problems with billions/trillions of degrees of freedom is charming, BUT
  - It can require tens/hundreds of man-year of work.
  - It requires tons of optimizations.
  - It is hard for newbies to get started
- We restrict **BART** to small (e.g. one-group) to median sized problems (e.g. C5G7)

# Equations/Discretizations

# Equations BART solves transport equation

## BART is solving multi-group transport equation with discrete-ordinates in angle

- Transport equation in operator form with:

$$\boldsymbol{\mathcal{T}}_{\mathrm{g,m}}\psi_{\mathrm{g,m}} + \boldsymbol{\mathcal{C}}_{\mathrm{g,m}}\psi_{\mathrm{g,m}} = \sum_{\substack{1\le \mathrm{g}'\le \mathrm{G}\\ 1\le \mathrm{m}'\le \mathrm{M}}} \left(\boldsymbol{\mathcal{S}}_{\mathrm{g}',\mathrm{m}'\to \mathrm{g,m}} + \boldsymbol{\mathcal{F}}_{\mathrm{g}',\mathrm{m}'\to \mathrm{g,m}}\right)\psi_{\mathrm{g}',\mathrm{m}'} \tag{1}$$

$$\psi_{\mathrm{g,m}} = \psi^{\mathrm{inc}}_{\mathrm{g,m}},\ \vec{r}\in\partial\mathcal{D},\ \vec{n}\cdot\vec{\Omega}_{\mathrm{m}} < 0$$

$\boldsymbol{\mathcal{T}}$: Streaming operator
$\boldsymbol{\mathcal{C}}$: Collision operator, $\sigma_{\mathrm{t}}$
$\boldsymbol{\mathcal{S}}$: Scattering operator
$\boldsymbol{\mathcal{F}}$: Fission operator
$\boldsymbol{\psi}$: Angular flux
$\mathrm{g},\ \mathrm{g}'$: Group indicies
$\mathrm{m},\ \mathrm{m}'$: Angular indices
$\mathrm{G},\ \mathrm{M}$: Numbers of groups and directions

- The formulation generalizes to diffusion and nonlinear diffusion

# **BART** solves first-order form of transport equations

## First-order form and DFEM discretization in space

- Transport equation has differential order of 1 in the streaming term, we refer it to as "first-order" form (FOF)

$$\vec{\Omega}_{\mathrm{m}} \cdot \nabla\psi_{\mathrm{g,m}} + \sigma_{\mathrm{t,g}}\psi_{\mathrm{g,m}} = \mathcal{Q}_{\mathrm{g,m}}\left(\boldsymbol{\Psi}\right) \tag{2}$$

$$\mathcal{Q}_{\mathrm{g,m}} := \sum_{\substack{1\leq \mathrm{g}'\leq \mathrm{G}\\ 1\leq \mathrm{m}'\leq \mathrm{M}}} \left(\boldsymbol{\mathcal{S}}_{\mathrm{g}',\mathrm{m}'\to\mathrm{g,m}} + \boldsymbol{\mathcal{F}}_{\mathrm{g}',\mathrm{m}'\to\mathrm{g,m}}\right)\psi_{\mathrm{g}',\mathrm{m}'} \tag{3}$$

- FOF is discretized using DFEM in **BART**: given polynomial function space $\mathcal{V}$, $\forall\psi^*_{\mathrm{g,m}} \in \mathcal{V}$, find $\psi_{\mathrm{g,m}} \in \mathcal{V}$, s.t.

$$\begin{aligned}&\sum_{\mathrm{e}\in\mathcal{D}}\left[\left(-\vec{\Omega}_{\mathrm{m}}\cdot\nabla\psi^*_{\mathrm{g,m}},\psi_{\mathrm{g,m}}\right)+\left(\psi^*_{\mathrm{g,m}},\sigma_{\mathrm{t,g}}\psi_{\mathrm{g,m}}\right)\right]_{\mathrm{e}}+\sum_{\mathrm{f}\in\mathcal{D}}\left|\vec{n}\cdot\vec{\Omega}_{\mathrm{m}}\right|\left\langle\left[\psi^*_{\mathrm{g,m}}\right],\tilde{\psi}_{\mathrm{g,m}}\right\rangle_{\mathrm{f}}\\&+\sum_{\substack{\mathrm{f}\in\partial\mathcal{D}\\ \vec{n}\cdot\vec{\Omega}_{\mathrm{m}}>0}}\vec{n}\cdot\vec{\Omega}_{\mathrm{m}}\left\langle\psi^*_{\mathrm{g,m}},\psi_{\mathrm{g,m}}\right\rangle_{\mathrm{f}}=\sum_{\mathrm{e}\in\mathcal{D}}\left(\psi^*_{\mathrm{g,m}},\mathcal{Q}_{\mathrm{g,m}}\right)_{\mathrm{e}}+\sum_{\substack{\mathrm{f}\in\partial\mathcal{D}\\ \vec{n}\cdot\vec{\Omega}_{\mathrm{m}}<0}}\left|\vec{n}\cdot\vec{\Omega}_{\mathrm{m}}\right|\left\langle\psi^*_{\mathrm{g,m}},\psi^{\mathrm{inc}}_{\mathrm{g,m}}\right\rangle_{\mathrm{f}}\end{aligned} \tag{4}$$

