Real Residence Time Distribution of Solids from Lagrangian Simulation in OpenFOAM®

Yujian Sun, Milorad P. Dudukovic

Chemical Reaction Engineering Laboratory
Department of Energy, Environmental & Chemical Engineering
Washington University in St. Louis
Outline

• Background and Motivation

• OpenFOAM Lagrangian Functionality

• Strategy and Implementation Details

• Preliminary Result and Future Work
Background

• CFB, solids back-mixing, solids RTD
• Conventional tracer method, open boundaries
• CARPT (Computer Automated Radioactive Particle Tracking)
Solids RTD from CARPT

Mean of RTD = 44.04 sec
Stdev of RTD = 44.4 sec
Total Occurrences = 1643

$U_g$ (riser) = 3.2 m/s
$G_s$ = 26.6 kg/m$^2$/s

Mean of RTD = 17 sec
Stdev of RTD = 42.3 sec
Total Occurrences = 1023

$U_g$ (riser) = 3.9 m/s
$G_s$ = 33.7 kg/m$^2$/s

Bhusarapu et al., 2004
Lagrangian Particle Tracking

- OpenFOAM, versatility & flexibility
- Lagrangian package: “clouds” and “parcels”
- Object oriented
- Inheritance and templates

```cpp
typedef CollidingCloud<
    KinematicCloud<
        Cloud<
            basicKinematicCollidingParcel
        >>
    basicKinematicCollidingCloud;

typedef CollidingParcel<
    KinematicParcel<
        particle
    >>
    basicKinematicCollidingParcel;
```
• Starting point
  KinematicParcel<ParcelType>::move(td, dt)
  while (td.keepParticle ...) { ...
  p.age() += dt; ...}

• Create a new class “RTDStatus” to be included in KinematicParcel, which calls the new class function to update RTD status
Challenges & Strategy
Implementation Details

class RTDStatus {
public:
    enum CFBRegion {
        riser,
        downer,
        riserToDowner,
        downerToRiser
    };

private:
    const polyMesh& mesh_;  
    const vector& position_; 
    const IOdictionary dict_;  
    const Switch active_;  
    scalar startTime_;  
    const vector riserBB1_; 
    const vector riserBB2_; 
    const vector downerBB1_; 
    const vector downerBB2_; 
    CFBRegion region_;  
    CFBRegion region0_;  
    bool fresh_;  
    bool recording_;  
    scalar tResT_;  
    scalar tResF_;  
    scalar firstEntrance_;  
    scalar latestExit_;  
    scalar finalExit_;
1. Check CFB Region

void Foam::RTDStatus::findRegion()
{
    if (inRiser()) region_ = riser;
    else {
        if (inDowner()) region_ = downer;
        else {
            if (position_.z() > riserBB1_.z())
                region_ = riserToDowner;
            else {
                if (position_.z() < riserBB2_.z())
                    region_ = downerToRiser;
                }}
    }}
2. Check if Recording Started

```cpp
if (region_ == downer) {
    fresh_ = true;
}

if (fresh_ && (region_ == riser)) {
    recording_ = true;
    fresh_ = false;
    tResT_ = 0;
    tResF_ = 0;
    firstEntrance_ = mesh_.db().time().value();
    if (firstEntrance_ < startTime_) {
        recording_ = false;
    }
}
```
3. Check if Recording Finished

if ((region0_ == riser) && (region_ == riserToDowner))
{
    latestExit_ = mesh_.db().time().value();
}

if (recording_ && (region_ == downer))
{
    recording_ = false;
    fresh_ = true;
    finalExit_ = latestExit_;
    tResF_ = finalExit_ - firstEntrance_;  
}
4. Update Residence Times

```c
if (recording_)
{
    if (region_ == riser)
    {
        tResT_ += dt;
    }
    tResF_ += dt;
}
region0_ = region_;```

Simple Test

• “RTDParticl” & “lagrangianRTD”
• Single particle cycles through two parts, oscillating at inlet and outlet of “riser”
# Preliminary Results

<table>
<thead>
<tr>
<th>time</th>
<th>position_x</th>
<th>position_y</th>
<th>position_z</th>
<th>recording</th>
<th>tResT</th>
<th>tResF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

• The algorithm works correctly, readily extensible to KinematicParcel
• Simulate CFB and reproduce CARPT data
Conclusions and Future Work

• The algorithm works correctly, readily extensible to KinematicParcel
• Simulate CFB and reproduce CARPT data
Thank you!