

---

# Spatial Discretization

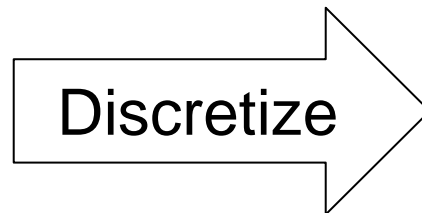
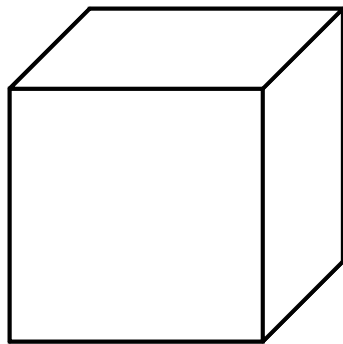
Toru Furukawa  
Department of Nuclear Engineering  
Texas A&M University

# Spatial Discretization

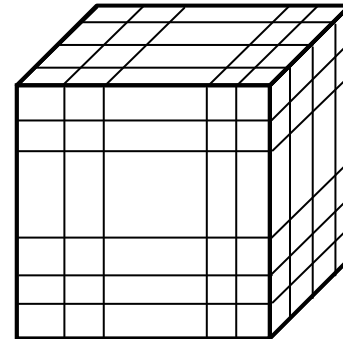
---

- We discretize a spatial domain into cells
- Density at point  $(x, y, z)$   
→ Density in cell  $ijk$

$\psi(x, y, z)$



$\psi_{ijk}$



# Sweep Methods

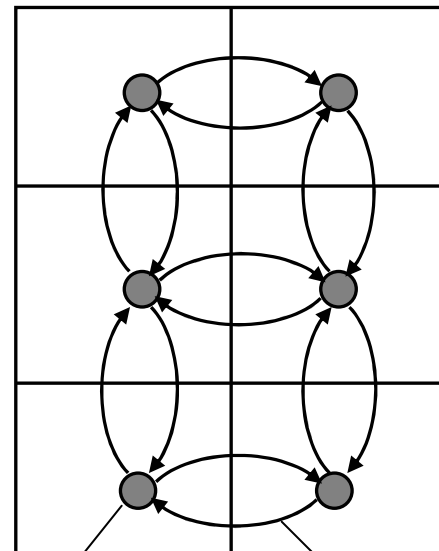
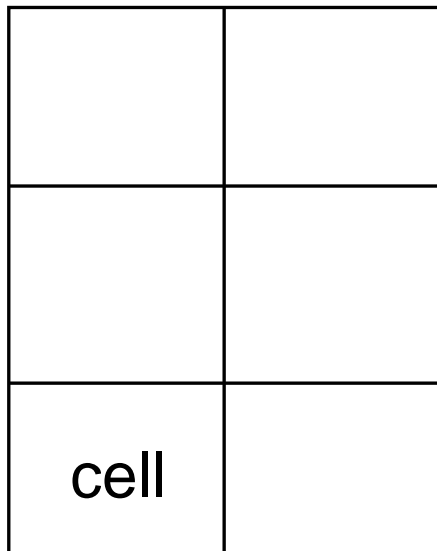
---

- Sweep method =  
    spatial discretization method +  
    assumption about spatial grid
- Every sweep method expects certain  
    {grid, cell, element} types
- We want simple interface

