Local grid refinement in OPM

Antonella Ritorto

Dr. Blatt HPC-Simulation-Software & Services
antonella.ritorto@opm-op.com

DUNE User Meeting 2023

September 18-19
Dresden
Who are we?
Collaborative software developers

What do we do?
Modeling and simulation of porous media processes
  - CO₂ sequestration
  - Oil recovery

What do we provide?
Open-source software and open data sets
Tools to create/customize solvers and simulators:
  - **OPM Flow**
    Reservoir simulator for three-phase black-oil problems
  - **Upscaling**
    Flow-based upscaling of (relative) permeability and capillary curves
  - **ResInsight**
    Visualization tool for reservoir simulation
Modules

OPM core modules

- **opm-common**
  - build system
  - Eclipse deck parser, Eclipse binary format I/O

- **opm-grid**

- **opm-simulators**
  - OPM Flow
  - automatic differentiation library
  - solvers: IMPES, 2-phase incompressible transport, ...

DUNE core modules (⋆)

- dune-localfunctions
  - finite element spaces on single elements

- dune-grid
  - grids and the grid interface

- dune-geometry
  - grid elements and quadrature

- dune-common
  - common infrastructure

- dune-istl
  - sparse linear algebra

(⋆) from **DUNE-The distributed and Unified Numerics Environment**, Oliver Sander.
Corner-point Grid
**DUNE** module supporting

- **CpGrid** $\rightarrow$ corner-point grid
- **PolyhedralGrid** $\rightarrow$ fully unstructured grid

**CpGrid** defined by **vertical pillars** and **layers** *(might vanish/collapse!)*

```
RUNSPEC
DIMENS
1 1 5 /
GRID
COORD
--- Vertical pillar
 0 0 0
 0 0 1
--- Vertical pillar
 1 0 0
 1 0 1
--- Vertical pillar
 0 1 0
 0 1 1
--- Vertical pillar
 1 1 0
 1 1 1 /
ZCORN --- depth of each corner point
4*0 --- top layer 1
8*1 --- bottom layer 1
8*2 --- top layer 2
8*3 --- bottom layer 2
8*4 --- top layer 3
4*5 --- bottom layer 3 /
ACTNUM
5*1 --- all 5 cells active /
PORO
5*0.15
```
opm-grid

**DUNE** module supporting

- **CpGrid** → corner-point grid
- **PolyhedralGrid** → fully unstructured grid

**CpGrid** defined by *vertical pillars* and *layers* *(might vanish/collapse!)*

```
RUNSPEC
DIMENS
1 1 1 /
GRID
COORD
--- Vertical Pillar
0 0 0 0 0 1
--- Vertical Pillar
1 0 0 1 0 0
--- Vertical Pillar
0 1 0 0 1 1
--- Vertical Pillar
1 1 0 1 1 0
/
ZCORN
4*0 --- top layer
4*1 --- bottom layer
/
ACTNUM
1*1
```
(Regular) Cartesian grid example - SPE1

Reservoir model from the 1st Society of Petroleum Engineers Comparative Solution Project

Designed for benchmarking 3D black-oil simulators

- Three-layer permeability structure
- Injection well
- Production well

The reservoir is filled in with oil and water. Gas is injected through the injection well.

---

RUNSPEC
DIMENS
10 10 3 /

GRID
DX
-- There are in total 300 cells with length 1000ft in x-direction
300*1000 /

DY
-- There are in total 300 cells with length 1000ft in y-direction
300*1000 /

DZ
-- The layers are 20, 30 and 50 ft thick, in each layer there are 100 cells
100*20 100*30 100*50 /

TOPS
-- The depth of the top of each grid block
100*8325 /

---

OPM Flow manual, David Baxendale
(Irregular) Corner-point Grid example - Norne oil field

OPM Flow manual, David Baxendale

Antonella Ritorto  Local Grid Refinement in OPM
Corner point grid

Corner point grid ~⇒ degenerated and distorted Cartesian grid

Degenerated/distorted cells

It provides mappings from cells to the underlying Cartesian Index

Each cell can be ACTIVE or INACTIVE

Dune::CpGrid::createCartesian method

```cpp
void createCartesian(const std::array<int, 3>& dims,
const std::array<double, 3>& cellsize,
const std::array<int, 3>& shift = {0, 0, 0});
```