# **BART** solves second-order forms of transport equations

## Second-order forms (SOF) of transport equations

- FOF can be cast to diffusion-like equations having streaming term with differential order of 2
- The casting allows for use of CFEM
- BART solves even-parity equation and self-adjoint angular flux equation (SAAF)

## SOF example: SAAF

- SAAF equation

$$-\vec{\Omega}_\mathrm{m}\cdot\nabla\frac{1}{\sigma_\mathrm{t,g}}\vec{\Omega}_\mathrm{m}\cdot\nabla\psi_\mathrm{g,m}+\sigma_\mathrm{t,g}\psi_\mathrm{g,m}=\mathcal{Q}_\mathrm{g,m}-\frac{1}{\sigma_\mathrm{t,g}}\vec{\Omega}_\mathrm{m}\cdot\nabla\mathcal{Q}_\mathrm{g,m} \tag{5}$$

- CFEM discretization

$$\left(\vec{\Omega}_\mathrm{m}\cdot\nabla\psi^*_\mathrm{g,m},\frac{1}{\sigma_\mathrm{t,g}}\vec{\Omega}_\mathrm{m}\cdot\nabla\psi_\mathrm{g,m}\right)_\mathcal{D}+\left(\psi^*_\mathrm{g,m},\sigma_\mathrm{t,g}\psi_\mathrm{g,m}\right)_\mathcal{D}+\sum_{\substack{\mathrm{f}\in\partial\mathcal{D}\\ \vec{n}\cdot\vec{\Omega}_\mathrm{m}>0}}\vec{n}\cdot\vec{\Omega}_\mathrm{m}\left\langle\psi^*_\mathrm{g,m},\psi_\mathrm{g,m}\right\rangle_\mathrm{f}$$
$$=\left(\psi^*_\mathrm{g,m}+\frac{\vec{\Omega}_\mathrm{m}\cdot\nabla\psi^*_\mathrm{g,m}}{\sigma_\mathrm{t,g}},\mathcal{Q}_\mathrm{g,m}\right)_\mathcal{D}+\sum_{\substack{\mathrm{f}\in\partial\mathcal{D}\\ \vec{n}\cdot\vec{\Omega}_\mathrm{m}<0}}\left|\vec{n}\cdot\vec{\Omega}_\mathrm{m}\right|\left\langle\psi^*_\mathrm{g,m},\psi^\mathrm{inc}_\mathrm{g,m}\right\rangle_\mathrm{f} \tag{6}$$

# **BART** solves diffusion and nonlinear diffusion

## BART solves diffusion equation

- Diffusion could be solved within the framework of **BART**

$$-\nabla \frac{1}{3\sigma_{\mathrm{t,g}}} \cdot \nabla\phi_{\mathrm{g}} + \sigma_{\mathrm{t,g}}\phi_{\mathrm{g}} = Q_{\mathrm{g}} := \sum_{\mathrm{g}'}\left[\left(\sigma_{s,g'\to g} + \frac{\chi_{\mathrm{g}}\nu\sigma_{\mathrm{f,g}'}}{k_{\mathrm{eff}}}\right)\phi_{\mathrm{g}'}\right] \tag{7}$$

## Nonlinear diffusion for acceleration (NDA)

- Diffusion can also be written in $\mathrm{P}_1$ form

$$\nabla\cdot\vec{J}_{\mathrm{g}} + \sigma_{\mathrm{t,g}}\phi_{\mathrm{g}} = Q_{\mathrm{g}}, \quad \vec{J}_{\mathrm{g}} = -\frac{1}{3\sigma_{\mathrm{t,g}}}\nabla\phi_{\mathrm{g}}\ \text{(Fick's law)} \tag{8}$$

