OPM
OPEN POROUS MEDIA

Local Grid Refinement in OPM Flow:
today and beyond

Artur Castiel · Antonella Ritorto

OPM Summit · April 2026

TNO innovation for life

OPM-OP

Local grid refinement in OPM Flow: today and beyond

Acknowledgements

- **OPM-OP**
  - Blatt, Markus
  - Ritorto, Antonella
- **SINTEF**
  - Skaflestad, Bard
- **TNO**
  - Castiel, Artur
  - Barros, Eduardo
  - Khoshnevis, Negar

OPM LGR
SUPER TEAM!

Local grid refinement in OPM Flow: today and beyond

Overview

Agenda

Part 1 — LGR Output Pipeline in opm-common

Artur Castiel

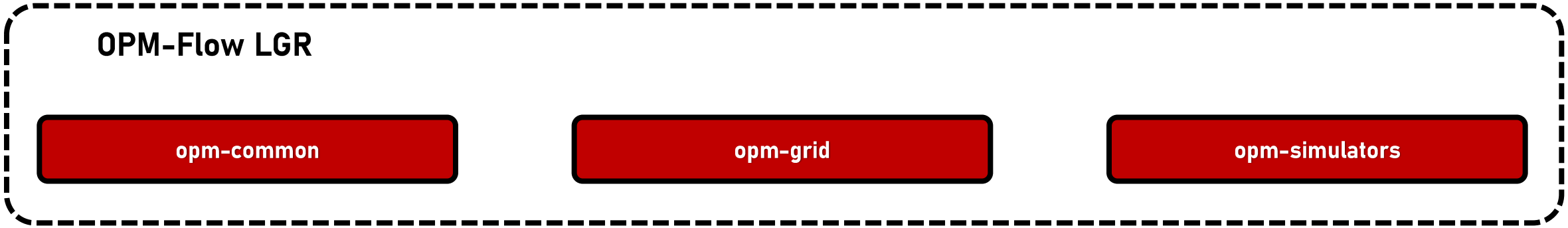
1. Shared infrastructure: grid model and well data
2. EGRID output: COORD, ZCORN, NNC topology
3. INIT output: headers, pore volumes, transmissibilities
4. UNRST output: well/group dynamic data and restart
5. NNC evolution: data structures & cross-format output

Part 2 — LGR in OPM Flow: Simulation

Antonella Ritorto

1. Parallel run, limited output
2. Restart values for level grids (serial)
3. TRAN* and NNCs for level grids (serial)
4. Nested refinement
5. Faults

Which parts of OPM need to be modified to support LGR ?



Part 1

LGR in OPM Flow: opm-common

Artur Castiel · TNO / opm-common

Local grid refinement in OPM Flow: today and beyond

Context

Three output files — one refined grid hierarchy

LGR nests child grids inside the global grid. Every Eclipse output format must understand this parent–child relationship and write a dedicated sub-section per LGR label.

EGRID

Grid Geometry

- COORD + ZCORN per refined block
- NNC topology: NNCG, NNCL
- NNA blocks for cross-LGR connections

INIT

Static Properties

- LGR headers: LGRHEADI / LGRHEADD
- Pore volumes + permeabilities
- Transmissibilities: TRANNNC, TRANGL, TRANLL

UNRST

Dynamic / Restart

- Well arrays: IWEL, ICON, LGWell_
- Group data: IGRP
- Dynamic rates, BHP, LGR restart headers

Each format required dedicated data structures, new write routines and integration tests — built over 18 months.

Local grid refinement in OPM Flow: today and beyond

Foundation

Shared infrastructure underpinning all three formats

Before any output format can be extended, OPM needs a C++ model of what an LGR is. These building blocks are shared by every write routine.

#4255 / #4393 — DATA STRUCT.

EclipseGridLGR

- Derived from EclipseGrid
- CARFIN: stores nx/ny/nz refinement params
- create_lgr_cells_tree() — parent-child map
- VectorUtil / GeometricUtil helpers

#4451 / #4466 / #4478 — DATA STRUCT.

Localization & Grid Utilities

- Arithmetic algorithm: places LGR zones
- pillar_to_flat_array(), convertUnsToCPG()
- Keyword parser for CARFIN blocks
- Maps global cell → LGR label

#4489

WELSPECL — LGR Well Specification

- New WELSPECL keyword support
- Associates wells with named LGR labels
- insert_index_lgr added to Well class
- Foundation for all LGR well output

#4491

ScheduleGrid LGR Extension

- ScheduleGrid extended for LGR wells
- filter_wells_for_lgr() utility function
- Enables per-LGR well aggregation
- Used by both INIT and UNRST writers

These structures are shared across EGRID, INIT and UNRST — every write routine builds on this common foundation.

Local grid refinement in OPM Flow: today and beyond

EGRID Output

Geometry — pillar coords, depth values, NNC topology

Keywords written → `COORD` `ZCORN` `NNCHEAD` `NNC1` `NNC2` `NNCG` `NNCL` `NNCHEADA` `NNA1` `NNA2`

#4255 / #4393 — DATA STRUCT.

EclipseGridLGR + first EGRID output

- EclipseGridLGR appends LGR section to EGRID
- CARFIN stores nx/ny/nz subdivision params
- Point-in-polyhedron → parent cell mapping
- EGRID section written per LGR label

#4839 / #4985

COORD — Pillar Geometry

- generate_refined_coord() — linear interpolation
- Refined pillar positions from parent pillars
- Refactored in #4985: reduced allocations
- Per-column pillar subdivision

#4843 / #4845

ZCORN + save() refactor

- generate_refined_zcorn() — depth values
- Handles variable z-layer sizes (CARFIN)
- EclipseGrid::save / EclipseGridLGR::save
- Clean save path for global + LGR grids

#5049 / #5060 — → also INIT

NNCDiffGrid → NNCG / NNCL / NNA

- NNCDiffGrid class — cross-grid connections
- Preserves insertion order, swap_adj()
- NNCG: global↔LGR | NNCL: LGR↔LGR
- NNA blocks: NNCHEADA, NNA1, NNA2

EGRID now carries full CPG geometry (COORD + ZCORN) and NNC topology (NNCG, NNCL, NNA1/2) for every LGR block.

Local grid refinement in OPM Flow: today and beyond

INIT Output

Static properties — LGR headers, pore volumes, transmissibilities

Keywords written → `LGRHEADI` `LGRHEADD` `TRANNNC` `TRANGL` `TRANLL` `LGRJOIN` `LGRSGONE`

#4566

EclipseIO extension — first INIT output

- LGRHEADI / LGRHEADD header structs
- writePoreVolumeLGRCell()
- writeGridGeometryLGRCell()
- get_print_order_lgr(), filterArray()

#5028

LGR transmissibilities in writeInitial

- EclipseIO::writeInitial() overloaded
- Accepts std::vector<data::Solution>
- LGR-specific TRANX/Y/Z per grid block
- First step of the NNC pipeline

#5037 DATA STRUCT.

NNCCollection

- Container: same-grid + cross-grid NNC maps
- hasNNCForGrid(), empty() query methods
- Stores full NNC objects per grid scope
- Shared by EGRID writer and INIT writer

#5077 / #5088 / #5089 → NNC slide

TRANNNC · TRANGL · TRANLL

- writeLGRTranNNC / TranGL / TranLL()
- NNCDataContainer — same-grid flat vector
- NNCDataContainerDiffGrid — cross-grid
- fromLGROutputContainers() — sim bridge

INIT carries LGR headers, pore volumes, and all three NNC transmissibility arrays — readable by ResInsight.

Local grid refinement in OPM Flow: today and beyond

UNRST Output

Dynamic data & restart — well/group vectors for LGR simulations

Keywords written → `IWEL` `ICON` `IGRP` `LGWell_` `LGRHEADI` `LGRHEADQ` `LGRHEADD`

#4667

Well & group data aggregation

- AggregateWellData: captureDeclaredWellDataLGR()
- AggregateConnectionData extended for LGR
- LGWell_ — LGR well reference array
- IWEL enhanced with LGRIndex field

#4694

Well dynamic data

- captureDynamicWellDataLGR() added
- Rates, BHP, pressures per LGR well
- Tested: producers + injectors in LGR
- Bugfixed in #4809 (WGIT/WGIR), #4915

#4717

Output Restart support

- RestartValue: new LGR data constructor
- Connection struct: lgr_grid index field
- createLgrHeadi/q/d() header writers
- hasArray(name, number, gridname) overload

#4799 / #4814 / #4827

all 3 formats

Group data + integration test

- AggregateGroupData extended for LGR
- writeGroupLGR() reactivated post-refactor
- Round-trip test: EGRID / INIT / UNRST
- UNRST LGR vectors written only if well present

UNRST carries complete well/group dynamic data for LGR-resident wells — LGR simulations are fully restartable.

Local grid refinement in OPM Flow: today and beyond

NNC Evolution

Non-Neighbor Connections — spans EGRID and INIT (6 PRs)

NNCs connect cells that don't share a face. Three types exist for LGR: same-grid, local↔global, and cross-LGR. Each required its own data structure and write routine.

DATA STRUCT.

