

# Ensemble Simulations With MRST

## With Examples from CO<sub>2</sub> Foam Simulation

Øystein Klemetsdal<sup>1</sup>   Håvard Heitlo Holm<sup>1</sup>   Alv-Arne Grimstad<sup>2</sup>

<sup>1</sup>Department of Mathematics and Cybernetics, SINTEF Digital, Norway

<sup>2</sup>Department of Petroleum, SINTEF Industry, Norway

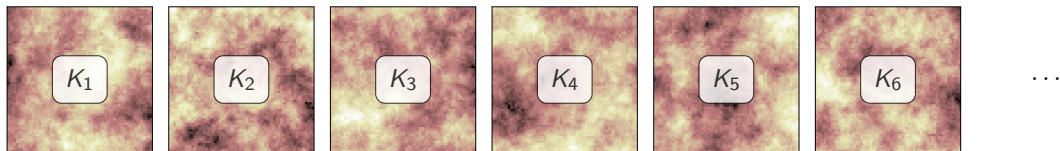
NCCS Webinar  
November 4, 2021

## Uncertainty quantification in reservoir simulation

- Quantity of interest  $u$  typically derived from simulation results  
Water production rate, saturation at time  $t'$ , total amount of  $\text{CO}_2$  stored etc.

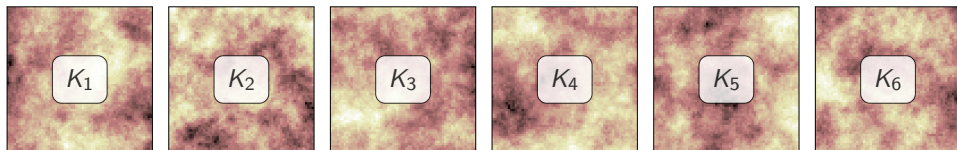
## Uncertainty quantification in reservoir simulation

- Quantity of interest  $u$  typically derived from simulation results  
Water production rate, saturation at time  $t'$ , total amount of  $\text{CO}_2$  stored etc.  
... which are all random variables due to uncertain subsurface properties

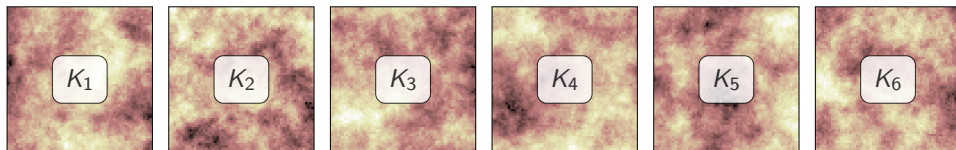


## Uncertainty quantification in reservoir simulation

- Quantity of interest  $u$  typically derived from simulation results  
Water production rate, saturation at time  $t'$ , total amount of  $\text{CO}_2$  stored etc.  
... which are all random variables due to uncertain subsurface properties
- Common to quantify this uncertainty by running **ensemble simulations**



**Ensemble simulations:** Instead of running a single simulation, we use a number of equiprobable realizations of the same model in order to account for **uncertainties in the geomodel** and **uncertainties in the mathematical formulation**.



## In this webinar

Show how to use the MATLAB Reservoir Simulation Toolbox (MRST) to do ensemble simulations, and how it can be applied to uncertainty quantification in CO<sub>2</sub> storage

1. Short description of MRST
2. Introduction to the ensemble module
  - Define a simulation problem, a quantity of interest, and stochastic samples
3. Example: uncertainty quantification CO<sub>2</sub> storage with mobility control
4. Extensions of the module
  - Monte-Carlo/multilevel Monte Carlo, Ensemble-based History Matching

# MATLAB Reservoir Simulation Toolbox (MRST)

Transforming research on  
reservoir modelling

Unique prototyping platform:

- Standard data formats
- Data structures/library routines
- Fully unstructured grids
- Rapid prototyping:
  - Differentiation operators
  - Automatic differentiation
  - Object-oriented framework
  - State functions
- Industry-standard simulation

```
% Three-phase template model
```

```
fluid = initSimpleADIFluid('mu', [1, 5, 0]*centi*po  
'rho', [1000, 700, 0]*kilogram/meter^3, 'n',
```

```
% Constant oil compressibility
```

```
fluid.b0 = @(p, varargin) exp((p/barsa - 10)
```

```
% Construct reservoir model
```

```
gravity reset on  
model = TwoPhaseOilWaterModel(G,
```