# Spatial Grid Representation

---

- Spatial grid is represented as graph



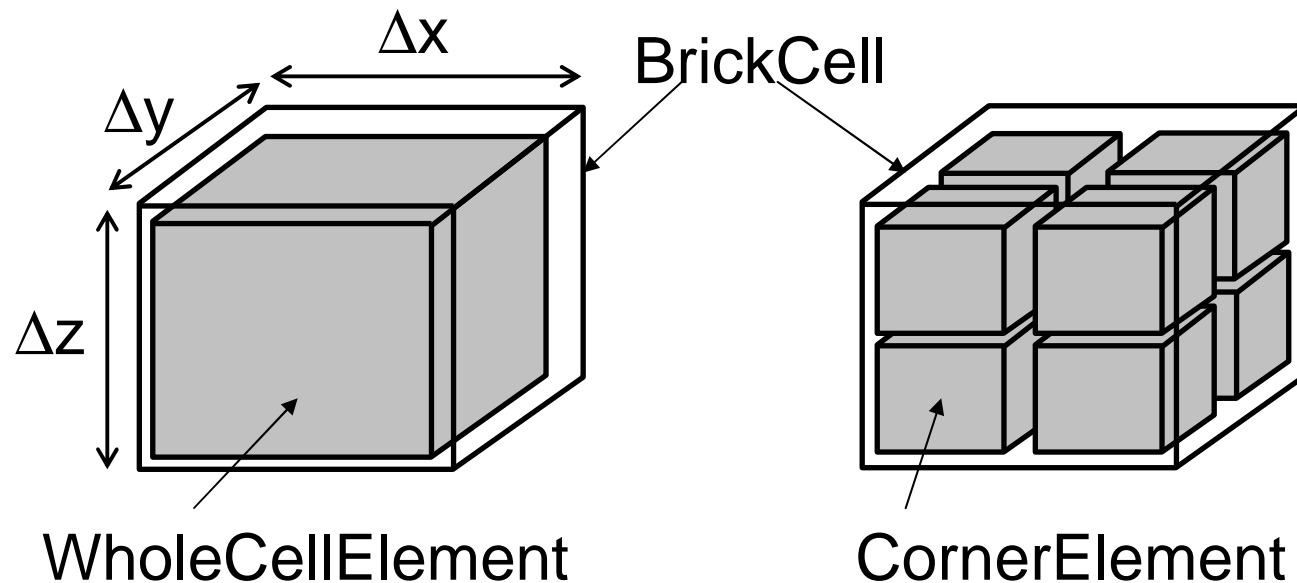
Graph-vertex

Graph-edge

# Cell and Element

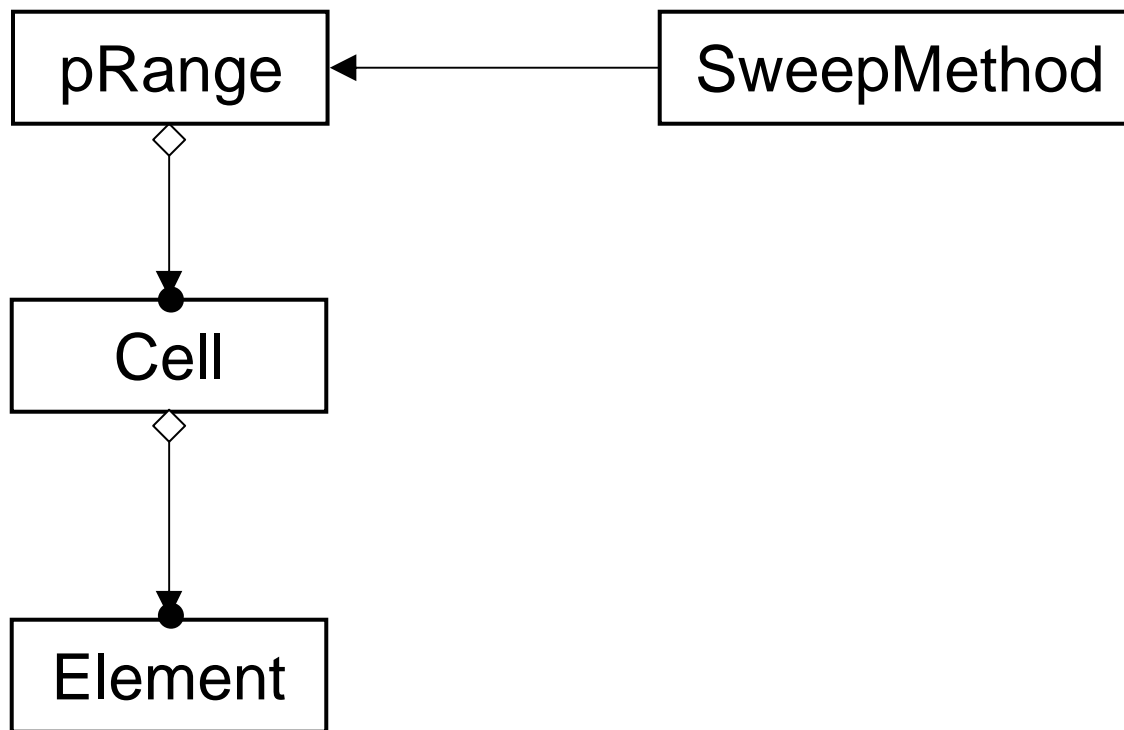
---

- A cell has geometric information, material properties, and element(s)
- 1 element per fundamental unknown



# pRange, Cell, Element

---



# Sweep Method

---

- Sweep method receives a graph and traverses cells to solve

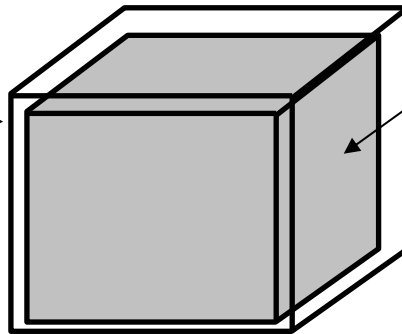
```
void SweepMethod(pRange& chunk) {  
    ...  
    pRange::iteratorType cell_it;  
    for (cell_it = chunk.get_boundary().start();  
         cell_it != chunk.get_boundary().finish();  
         ++cell_it) {  
        ... solve ...  
    }  
    ...  
}
```