#5037

NNCCollection

Container:
same-grid +
cross-grid maps

→ EGRID

#5049

NNCDiffGrid

Cross-grid class
EGRID: NNCG+NNCL

→ EGRID

#5060

NNA Blocks

save_nna():
NNCHEADA
NNA1/NNA2

→ INIT

#5077

TRAN* Output

TRANNNC
TRANGL
TRANLL → INIT

DATA STRUCT.

#5088

NNCDataContainers

Lightweight
runtime
containers

→ INIT

#5089

Simulator Link

fromLGROutput
Containers()
opm-simulators

NNC Type	Keywords in EGRID	Keywords in INIT	C++ Class
Same-grid (local-local)	NNC1, NNC2, NNCHEAD	TRANNNC	NNCDataContainer
Local ↔ Global	NNCG, NNCHEAD	TRANGL	NNCDiffGrid
Cross-LGR	NNA1, NNA2, NNCHEADA	TRANLL, LGRJOIN	NNCDataContainerDiffGrid

Local grid refinement in OPM Flow: today and beyond

Data Structures

Six new C++ classes introduced for LGR output

Coloured badges show which output formats each class serves. Classes marked DATA STRUCT. are shared building blocks with no direct file output.

EGRID

`EclipseGridLGR`

Derived from EclipseGrid. CARFIN params, parent↔child maps, generate_refined_coord/zcorn(), create_lgr_cells_tree().

EGRID INIT

`NNCCollection`

Two maps: same-grid and cross-grid NNC objects. hasNNCForGrid(), empty(). Foundation for all NNC output.

EGRID INIT

`NNCDiffGrid`

Derived from NNC. Cross-grid connections, insertion-order preserved. swap_adj(), sort(). Writes NNCG/NNCL to EGRID.

INIT

`NNCDataContainer`

Flat vector of same-grid NNCs, cell1 ≤ cell2 ordering. Lightweight runtime type — feeds writeLGRTranNNC().

INIT

`NNCDataContainerDiffGrid`

Derived from NNCDataContainer. No ordering constraint. Feeds writeLGRTranGL() and writeLGRTranLL().

UNRST

`AggregateWellData / AggregateGroupData_LGR`

captureDeclaredWellDataLGR(), captureDynamicWellDataLGR(). Writes IWEL, ICON, IGRP, LGWell_ to UNRST.

6 new C++ classes — each purpose-built and tested — are the backbone of LGR output in opm-common.

Local grid refinement in OPM Flow: today and beyond

Part 1 — Summary

opm-common LGR output pipeline: what was built

EGRID

COORD + ZCORN + NNCG/NNCL + NNA blocks — full CPG geometry and NNC topology per LGR block

INIT

LGRHEADI/D + pore volumes + TRANNNC / TRANGL / TRANLL — all static LGR properties

UNRST

IWEL + ICON + IGRP + dynamic well/group data — LGR simulations are fully restartable

Data Structures

6 new C++ classes: EclipseGridLGR, NNCCollection, NNCDiffGrid, NNCDataContainer*

Test Coverage

End-to-end integration test (#4827) validates round-trip across all three formats

Scope

34 merged PRs · 18 months · OPM/opm-common (author: arturcastiel)

Part 2

LGR in OPM Flow: Grid/Simulation

Antonella Ritorto · OPM-OP

Where we were, where we are

	Last OPM Summit		LGR today	
Simulation	Serial	Parallel	Serial	Parallel
–parsing-strictness=**low** –enable-**vtk-output**=**true**	✓	✗		
–parsing-strictness=**low** –enable-**ecl-output**=…	**false**	✗		
WELSPEC**L** COMPDAT**L**	Bug in **strict** local refinement	✗		

antonella.ritorto@opm-op.com

Where we were, where we are

	Last OPM Summit		LGR today	
Simulation	Serial	Parallel	Serial	Parallel
–parsing-strictness=**low** –enable-**vtk-output**=**true**	✓	✗	✓	
–parsing-strictness=**low** –enable-**ecl-output**=…	**false**	✗		
WELSPEC**L** COMPDAT**L**	Bug in **strict** local refinement	✗		

antonella.ritorto@opm-op.com

Where we were, where we are

	Last OPM Summit		LGR today	
Simulation	Serial	Parallel	Serial	Parallel
–parsing-strictness=**low** –enable-**vtk-output**=**true**	✓	✗	✓	✓
–parsing-strictness=**low** –enable-**ecl-output**=…	**false**	✗	true	✗
WELSPEC**L** COMPDAT**L**	Bug in **strict** local refinement	✗	✓	✗

antonella.ritorto@opm-op.com

LGR today

- **Parallel run**

  Limited output

  –enable-**vtk-output**=**true**
  –enable-**ecl-output**=**false**

- **Serial run**

  –enable-**ecl-output**=**true**

  — **Restart Values** for **level grids** → **EGRID** file

  — **TRANX/Y/Z** and **NNCs** for **level grids** → **EGRID, INIT** files

antonella.ritorto@opm-op.com

LGR today

- **Parallel run**

  Limited output

  –enable-**vtk-output=true**
  –enable-**ecl-output=false**

- **Serial run**

  –enable-**ecl-output=true**

  — **Restart Values** for **level grids** → **EGRID** file *Artur's work, covered*

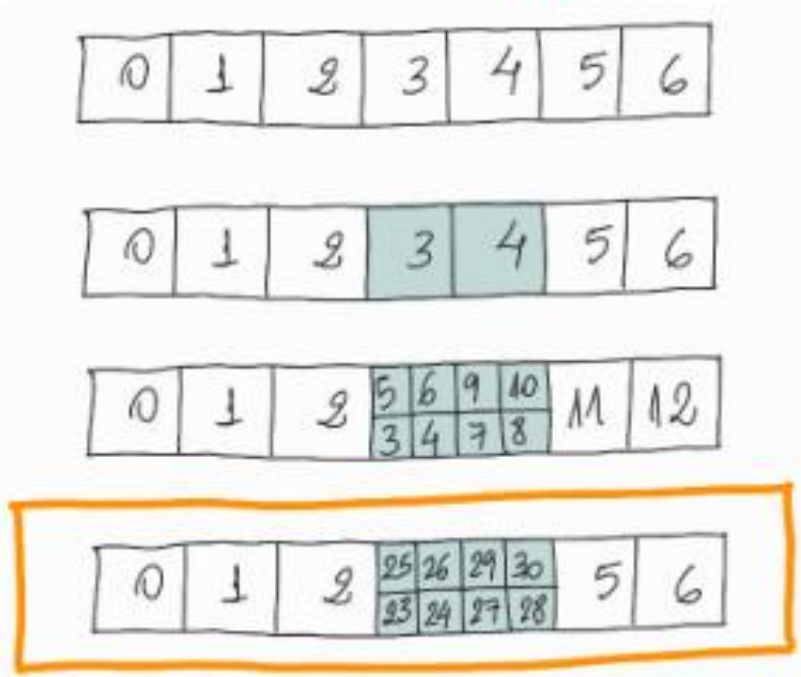
  — **TRANX/Y/Z** and **NNCs** for **level grids** → **EGRID, INIT** files *Artur's work, covered*

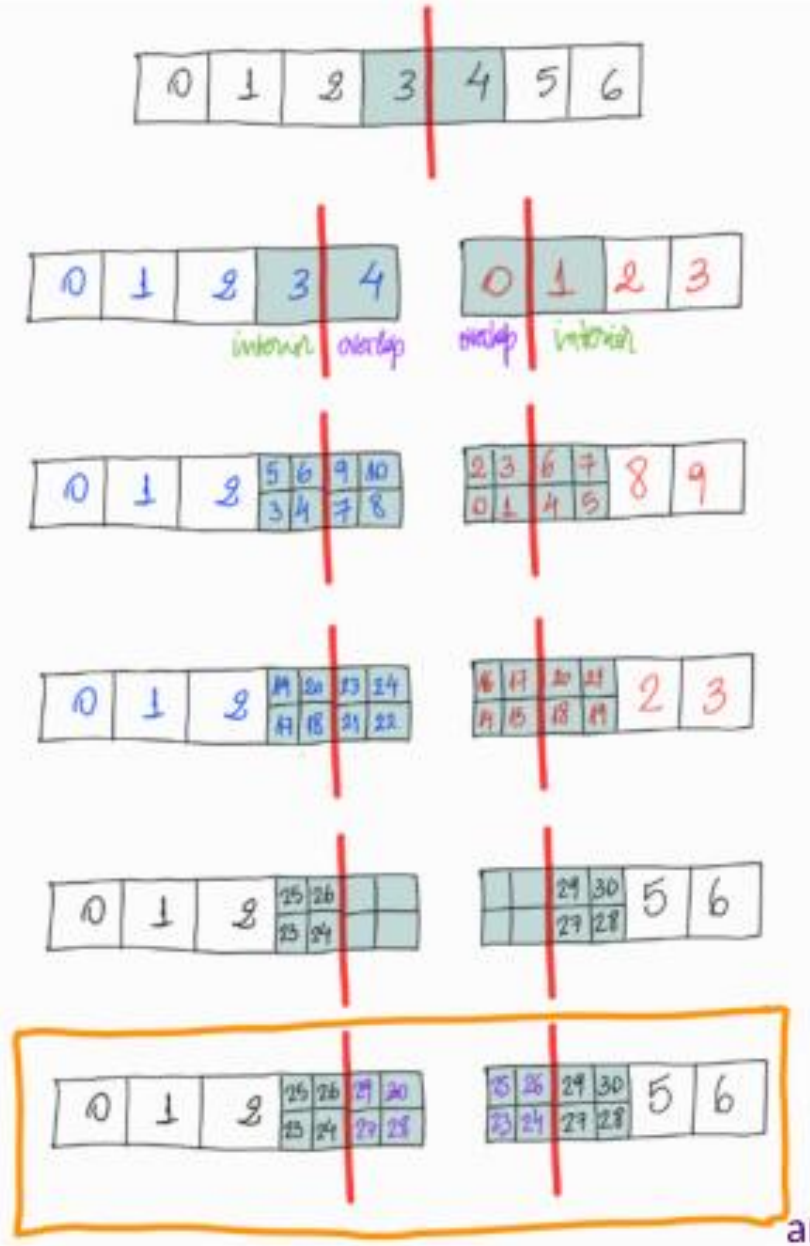
antonella.ritorto@opm-op.com

Parallel run, limited output

mpirun -np NUMBER ../**flow**
–parsing-stritcness=**low**
–enable-**vtk-output**=**true**
–enable-**ecl-output**=**false**
CASE.DATA