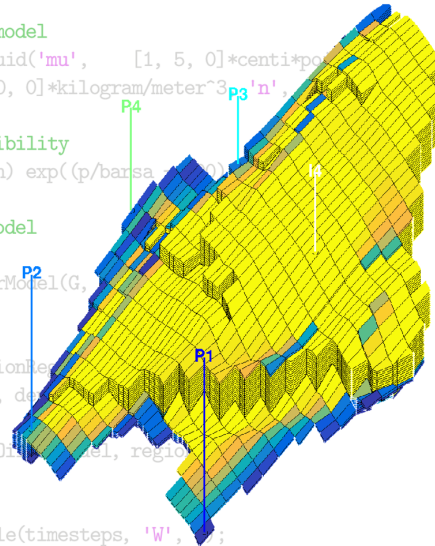
```
%% Define initial state
```

```
region = getInitializationKey  
'datum_depth', de
```

```
state0 = initStateBlackOil(model, region
```

```
% Define schedule
```

```
schedule = simpleSchedule(timesteps, 'W',
```



<http://www.mrst.no>

# MATLAB Reservoir Simulation Toolbox – useful resources

SEARCH

MRST
FAQ
Forum
Modules
Gallery
Download
Documentation
Publications
Contact
Downloadable Resources

### The MATLAB Reservoir Simulation Toolbox (MRST)

MRST is a free open-source software for reservoir modeling and simulation, developed primarily by the [Computational Geosciences](#) group in the [Department of Mathematics](#) and [Physics](#) at [UNIST](#) (Ulsan National Institute of Science and Technology), University of Ulsan, Korea, and TUM.

#### Basic functionality

**Create data attributes**

**Visualization**

**Problem data sets**

#### Discretization and solvers

**TEMA**

**Gridflow**

**Infocube/Infocub**

**Gridflow**

**Gridflow**

**Gridflow**

**Gridflow**

#### Workflow tools

**Basic workflow**

**Flow diagnostics**

**Flow diagnostics**











**Flow diagnostics**

**Flow diagnostics**

**Flow diagnostics**

**Flow diagnostics**

[website](#)

-  **Welcome to the MRST User Group (1)**  
Av MRST-users: The Matlab Reservoir Simulation Toolbox User Group - 6 inlog - 648 visninger 
-  **Could you get the data? (9)**  
Av jsehw\_ @gmail.com - 9 inlog - 58 visninger
-  **Set wells in geology model (and perforation intervals) (3)**  
Av Алексей Илюков - 3 inlog - 7 visninger
-  **EXAMPLE (2)**  
Av 趙杰 底 - 2 inlog - 18 visninger
-  **Invalid MEX-file in the mac os platform (3)**  
Av frankjo\_ @hotmail.com - 3 inlog - 11 visninger
-  **Rate constraints for Compositional models (4)**  
Av xcu\_ @yahoo.com - 4 inlog - 35 visninger
-  **error during interpolation of vaporized oil table (2)**  
Av ltu\_ @gmail.com - 2 inlog - 22 visninger
-  **well equation in TwoPhaseWaterGasModel model (2)**  
Av kai wang - 2 inlog - 16 visninger
-  **incompTPFA generating pressure and flux Nan's..... (3)**  
Av Paul Morris - 3 inlog - 4 visninger

**An Introduction to Reservoir Simulation Using MATLAB/GNU Octave**  
User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)  
  
Knut-Andreas Lie

✓ Access     Open access  
Knut-Andreas Lie, SINTEF, Norway

Publisher: Cambridge University Press  
Online publication date: July 2019  
Print publication year: 2019  
Online ISBN: 9781108591416