- NDA is derived from correcting Fick's law using transport corrections
- $\boxed{\textit{Correction}}$ to preserve current is derived from transport high-order solutions (HO)

$$\vec{J}_{\mathrm{g}} = -\frac{1}{3\sigma_{\mathrm{t,g}}}\nabla\phi_{\mathrm{g}} + \boxed{\frac{\sum_{\mathrm{m}<M} w_{\mathrm{m}}\vec{\Omega}_{\mathrm{m}}\vec{\Omega}_{\mathrm{m}}\cdot\nabla\psi_{\mathrm{g,m}}^{\mathrm{HO}} + \frac{1}{3\sigma_{\mathrm{t,g}}}\nabla\phi_{\mathrm{g}}^{\mathrm{HO}}}{\phi_{\mathrm{g}}^{\mathrm{HO}}}\phi} \tag{9}$$

# BART solves diffusion and nonlinear diffusion (cont'd)

## BART solves in multiple ways

- CFEM is natural to use to solve diffusion/nonlinear diffusion
- DFEM is realizable through penalty method
  - Useful when accelerating transport solves with DFEM
- A hybrid-FEM is a near-future project for NDA in $P_1$-like form
  - Piece-wise constant test function for $\phi_\mathrm{g}$ while Raviat-Thomas test functions for $\vec{J}_\mathrm{g}$

# Parallelism/Linear Algebra/Meshing

# BART is designed/implemented to be a parallel code

## BART computes on distributed memory

- Message Passing Interface, aka **MPI**, is used for distributed computations.
- Meshing is correspondingly distributed based on **p4est**'s functionality wrapped by **deal.II**.
  - Each processor mainly knows mesh cells on itself
- Linear algebra related objects are distributed as well
  - **PETSc** data structure wrappers in **deal.II** are heavily used to enable the parallel linear algebra.

## BART is parallelizing in space

- While extending parallelism to be suitable for other dimensions in phase space, we currently only parallelize in space
  - No special treatment on MPI/scheduling, natural support from **deal.II**
- Computational efficiency in parallel rather depends on solvers/preconditioners, but little on the mesh

# 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 in **BART**

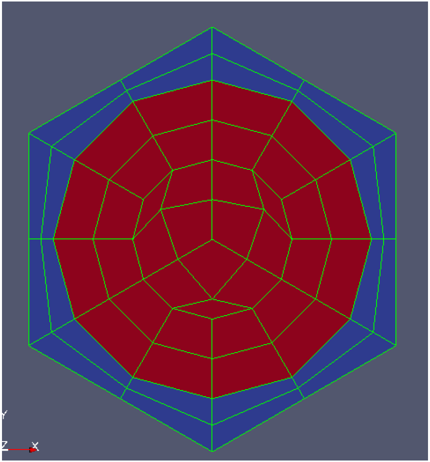
## BART was initially implemented for homogenized mesh

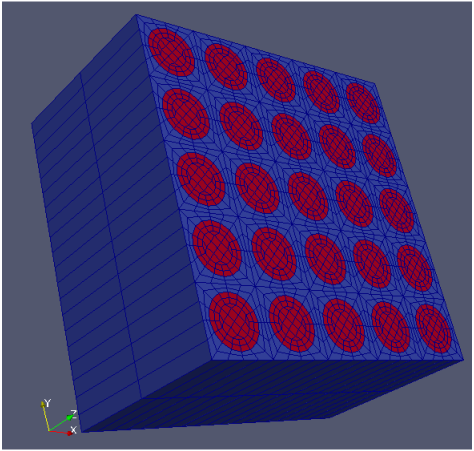
- Hyper-rectangle meshing based on **deal.II**:
  - Lines in 1D, rectangles in 2D and regular cuboid in 3D
  - Material ID assigned to coarsest mesh and stored in cell objects tractable when refining

## 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

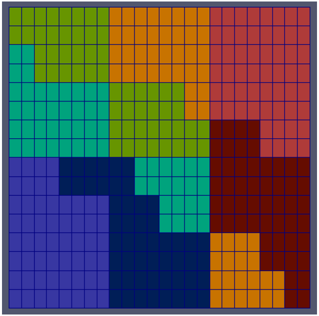


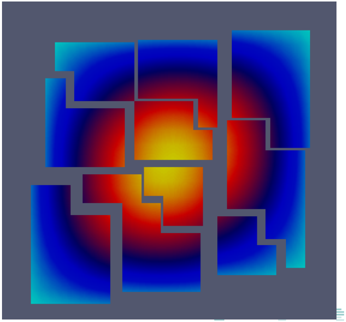



# Meshing in Parallel

## 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





# Testing

# 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.

# Ongoing Projects

# We are conducting projects on **BART**

## Ongoing projects

- Advanced spatial discretization methods.
- Advanced energy acceleration methods.

## We are designing students' projects in **BART**

- We are designing students' projects based on **BART**
  - **BART** will grow, so intellectually do students.