Goal:

- Simple Implementation:
  - Discretization method,
  - Quadrature set,
  - Geometry,
  - Dimensionality.

- Modify TAXI infrastructure to have multi-dimensional and multi-geometry support:
  - If possible: template current structures
  - Otherwise: create new objects

- Add only the new sweepchunk and quadrature subroutines.
  - Currently: WDD, XY geometry, Level Symmetric.
  - Next: WDD, RZ, Level Symmetric.
Goal (Cont.)

- 2 D case:
  - Test:
    - WDD method
    - XY geometry
    - Level Symmetric quadrature.
  - Expand to:
    - RZ WDD: angular differencing.
    - 2D Corner Elements: new discretizations.
Code Modifications Overview

Spatial Discretization Method:
- XY_Wt_Diamond_Difference
- Wt_Diamond_Difference

Cells:
- RectangularCell
- HexahedralCell

Edges:
- 2D_Edge
- 3D_Edge

Grid:
- ArbitraryPGrid

PRange Manager

Quadrature:
- LSquad

Logically Rectangular Grid
- Logically Hexahedral Grid

Inputs
- Templated
- Rewritten
Spatial Discretization Method

- Quadrature
- Cells
- Edge

- $\mu, \eta, \omega t$
- $\Sigma_t \Delta x \Delta y$
- $\psi_{inc}$
- $\psi_{mp}, \psi_{ext}$

Energy
Angle
Elements
Cells
Quadrature

- Organize quadrature information:
  - Quadrature type (Geometry + Number of Dimensions).
  - Angle and Weight Information.
  - Number of Levels.
TAXI Data Structures

- Spatial domain decomposed into Cells
  - Cells are stored in a grid
  - Cell contains Elements
- Elements are the base spatial data structure
Grid Overview

- Grid is templated on
  - cell type (Rectangular, Hexahedral, …)
  - element type (whole_cell, corner, …)
  - Edge type (2D, 3D,…)

- It contains instantiations of
  - cells (one per graph_vertex)
  - faces (really graph_edges)

- It contains methods for
  - putting outgoing-surface intensities into graph_edges
  - getting incoming-surface intensities from graph_edges
Grid

- Base grid class templated with the type of cell and the type of edge
  \[ \text{ArbitraryPGrid}<\text{CellType}, \text{ElementType}, \text{EdgeType}> \]

- Specific grids inherit from `ArbitraryPGrid`
  - `LogicallyRectangularGrid`
  - `LogicallyHexahedralGrid`

- Example:
  ```java
  Template <class CellType, class ElementType>
  Class LogicallyRectangularGrid :
      public ArbitraryPGrid<CellType, ElementType, 2D_Edge>
  ```
Cell Overview

- Cell class doesn’t contain much.
  - list of isotopes and their number densities
  - list of elements
- The “element” concept gives us great generality and flexibility.
- There is one element for each “fundamental” spatial unknown. For example:
  - simple methods with one unknown per cell use the “whole-cell” element
  - methods with one unknown per vertex per cell use the “corner” element
Cell and Edge

- **Cell**
  - BaseCell – interface that all the cells have to implement.
  - Geometry: face normals, corners, face centers.
  - Example:
    ```cpp
    Class RectangularCell : public Rectangle, BaseCellwElement{};
    Class RectangularCell : public Hexahedra, BaseCellwElement{};
    ```

- **Edge**
  - BaseEdge, 2D_Edge, 3D_Edge
PRange Manager

- Interacts with Grid, Scheduler and Quadrature in order to build the pRanges that will be used for sweep.
- PRange Manager was changed to interact in a generic way with geometry specific classes (Quadrature) in order to get the sweep directions for ordering the DDGs.
Summary

- Simple implementation of new geometries, dimensionalities and grid type:
  - \textit{Cell, edge, quadrature, grid}

- Simple Implementation of new spatial discretization method:
  - \textit{sweepchunk}

- 2D modification was an implementation and test of code feature.

- Apply to RZ Weighted Diamond Difference and 2D Corner Element methods.