- dims ~⇒ number of cells in each Cartesian direction
- cellsize ~⇒ size of each cell in each direction
- shift ~⇒ origin of the grid
  ~⇒ corner of the cell with Cartesian Index (0, 0, 0)
Local Grid Refinement
Cooperation between

TNO

SINTEF

Dr. Blatt - HPC-Simulation-Software & Services
LGR project is a cooperation between

TNO

Bård Skaflestad
SINTEF

Dr. Blatt-
HPC-Simulation-
Software
& Services
How it was one year ago

**LGR ACTIVITIES**

- Literature and simulators reviewed: Eclipse, CMG, MRST, papers...
- Tests cases created and pushed to opm-tests
- Designing plan for code implementation to support the following aspects of LGR functionality (in cooperation with SINTEF and OPM-OP AS)
  - Requires code changes in several repositories: opm-common, opm-grid and opm-simulators
- Challenges:
  - Allow wells to cross over grids (means solving the all)
  - Output and restart of models with LGR
  - Extensive code refactoring in opm-grid needed

**Changes to opm-grid:**
  - Store multiple level grids in CpGrid
  - Identify Mother-Child relationships
  - Store/Compute/Iterate over leaf grid view

Cintia Machado et al. (TNO) – Recent OPM Devel

```cpp
std::vector<Geometry<3, cdim>> refine(const std::array<int, 3>& cells_per_dim,
           std::vector<EntityVariable<Geometry<0, 3>, 3>>& corner_storage,
           std::vector<std::array<int, 8>>& indices_storage)
```
What has been done in opm-grid

Using DUNE’s adaptive grid interface (⋆)

- Create and store multiple level grids in CpGrid
- Identify Parent-Child relationships
- Store/compute/iterate over leaf grid view

(⋆) from DUNE-The distributed and Unified Numerics Environment, Oliver Sander.
What has been done in opm-grid

Using DUNE’s adaptive grid interface

- Create and store multiple level grids in CpGrid ✓
- Identify Parent-Child relationships ✓
- Store/compute/iterate over leaf grid view ✓

How?

(*) from DUNE-The distributed and Unified Numerics Environment, Oliver Sander.
What has been done in opm-grid - How?

Using DUNE’s adaptive grid interface(

- Hierarchical Grids (refinement; geometrical and topological features)
- Entity::father(), hasFather(), geometryInFather(), ...
- Hierarchical Grids and Leaf Grid View (iterators, ...)
Hierarchical grids and grid views in OPM - Now!

DUNE(⋆)

Level 2
Level 1
Level 0
Level grid views

Leaf grid view

OPM - Now!

Level 3
Level 2
Level 1
Level 0
Level grid views

Leaf grid view

(⋆) from DUNE-The distributed and Unified Numerics Environment, Oliver Sander.
Technical (temporary!) limitations

1. **Corner point / Cartesian grid** → cells with 8 corners

Non (yet) supported cells:

2. All cells are **ACTIVE**
Local grid refinement in OPM

Refine single cell

Refine block-shaped patch
(*) from DUNE-The distributed and Unified Numerics Environment, Oliver Sander.
Grid elements with descendants

(*) from **DUNE-The distributed and Unified Numerics Environment**, Oliver Sander.
When do LGRs appear in the simulation?

OPM

LGRs in (oil) simulation
- are defined at the beginning of the simulation
- can be switched on and off during the simulation

DUNE

Decide which cells should be - refined - coarsened during simulation
After that, simulation continues
Refine/coarsen takes place again
Wrapping up

<table>
<thead>
<tr>
<th>One year ago...</th>
<th>Now!</th>
<th>What’s next?</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Level 0 = Leaf Grid View</td>
<td>• Multiple level grids</td>
<td>• Mark subset of leaf grid elements</td>
</tr>
<tr>
<td>• <em>Incomplete</em> Geometry::refine single cell</td>
<td>• Parent-Child relations</td>
<td>- to refine</td>
</tr>
<tr>
<td></td>
<td>• Update Leaf Grid View</td>
<td>- to coarsen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Call</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- adapt()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- preAdapt()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- postAdapt()</td>
</tr>
</tbody>
</table>

Further...

• Remove *temporary* constrains:
  - allow **INACTIVE** elements in the LGRs
  - allow **elements with fewer corners** (8 currently)
Thank you for your attention!

Dr. Blatt HPC-Simulation-Software & Services

Check our GitHub repository OPM/opm-grid!