Local indices vs. **Ids**



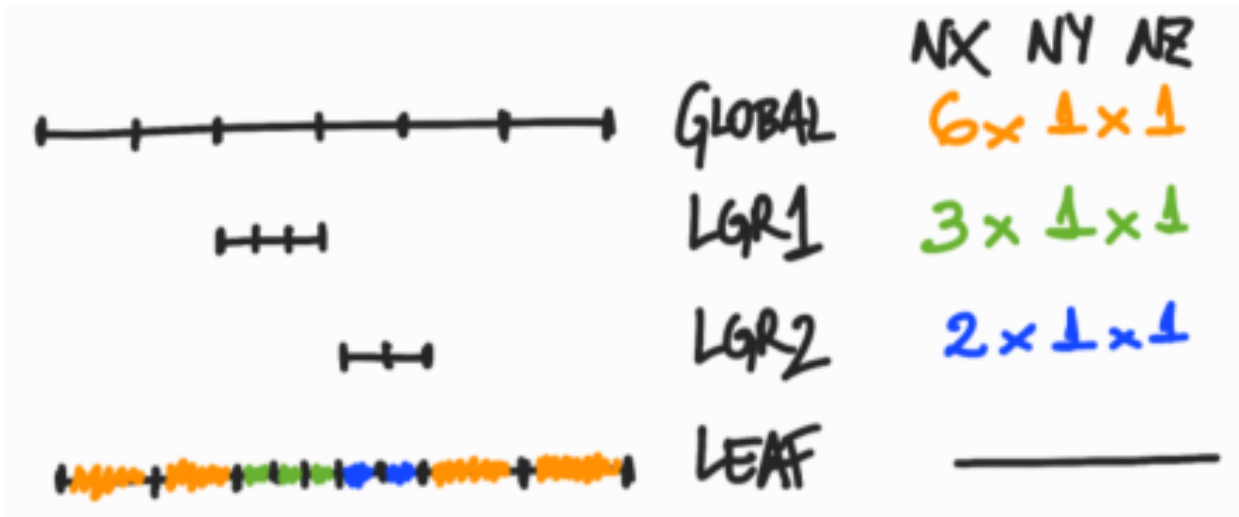


antonella.ritorto@opm-op.com

Restart Values for level grids (serial)

leaf data::Solution

Cartesian Indices

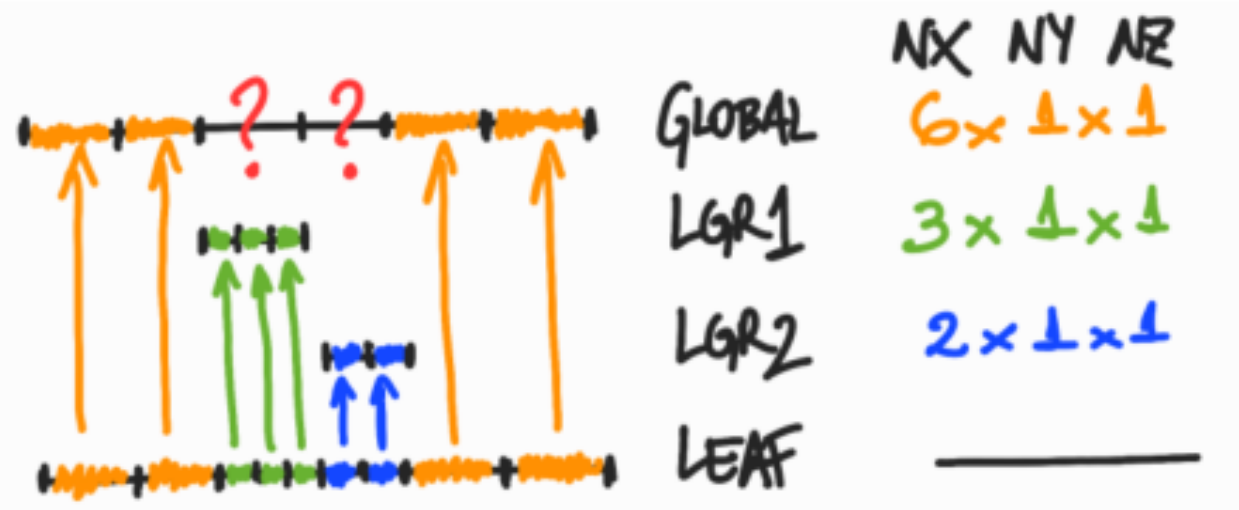


antonella.ritorto@opm-op.com

Restart Values for level grids (serial)

- Restrict **leaf data::Solution** to level grids
- Reorder containers based on **Level Cartesian Indices**
- Assign **average/max** **output values to parent cells**
  (!) Not visualized in ResInsight
- ../**flow** –parsing-stritcness=**low** –enable-**ecl-output**=**true** CASE.DATA