DOI: <https://doi.org/10.1017/9781108591416>

textbook

```
>> help computeTrans
Compute transmissibilities.

SYNOPSIS:
    T = computeTrans(G, rock)
    T = computeTrans(G, rock, 'pm', pv, ... )

PARAMETERS:
    G - Grid structure as described by grid_structure.

    rock - Rock data structure with valid field 'perm'. The permeability
    is assumed to be in measured in units of metres squared (m^2).
    The function 'darcy' to convert from darcies to m^2, e.g..

        perm = convertFrom(perm, m2li=darcy)

    if the permeability is provided in millidarcies.

    The field rock.perm may have ONE column for a scalar
    permeability in each cell, TWO/THREE columns for a diagonal
    permeability in each cell (in 2/3 D) and THREE/SIX columns for a
    symmetric full tensor permeability. In the latter case, each
    cell gets the permeability tensor

        K_u1 = [ k1 k2 ]      in two space dimensions
               [ k2 k3 ]

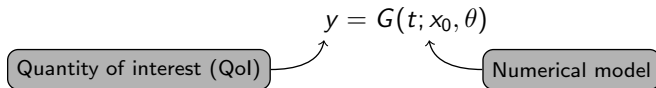
    |
    |
    RETURNS:
        T - half-transmissibilities for each local face of each grid
        The number of half-transmissibilities equals the number
```

[illegible]

online tutorials

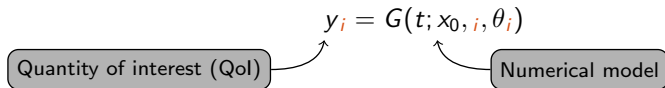


# The Ensemble Module



- **We have** numerical representation of reservoir  $\theta$ , initial state  $x_0$ , numerical model  $G$
- **We want** quantity of interest (Qol)  $y$  at given time  $t$
- Reservoir parameters  $\theta$  and initial state  $x_0$  known exactly  $\rightarrow$  compute Qol directly

# The Ensemble Module



- **We have** numerical representation of reservoir  $\theta$ , initial state  $x_0$ , numerical model  $G$
- **We want** quantity of interest (Qol)  $y$  at given time  $t$
- Reservoir parameters  $\theta$  and/or initial state  $x_0$  given by **probability distributions**  $p(\theta)$ ,  $p(x_0)$ 
  - Generate ensemble with  $N$  members by sampling  $\theta_i \sim p(\theta)$ ,  $x_{0,i} \sim p(x_0)$
  - Compute Qol for each ensemble member
  - Estimate Qol (ensemble mean) and associated uncertainty (ensemble variance)

# The Ensemble Module

$$y = G(t; x_0, \theta)$$

## BaseProblem

All elements of a simulation model that is common across ensemble members

Examples: grid, injection strategy, well position

## Samples

Stochastic realization of uncertain parameters

Examples: permeability, porosity, initial saturation

## QoI

Quantity of interest (QoI) that we want to estimate with ensemble

Examples: stored CO<sub>2</sub> volume, breakthrough time

# The Ensemble Module

$$y = G(t; x_0, \theta)$$

## BaseProblem

All elements of a simulation model that is common across ensemble members

Examples: grid, injection strategy, well position

## Samples

Stochastic realization of uncertain parameters

Examples: permeability, porosity, initial saturation

## QoI

Quantity of interest (QoI) that we want to estimate with ensemble

Examples: stored CO<sub>2</sub> volume, breakthrough time

# The Ensemble Module

$$y = G(t; \mathbf{x}_0, \theta)$$

## BaseProblem

All elements of a simulation model that is common across ensemble members

Examples: grid, injection strategy, well position

## Samples

Stochastic realization of uncertain parameters

Examples: permeability, porosity, initial saturation

## QoI

Quantity of interest (QoI) that we want to estimate with ensemble

Examples: stored CO<sub>2</sub> volume, breakthrough time

# The Ensemble Module

$$y = G(t; x_0, \theta)$$

## BaseProblem

All elements of a simulation model that is common across ensemble members

Examples: grid, injection strategy, well position

## Samples

Stochastic realization of uncertain parameters

Examples: permeability, porosity, initial saturation

## QoI

Quantity of interest (QoI) that we want to estimate with ensemble

Examples: stored CO<sub>2</sub> volume, breakthrough time

# Base problem

- Defines all elements of a simulation model that is common across ensemble
- Worked example: quarter five-spot problem with water injection into oil
  - Common elements: fluid model, injection rates, well placement
  - Stochastic parameters: permeability and porosity