# Weighted Diamond Diff. Method

---

```
void WtDiamondDiffMethod(pRange& chunk) {  
    ...  
    pRange::iteratorType cell_it;  
    for (cell_it = chunk.get_boundary().start();  
         cell_it != chunk.get_boundary().finish();  
         ++cell_it) {  
        ... solve WD balance equation ...  
    }  
    ...  
}
```

BrickCell



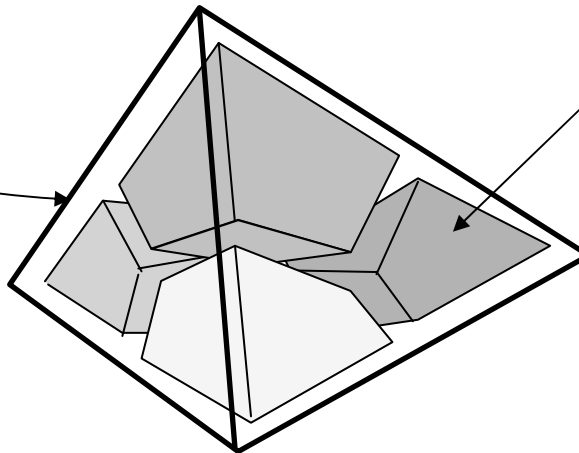
WholeCellElement

# Simple Corner Balance Method

---

```
void SimpleCornerBalanceMethod(pRange& chunk) {  
    ...  
    pRange::iteratorType cell_it;  
    for (cell_it = chunk.get_boundary().start();  
        cell_it != chunk.get_boundary().finish();  
        ++cell_it) {  
        ... solve SCB balance equations ...  
    }  
    ...  
}
```

PolyhedralCell



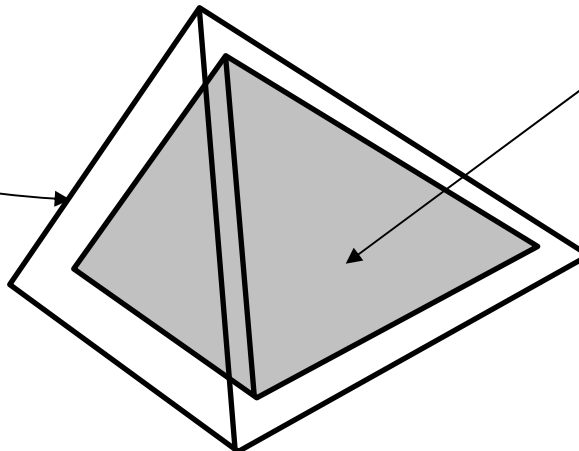
CornerElement

# Step Characteristic Method

---

```
void StepCharacteristicMethod(pRange& chunk) {  
    ...  
    pRange::iteratorType cell_it;  
    for (cell_it = chunk.get_boundary().start();  
         cell_it != chunk.get_boundary().finish();  
         ++cell_it) {  
        ... solve SC equation ...  
    }  
    ...  
}
```

PolyhedralCell



WholeCellElement

# How to Add New Sweep Method

---

```
driver(...) {  
    problem = read_problem();  
    problem->solve();  
}
```

```
BaseProblem* read_problem(...) {  
    Hex_cell_creator creator(*pinput);  
    asci_preprocessing::ASCII_Regular_pRange_Preprocessor*  
    sched = build_brick_scheduler(..., *pinput);  
    BaseProblem* bp = new  
    GenericSweepProblem<LogicallyRectangularGrid,  
    Wt_Diamond_Diff_Method>(pinput, creator, egs, sched,  
    output_stream, error_stream);  
    return bp;  
}
```

# Just Add It

---

```
driver(...) {  
    problem = read_problem();  
    problem->solve();  
}
```

```
BaseProblem* read_problem(...) {  
    Another_cell_creator creator(*pinput);  
    asci_preprocessing::ASCII_Regular_pRange_Preprocessor*  
    sched = build_brick_scheduler(..., *pinput);  
    BaseProblem* bp = new  
    GenericSweepProblem<LogicallyRectangularGrid,  
    Another_method>(pinput, creator, egs, sched,  
    output_stream, error_stream);  
    return bp;  
}
```

# Conclusion

---

- pRange → Cell → Element structure makes coding easy
  - Sweep methods don't care about outside
  - Others don't care about inside sweep methods
- I don't care how computation is parallelized
  - STAPL spoils me