  EGRID file generated :)
  Artur's work, covered

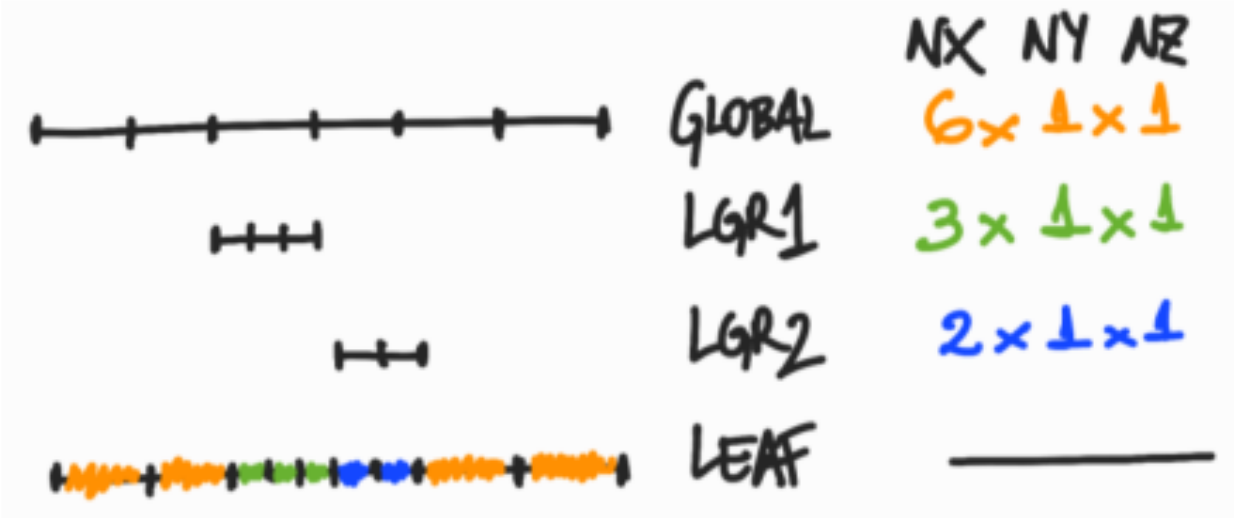


antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

leaf TRAN*

Cartesian Indices

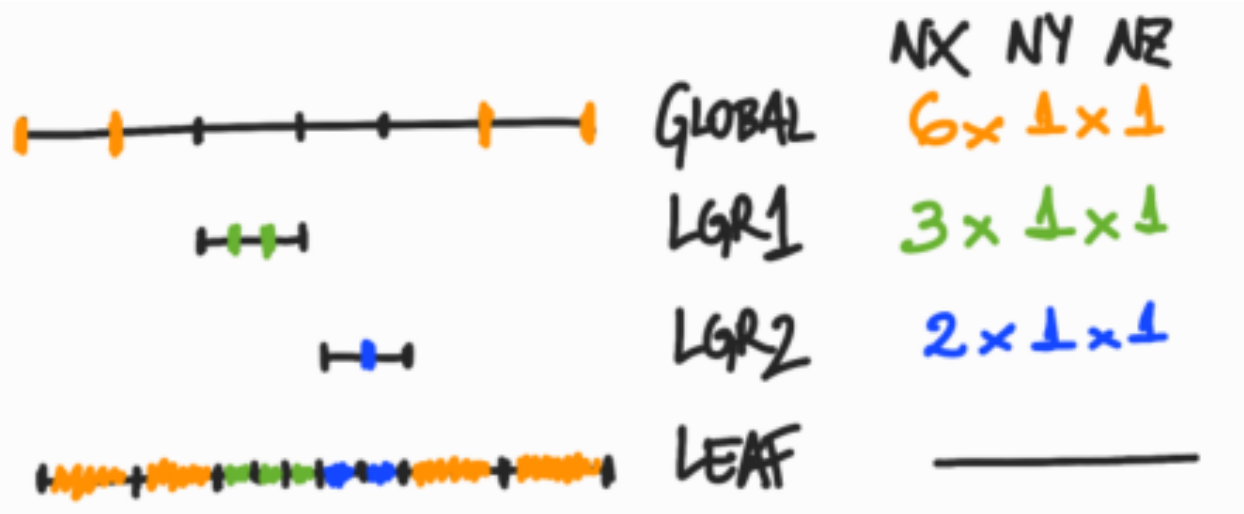


antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

- Restrict **leaf TRAN*** to level grids
- Containers based on **Level Cartesian Indices**
- Per level grid, **TRANX**, **TRANY**, **TRANZ**

(!) potential NNCs

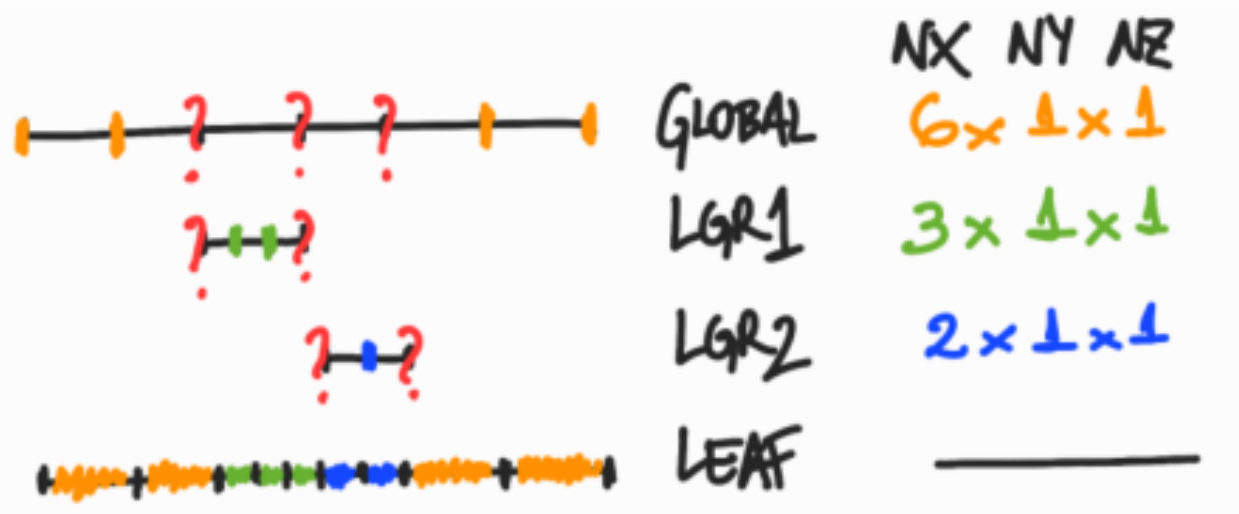


antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

- Restrict **leaf TRAN*** to level grids
- Containers based on **Level Cartesian Indices**
- Per level grid, **TRANX**, **TRANY**, **TRANZ**

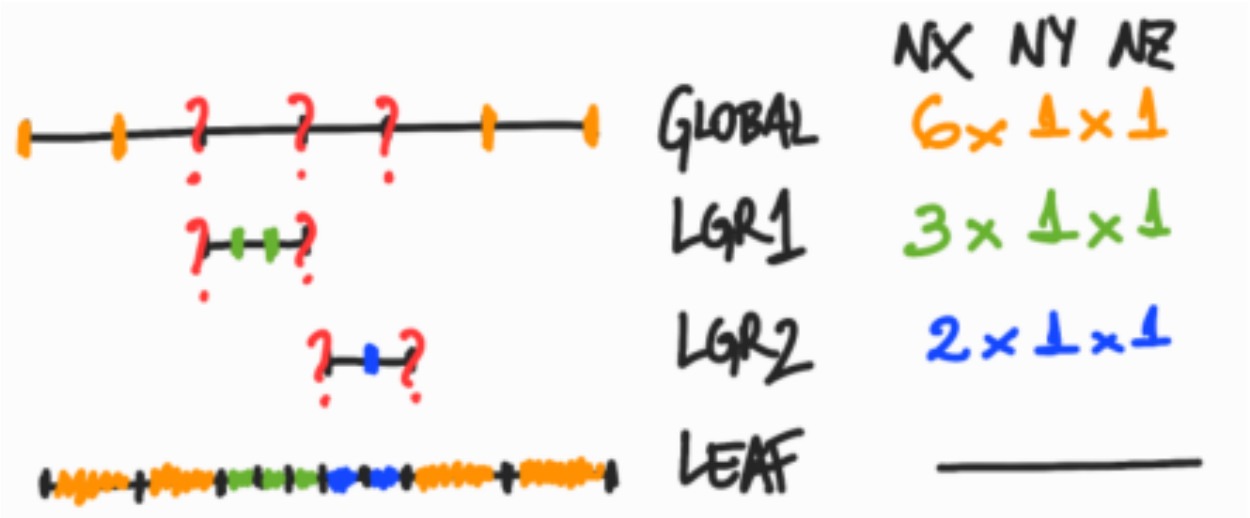
(!) potential NNCs



antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

NNC type	In .EGRID	In .INIT
Regular	NNC**1**, NNC**2**	TRAN**NNC**
Global to Local	NNC**G**, NNC**L**	TRAN**GL**
Amalgamated	NN**A1**, NN**A2**	TRAN**LL**



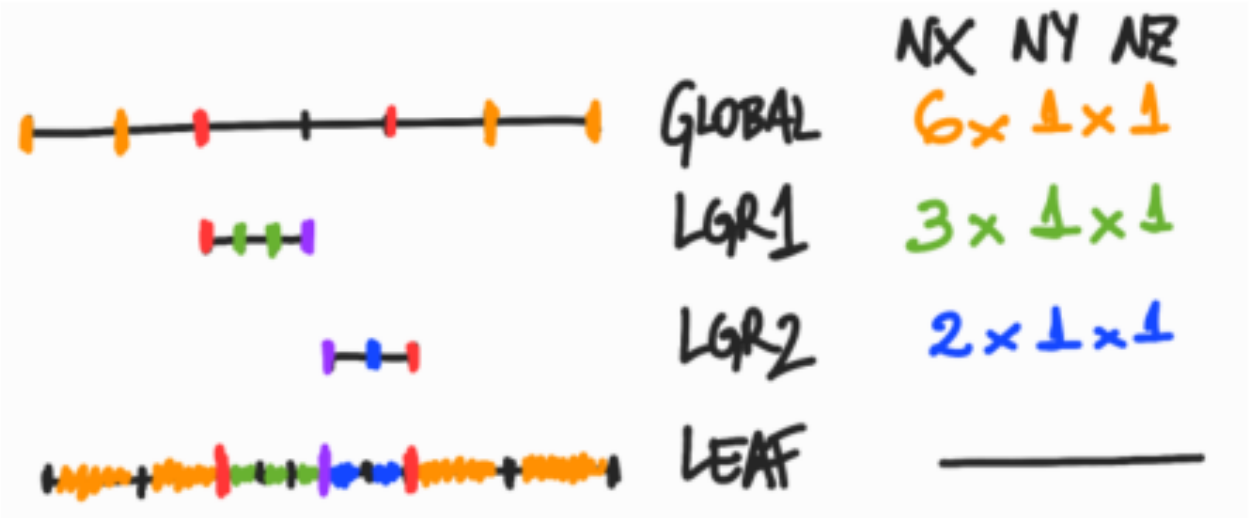
EGRID, INIT : *more details covered by Artur*

antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

NNC type	In .EGRID	In .INIT
Regular	NNC**1**, NNC**2**	TRAN**NNC**
Global to Local	NNC**G**, NNC**L**	TRAN**GL**
Amalgamated	NN**A1**, NN**A2**	TRAN**LL**