# Base problem

- Defines all elements of a simulation model that is common across ensemble
- Worked example: quarter five-spot problem with water injection into oil
  - Common elements: fluid model, injection rates, well placement
  - Stochastic parameters: permeability and porosity

```
example = MRSTExample('qfs_wo'); % Setup up example  
example.plot();                  % Plot setup
```



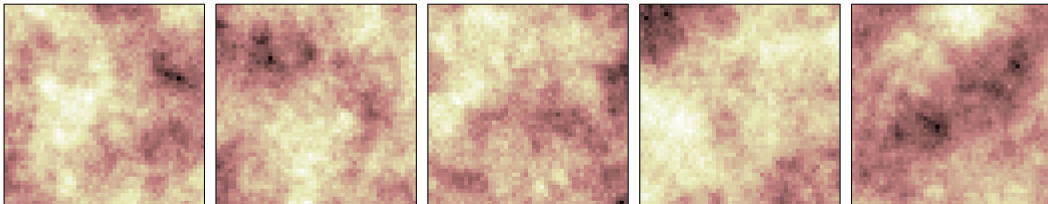


# Stochastic Samples

- Stochastic samples can be defined in three ways:
  1. By a function that generates a sample when called
  2. By precomputed samples in a cell array
  3. By precomputed samples stored on disk (e.g., as DECK files)
- Here: use RockSamples to define stochastic permeability/porosity

```
% Set up samples using a generator function
generatorFn = @(problem, seed) ...
    generateRockSample(example.model.G.cartDims, ... % Gaussian field dimensions
        'seed'      , seed      , ... % Random number seed
        'toVector', true      );    % Stretch out to vector
samples = RockSamples('generatorFn', generatorFn); % Set up rock samples
```

# Stochastic Samples



```
for i = 1:5
    data    = samples.getSample(i, problem);    % Get sample number i
    problem = samples.setSample(data, problem); % Set sample to problem
    % Inspect rock sample
    example.plot(problem.SimulatorSetup.model.rock, 'log10', true); colormap(pink)
end
```

# Quantity of Interest (QoI)

- QoI is what we are actually interested in obtaining from the simulations
- Can be any user-defined quantity – implement based on one of the QoI classes
- In this example: oil production rate as a function of time

```
% Oil production rate as a function of time  
qoi = WellQoI('fldname', 'qOs', 'wells', 2);  
% Water saturation after 20, 300 and 700 days  
qoi = ReservoirStateQoI('name', 'sW', 'time', [20, 300, 700]*day);  
% User-defined QoI based on a suitable parent class  
qoi = CO2StorageQoI('type', 'mass');
```

# The MRSTEnsemble Class – Putting it All Together

- Class MRSTEnsemble collects base problem, samples and QoI
  - Functionality for simulating ensemble members (all or a subset)
  - Functionality for writing/reading data to/from disk

```
ensemble = MRSTEnsemble(example, samples, qoi, ... % Gather components
                        'directory', dataDir, ... % Data directory
                        'simulationStrategy', 'background'); ... % Simulate in background
```

# The MRSTEnsemble Class – Putting it All Together

- Class MRSTEnsemble collects base problem, samples and QoI
  - Functionality for simulating ensemble members (all or a subset)
  - Functionality for writing/reading data to/from disk
- Three ways to simulate multiple ensemble members
  1. In serial (only for small ensembles and debugging)
  2. In parallel using background MATLAB sessions
  3. In parallel using the MATLAB parallel computing toolbox

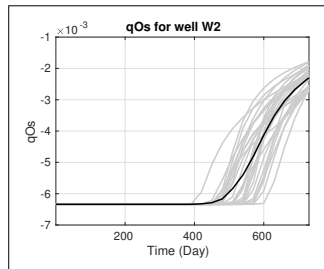
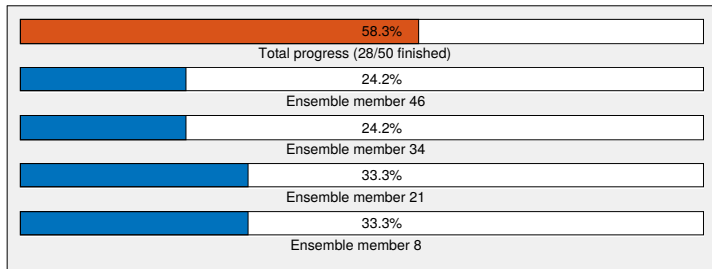
```
ensemble = MRSTEnsemble(example, samples, qoi, ... % Gather components
                        'directory', dataDir, ... % Data directory
                        'simulationStrategy', 'background'); ... % Simulate in background
```