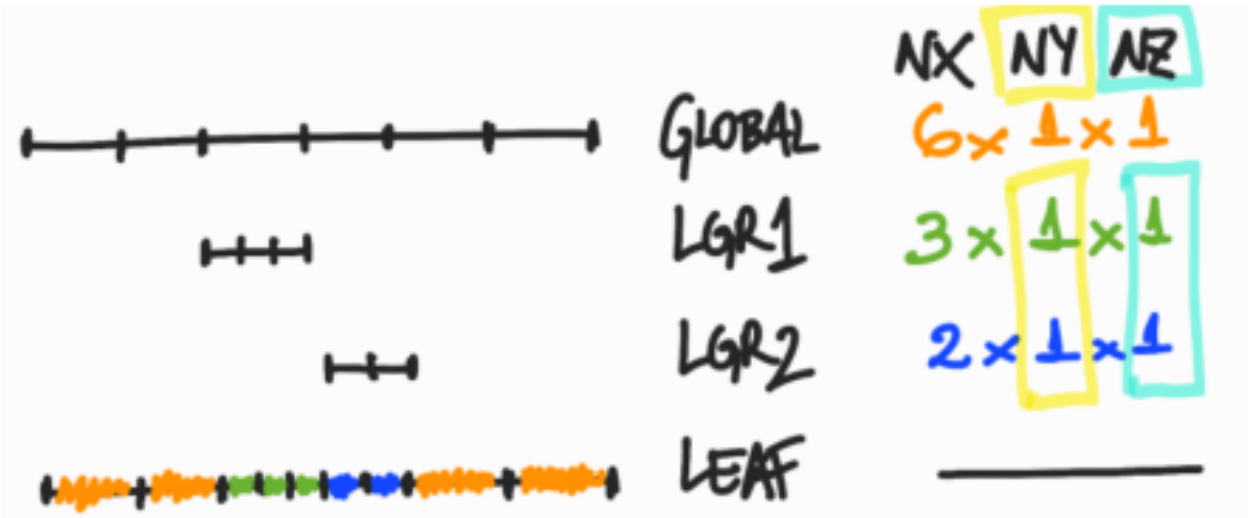
EGRID, INIT : *more details covered by Artur*



antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

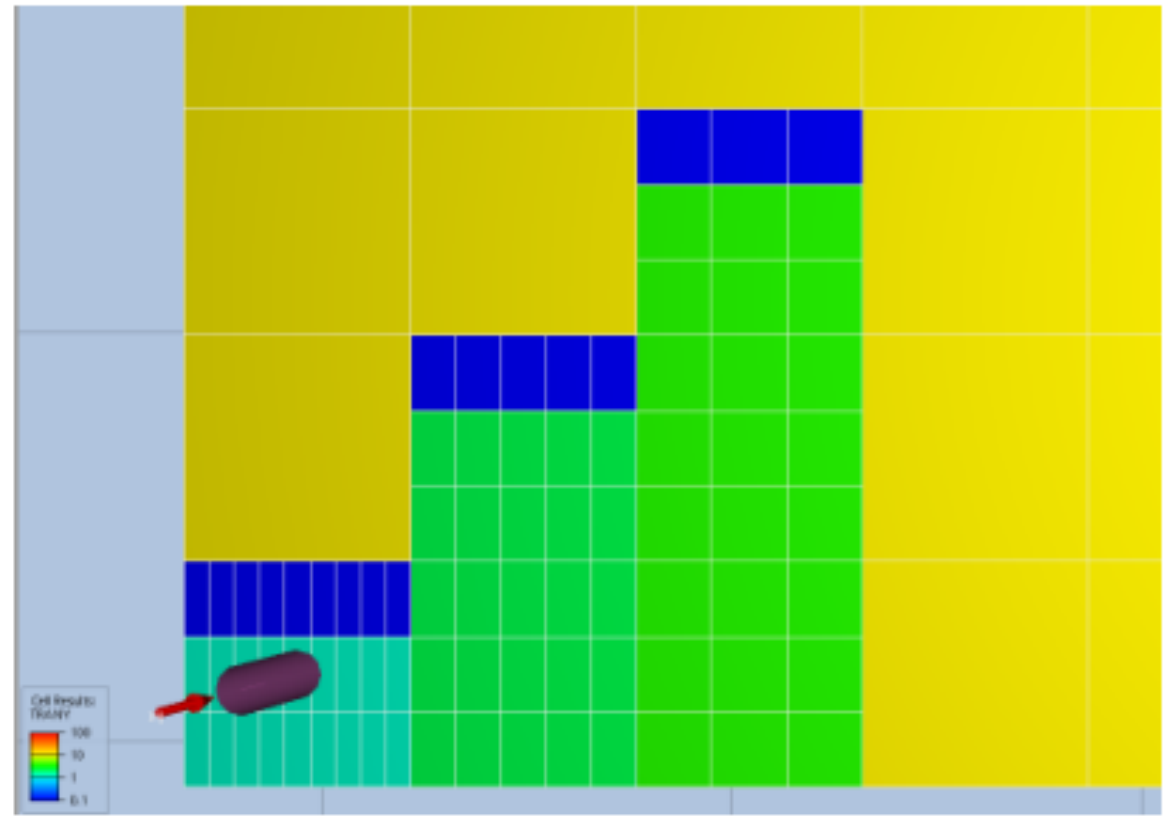
NNC type	In .EGRID	In .INIT
Regular	NNC1, NNC2	TRAN**NNC**
Global to Local	NNC**G**, NNC**L**	TRAN**GL**
Amalgamated	NN**A1**, NN**A2**	TRAN**LL**



antonella.ritorto@opm-op.com

TRAN* and NNCs for level grids (serial)

NNC type	In .EGRID	In .INIT
Regular	NNC**1**, NNC**2**	TRAN**NNC**
Global to Local	NNC**G**, NNC**L**	TRAN**GL**
Amalgamated	NN**A1**, NN**A2**	TRAN**LL**



antonella.ritorto@opm-op.com

LGR beyond today

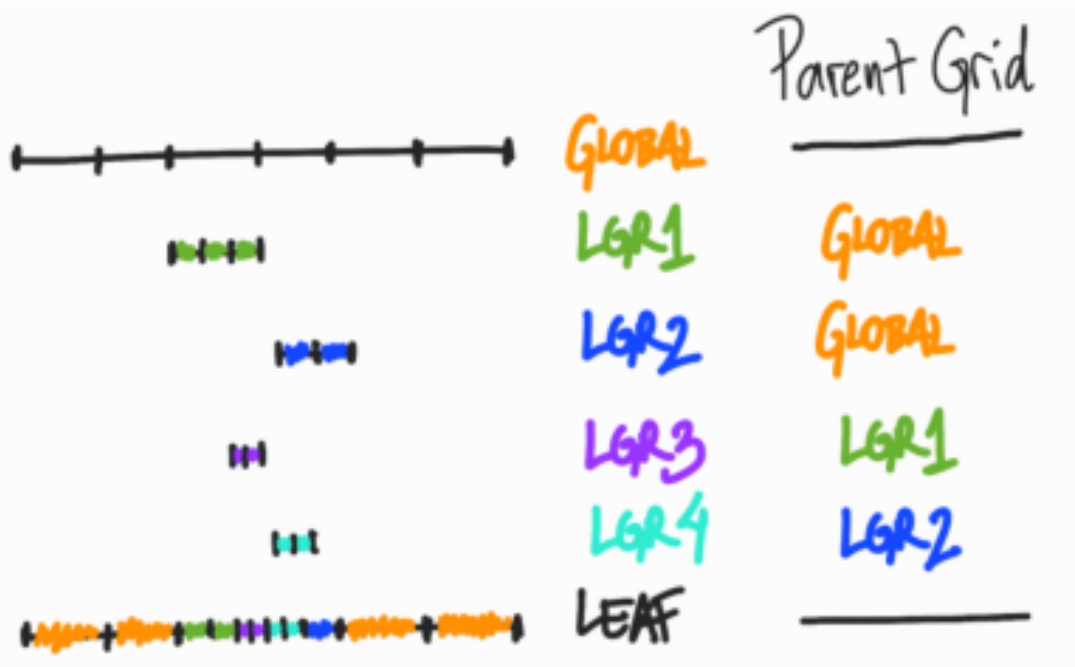
Not yet available

- **Parallel** run with **output** files
- **Nested refinement**
  - Provide **parent grid name** in **CARFIN** section
- **Faults**