# The MRSTEnsemble Class – Simulating Ensemble Members

- Simulate ensemble members in three different ways
  1. Specific ensemble members, referenced by number
  2. Batch of  $n$  ensemble members, starting from the last simulated member
  3. The entire ensemble
- Plot progress while the simulations are running

```
ensemble.simulateEnsembleMembers('range', 3:5);           % Simulate samples 3 to 5  
ensemble.simulateEnsembleMembers('batchSize', 20);        % 20 more samples  
ensemble.simulateEnsembleMembers('plotProgress', true);   % Simulate entire ensemble
```

# The MRSTEnsemble Class – Simulating Ensemble Members



```
ensemble.simulateEnsembleMembers('range', 3:5);           % Simulate samples 3 to 5
ensemble.simulateEnsembleMembers('batchSize', 20);         % 20 more samples
ensemble.simulateEnsembleMembers('plotProgress', true);    % Simulate entire ensemble
```

# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

## Efficiency of CO<sub>2</sub> foam mobility control with heterogeneous reservoir properties (Grimstad and Klemetsdal, TCCS-11)

- Storage of 250 kt CO<sub>2</sub> per year into brine-filled, 1400×1400×100 m reservoir
- Quadratic, Corey-type relative permeabilities, capillary entry pressure of 0.15 bar
- Three injection strategies (all simulations are run until breakthrough in production well)
  1. Inject CO<sub>2</sub> only
  2. Inject surfactant (1% wt) during first four years
  3. Inject surfactant (1% wt) during first four years, perm-dependent surfactant strength (surfactant model from Vassenden and Holt, 2000)

```
% Base case is defined in the function qfs_foam  
example = MRSTExample('qfs_foam', 'cartDims', [30, 30, 20]);
```



# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

## Setting up stochastic rock samples

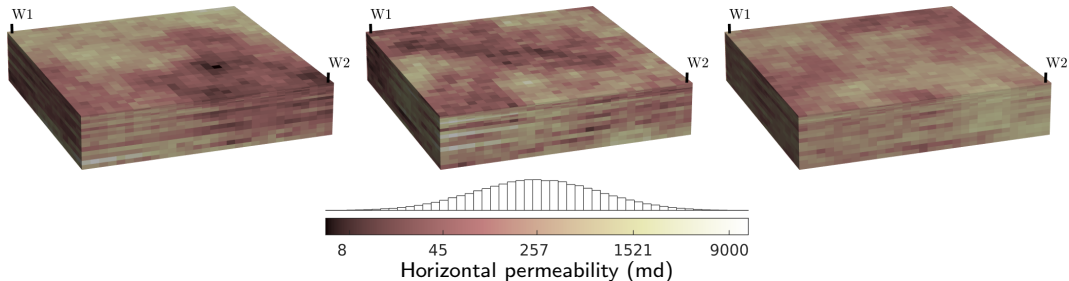
- Layered structure with log-normal permeability (mean 260 md) and normal porosity

$$\log(K) = \mathcal{N}(-12.5, 0.5), \quad \phi = \mathcal{N}(0.25, 0.1)$$

- Vertical to horizontal permeability ratio = 1/10, 100 realizations

```
% Set up function for generating rock samples
generatorFn = @(problem, seed) generateLayeredRockSample(cartDims, seed, ...
    'avg_poro', 0.25, ...
    'min_poro', 1e-3, ...
    'avg_perm', 260*milli*darcy);
% Generate realizations, set kv/kh = 1/10, store to disk using ResultHandler rh (truncated)
samples = RockSamples('data', rh); % Set up rock samples
```

# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control



```
% Set up function for generating rock samples
```

```
generatorFn = @(problem, seed) generateLayeredRockSample(cartDims, seed, ...  
                                                         'avg_poro', 0.25, ...  
                                                         'min_poro', 1e-3, ...  
                                                         'avg_perm', 260*milli*darcy);
```

```
% Generate realizations, set kv/kh = 1/10, store to disk using ResultHandler rh (truncated)  
samples = RockSamples('data', rh); % Set up rock samples
```

# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

## Define quantity of interest (QoI)

- QoI is storage efficiency (fraction of pore volume occupied by CO<sub>2</sub>) at breakthrough
- Implemented in user-defined QoI class with overloaded method computeQoI

```
classdef CO2StorageQoI < ReservoirStateQoI
    ...
    methods
        function u = computeQoI(qoi, problem)
            state = problem.OutputHandlers.states{end}; % Get final state
            [pv, sG] = model.getProps(state, 'PoreVolume', 'sG'); % Get PV and CO2 sat
            u = sum(pv.*sG)./sum(pv); % PV-weighted CO2 sat
        end
    end
end
```

# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

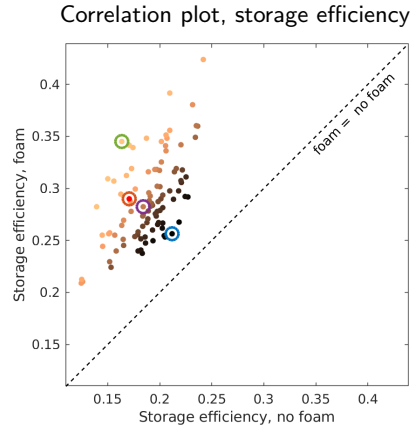
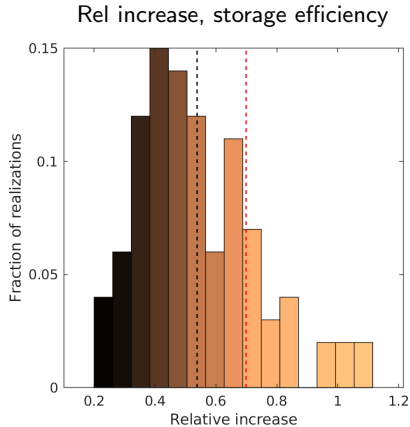
## Set up and simulate ensemble

- One ensemble for each strategy (no foam, foam, and permeability-dependent foam)
- Simulate using background sessions and store all simulation data, not just the QoI

```
ensemble = MRSTEnsemble(example, samples, qoi, ...  
    'directory'      , dataDir      , ... % Directory for storing results  
    'simulationStrategy', 'background', ... % Use background MATLAB sessions  
    'maxWorkers'     , maxWorkers  , ... % Max # concurrent MATLAB sessions  
    'storeOutput'    , true        );    % Save all output (not just QoI)  
ensemble.simulateEnsembleMembers();      % Simulate all 100 members
```

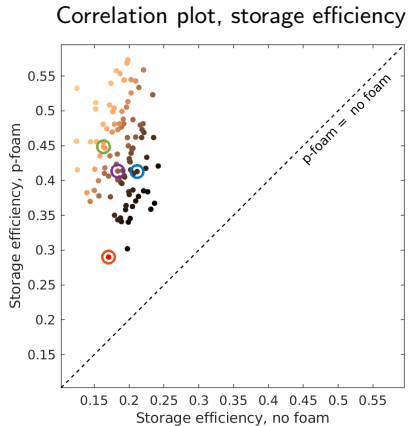
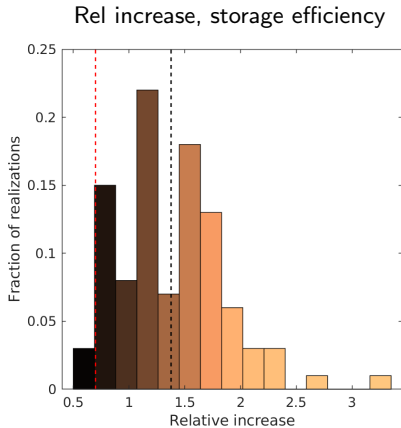
# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