antonella.ritorto@opm-op.com

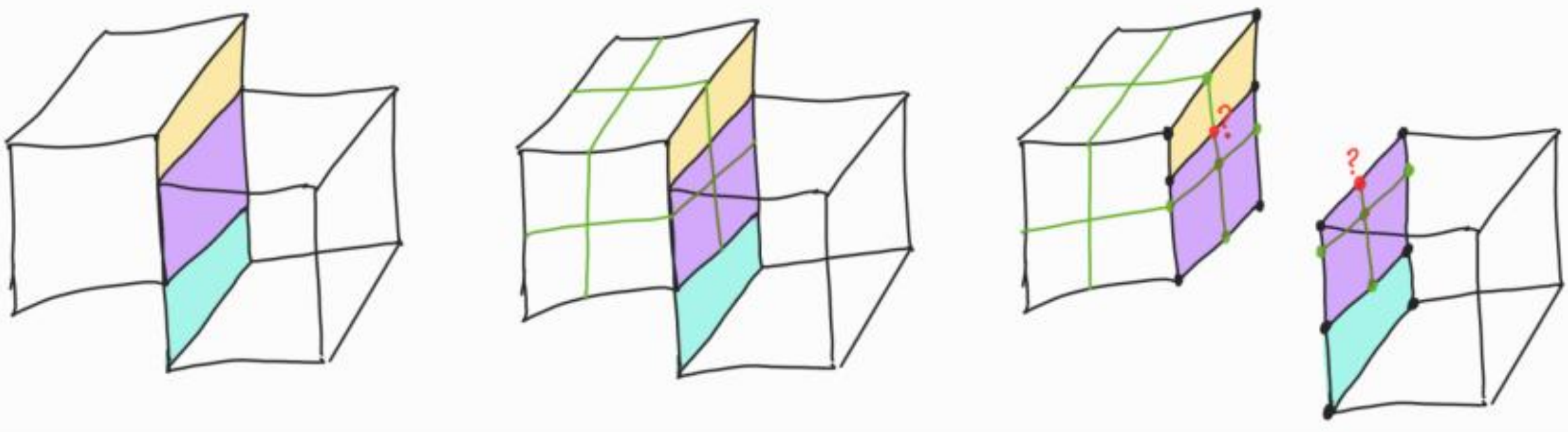
Nested refinement

```
CARFIN
-- NAME I1-I2 J1-J2 K1-K2   NX NY NZ
'LGR1'   3  3  1  1  1  1    3  1  1  1* 'GLOBAL'/
'LGR2'   4  4  1  1  1  1    2  1  1  1* 'GLOBAL'/
ENDFIN

CARFIN
-- NAME I1-I2 J1-J2 K1-K2  NX NY NZ
'LGR3'   3  3  1  1  1  1   2  1  1   1* 'LGR1'/
ENDFIN

CARFIN
-- NAME I1-I2 J1-J2 K1-K2  NX NY NZ
'LGR4'   1  1  1  1  1  1   2  1  1   1* 'LGR2'/
ENDFIN
```



antonella.ritorto@opm-op.com

Faults



antonella.ritorto@opm-op.com

Details in PR links & extras

Topic	opm-grid	opm-simulators	opm-test
Parallel run, limited output	–	PR6330	–
(Serial) Restart Values for level grids	PR925, PR945, PR955	PR6604, PR6702	–
(Serial) TRAN* and NNCs for level grids	–	PR6799, PR6844, PR6921, PR6972	–
Nested refinement	PR921, PR922, PR1018	PR6657	PR1505
Extras			
CARFIN + WELSPECL, COMPDATL	–	PR6304	PR1343, PR1458, PR1459
Aquifer cells/connections in LGRs	PR1008	PR6941	–

antonella.ritorto@opm-op.com

What users can do NOW and what SOON?

Simulation	Serial	Parallel
–parsing-strictness=**low** –enable-**vtk-output**=**true**	✓	✓
–parsing-strictness=**low** –enable-**ecl-output**=...	true	?
WELSPEC**L** COMPDAT**L**	✓	?

antonella.ritorto@opm-op.com

Final

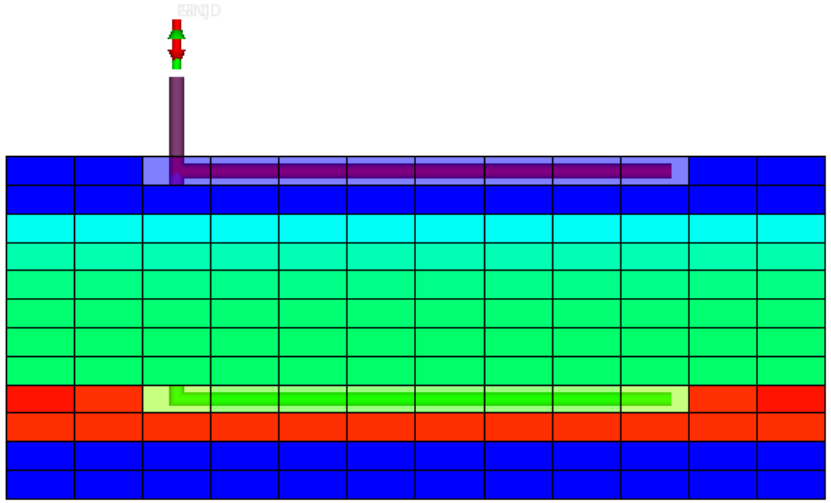
LGR in OPM Flow: Results

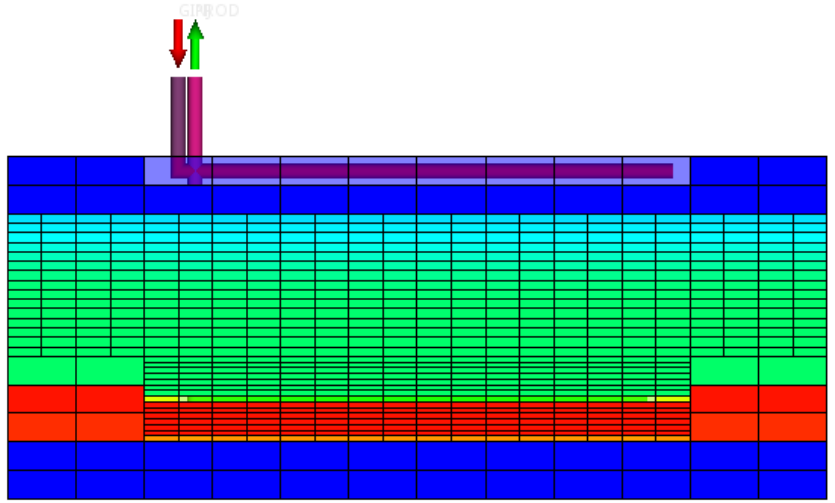
Artur Castiel · TNO / Antonella Ritorto · OPM-OP

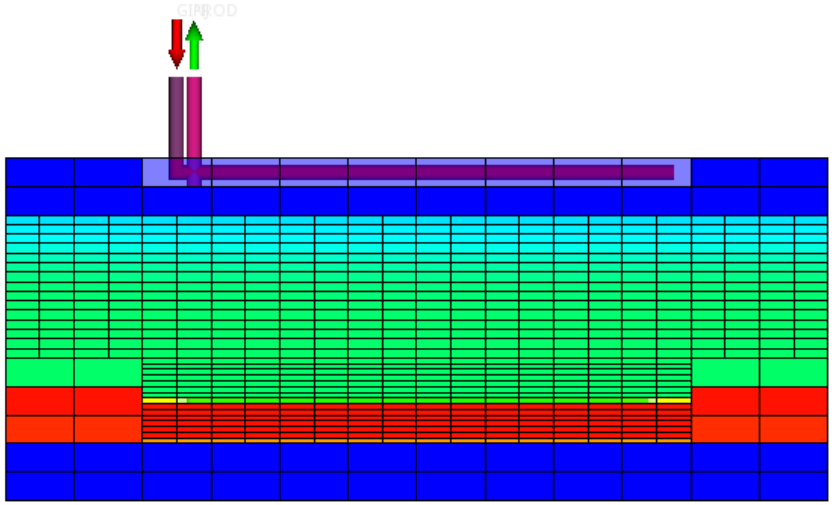
Local grid refinement in OPM Flow: today and beyond

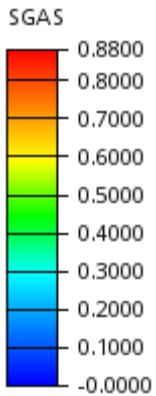
LGR Horizontal Well Injection

Oil Saturation









Reference Simulator
Coarse

Reference Simulator
LGR

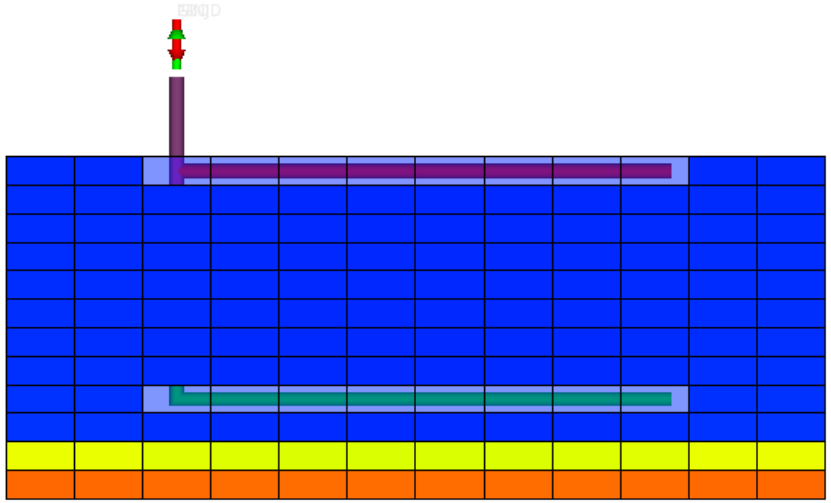
OPM Simulator
LGR

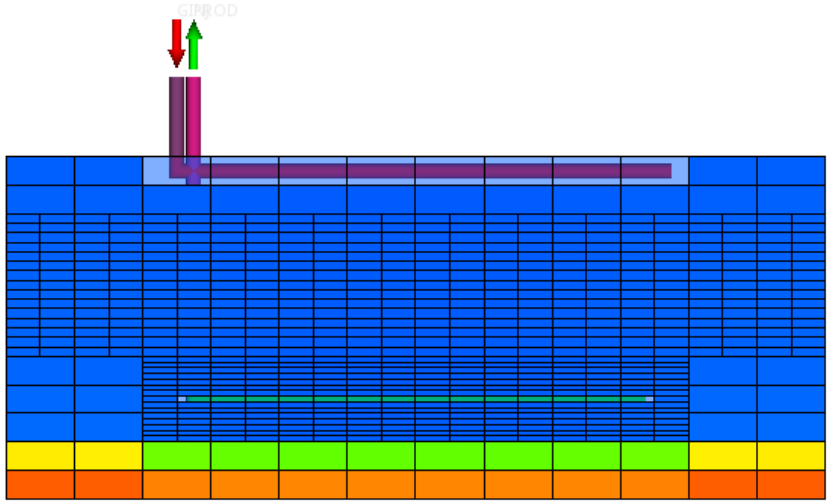
OPM can reproduce qualitatively results of reference simulator!