## Results – foam vs. no foam



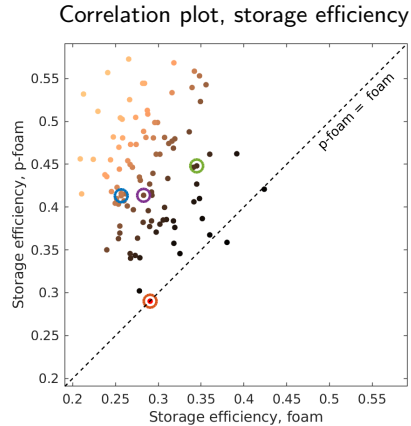
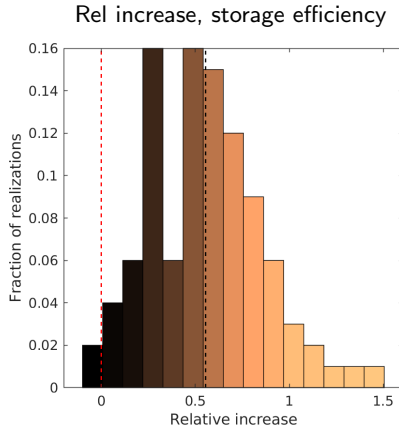
# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

## Results – perm-dependent foam vs. no foam



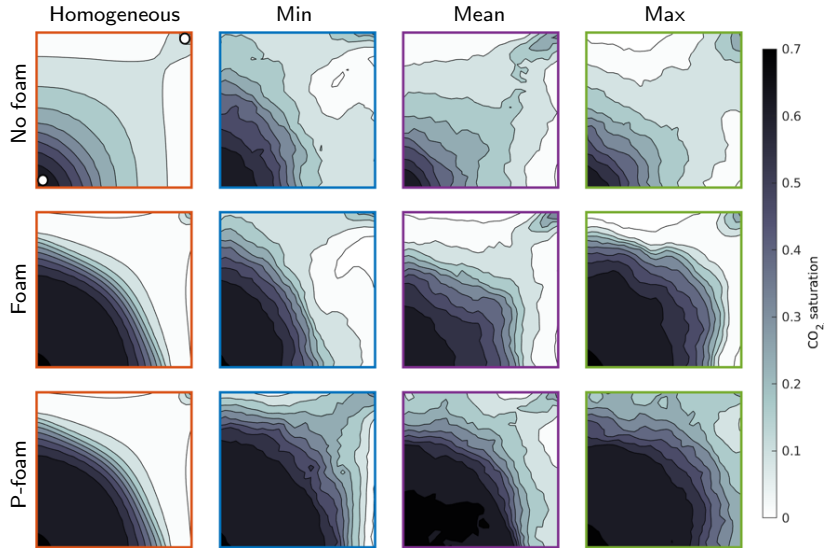
# Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

## Results – perm-dependent foam vs. foam



## Example: Efficiency of CO<sub>2</sub> Foam Mobility Control

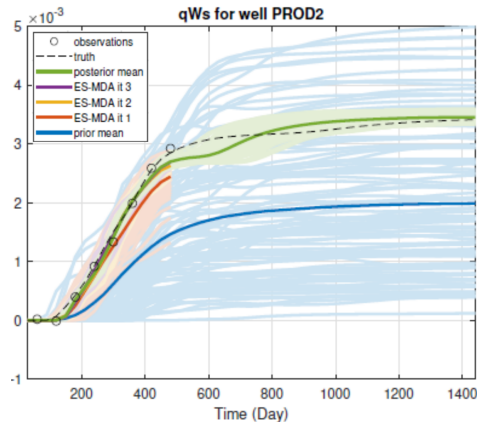
CO<sub>2</sub> plumes (horizontal)  
Min/Mean/Max  
correspond to gain with  
foam vs. no foam





# Extensions of the Ensemble Module

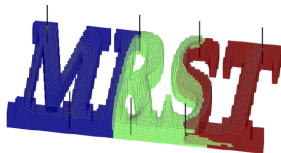
- Monte Carlo simulation (MC)
  - Simulate new realizations until QoI uncertainty (RMSE) is less than prescribed tolerance
- Multilevel Monte Carlo simulation (MLMC)
  - Highly generic level hierarchy (spatial/temporal upscaling, solver-based, etc.)
  - Potentially much faster alternative to MC
- Ensemble-based history matching
  - Ensemble smoother with multiple data assimilations (ES-MDA)



# Acknowledgements

*Thanks to Stein Krogstad for important contributions to the ensemble module*

*The research reported in this presentation was funded in part by the Research Council of Norway through grant no. 280950 and in part by Equinor Energy AS, Total E&P Norge AS, and Wintershall DEA Norge AS*