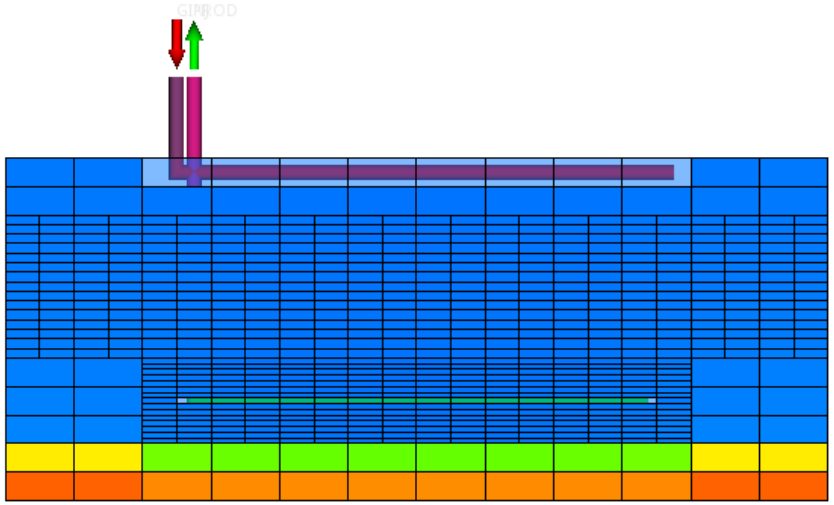
Local grid refinement in OPM Flow: today and beyond

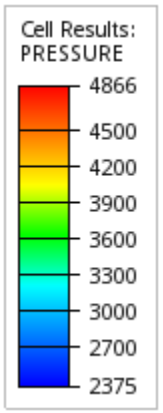
LGR Horizontal Well Injection

Pressure









Reference Simulator
Coarse

Reference Simulator
LGR

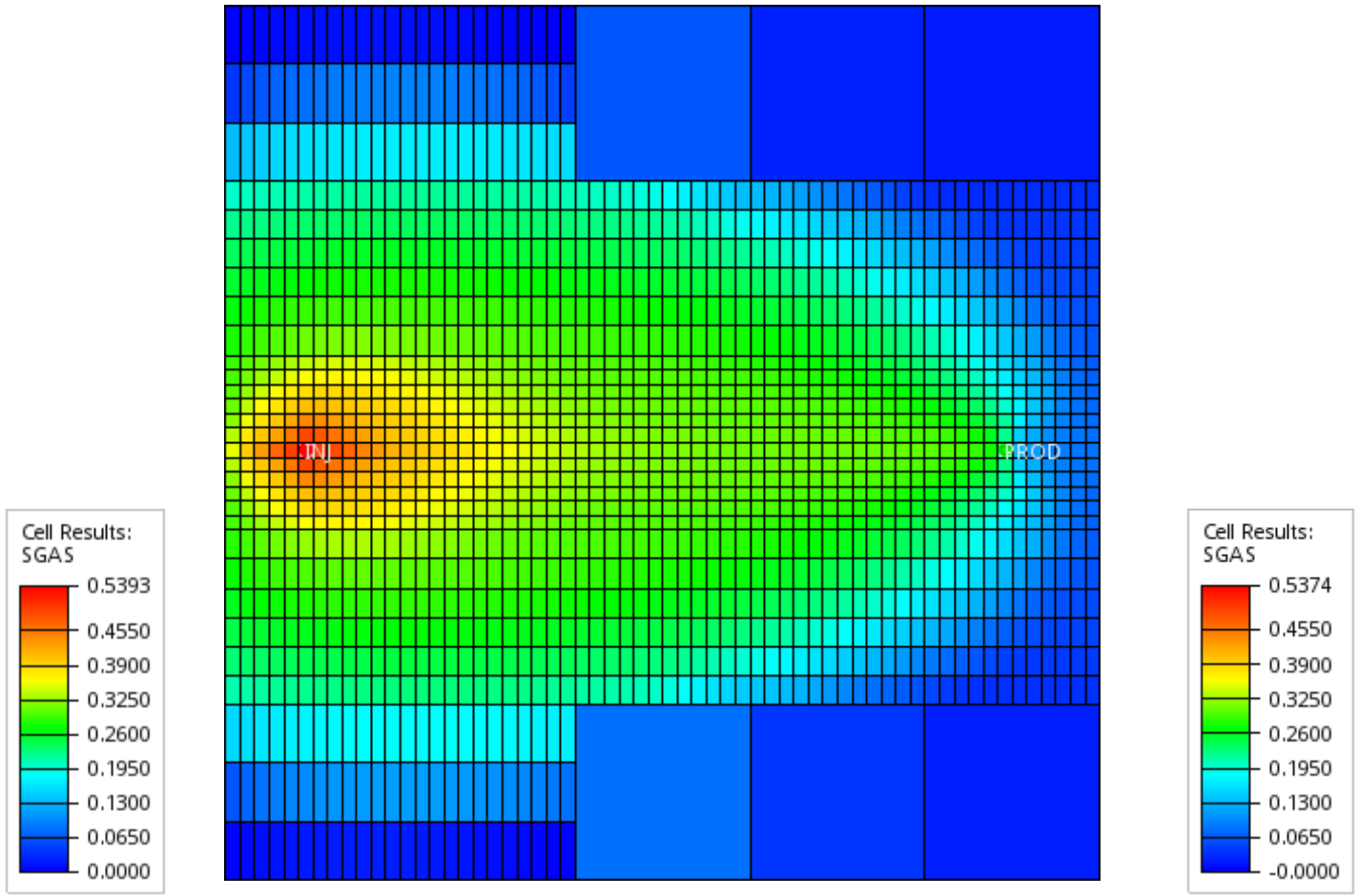
OPM Simulator
LGR

OPM can reproduce qualitatively results of reference simulator!

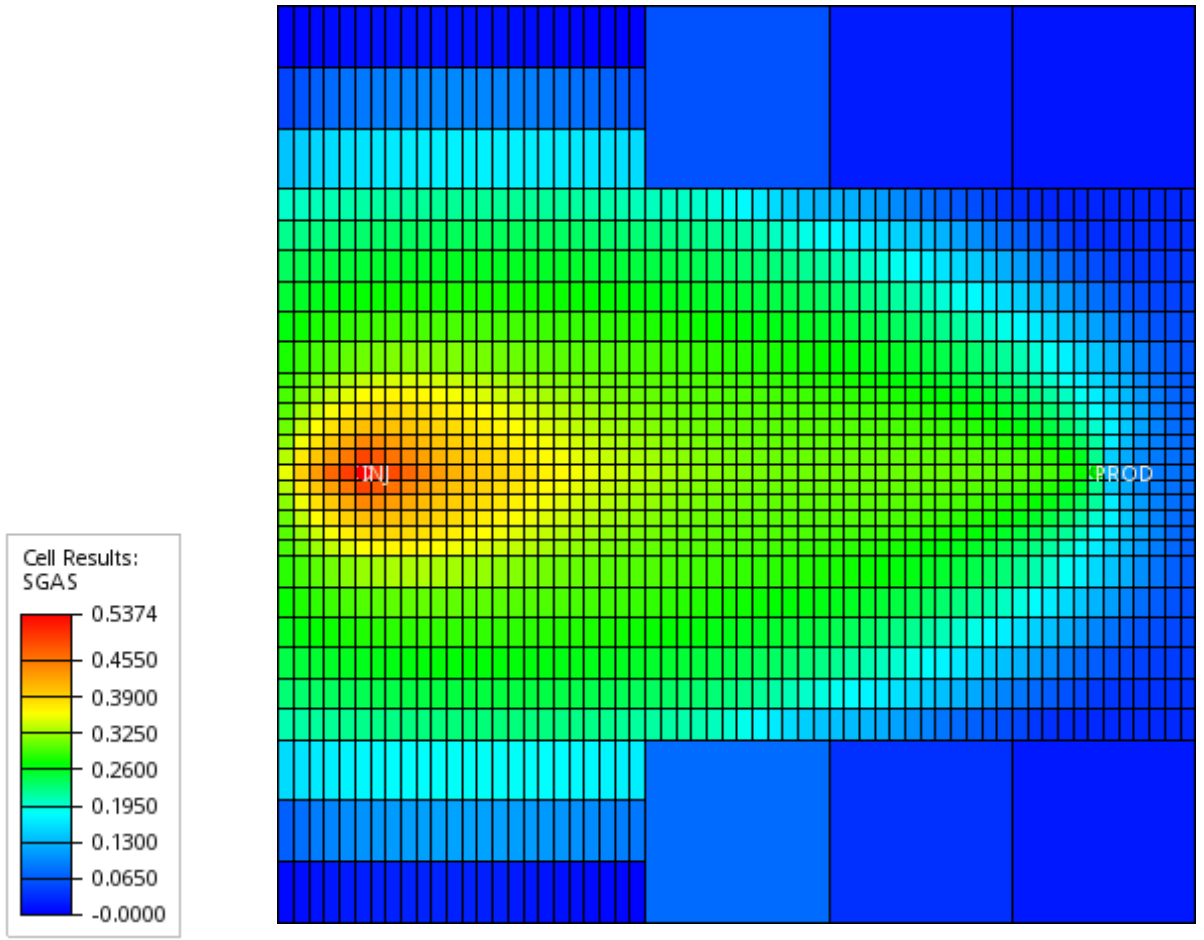
Local grid refinement in OPM Flow: today and beyond

Multiple Carfin

Oil Saturation



Reference Simulator
LGR

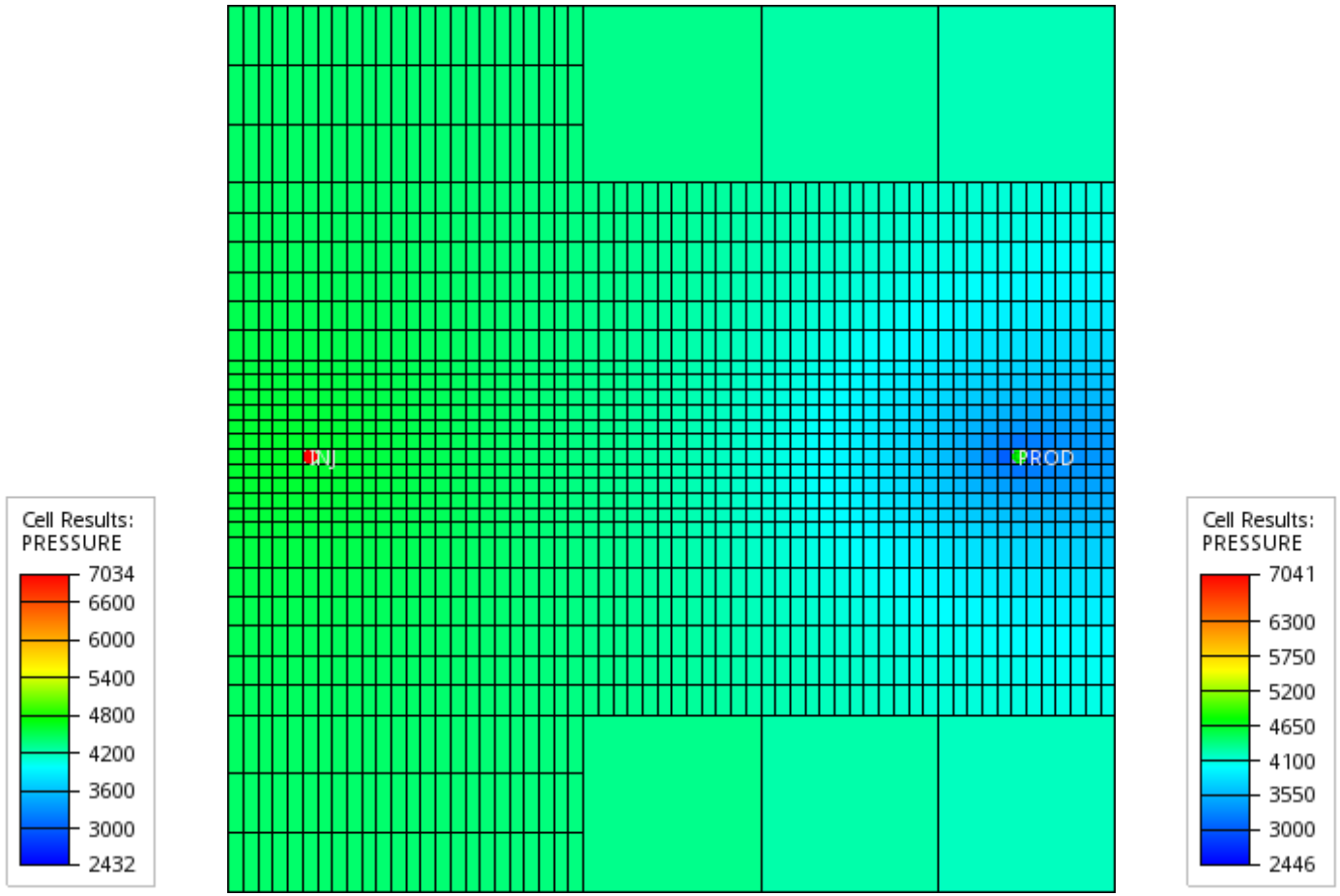


OPM Simulator
LGR

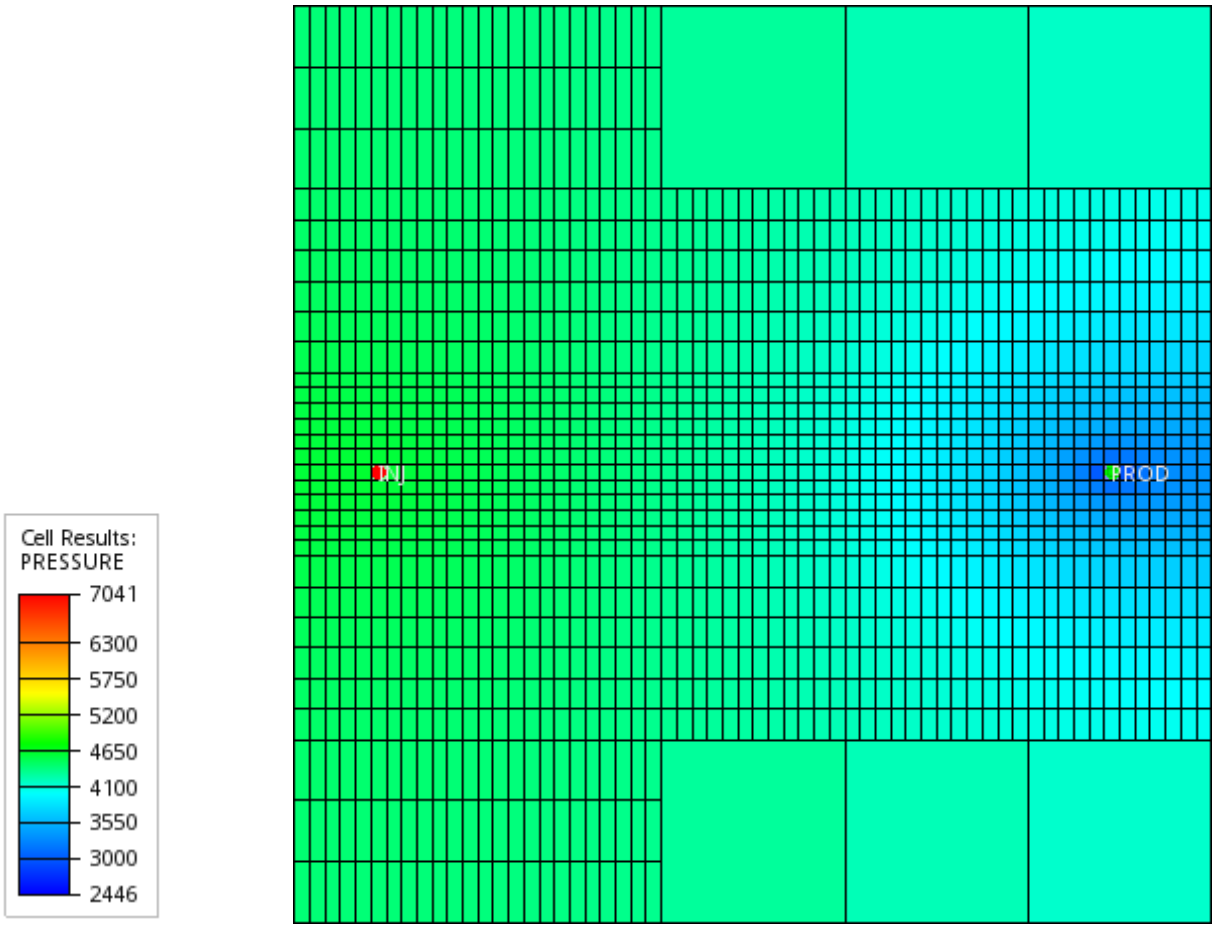
Local grid refinement in OPM Flow: today and beyond

Multiple Carfin

Pressure



Reference Simulator
LGR



OPM Simulator
LGR

OPM
OPEN POROUS MEDIA

Local Grid Refinement in OPM Flow:

today and beyond

Artur Castiel · Antonella Ritorto

OPM Summit · April 2026

Check us out!

https://opm-project.org
https://github.com/OPM

TNO innovation for life

OPM-OP