`mrst.no`

Full source code available in the open-source MATLAB  
Reservoir Simulation Toolbox

# The MRSTEnsemble Class – Simulating Ensemble Members

simulateEnsembleMember does everything involved in simulating a single member

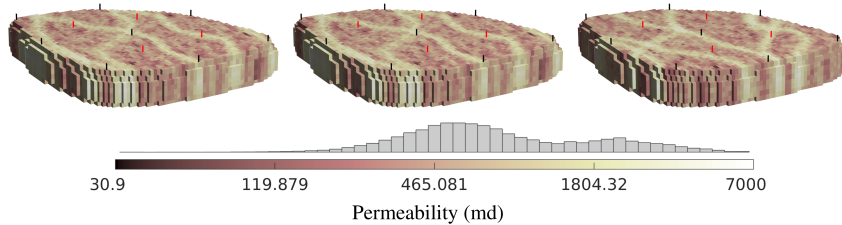
```
function simulateEnsembleMember(ensemble, seed, varargin)
    % Get base problem
    baseProblem = ensemble.getBaseProblem;
    % Set up sample problem from seed
    problem = ensemble.samples.getSampleProblem(baseProblem, seed);
    % Solve problem
    ensemble.solve(problem);
    % Compute QoI
    ensemble.qoi.getQoI(problem);
end
```

## Example: samples from disk – the Egg model

- Geomodel realizations often given as a collection of files (e.g., DECK)
- The ensemble module provides a convenient way of reading samples from disk
- Here: The Egg ensemble (Jansen et. al, 2014)
  - Quantity of interest: oil rate in all production wells

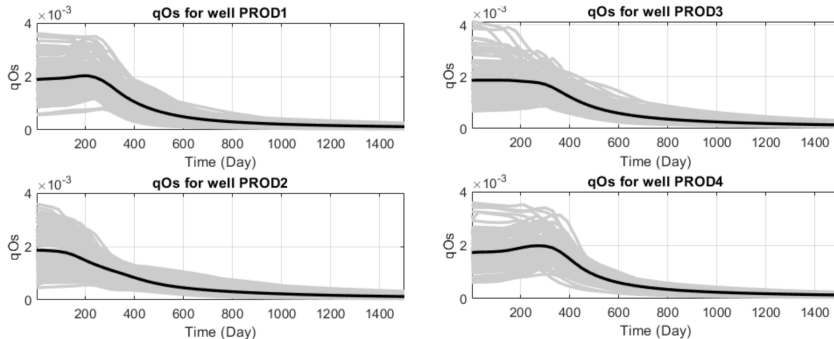
```
example = MRSTExample('egg_wo'); % Set up example
% Function getDeckEGG reads an Egg deck file from disk and returns a model
% Realizations are numbered from 0 to 100 but seed must be > 0, hence seed-1.
generatorFn = @(problem, seed) getDeckEGG('realization', seed-1);
samples      = DeckSamples('generatorFn', generatorFn, ... % Generator function
                           'num'          , 101          ); % Number of samples
% For our QoI, we choose the total oil production rate
is_prod = vertcat(example.schedule.control(1).W.sign) < 0;
qoi      = WellQoI('wellIndices', is_prod, 'fldname', 'qOs');
```

## Example: samples from disk – the Egg model



```
example = MRSTExample('egg_wo'); % Set up example
% Function getDeckEGG reads an Egg deck file from disk and returns a model
% Realizations are numbered from 0 to 100 but seed must be > 0, hence seed-1.
generatorFn = @(problem, seed) getDeckEGG('realization', seed-1);
samples      = DeckSamples('generatorFn', generatorFn, ... % Generator function
                           'num'          , 101          ); % Number of samples
% For our QoI, we choose the total oil production rate
is_prod = vertcat(example.schedule.control(1).W.sign) < 0;
qoi      = WellQoI('wellIndices', is_prod, 'fldname', 'qOs');
```

# Example: samples from disk – the Egg model



```
ensemble = MRSTEnsemble(example, samples, qoi, ...  
    'simulationStrategy', 'background'); % Run in the background  
% We simulate 8 samples. Each time the code in this block is called, 8 new samples will be  
% simulated until all ensemble members have been run. To reset, call ensemble.reset();  
ensemble.simulateEnsembleMembers('batchSize', 8, 'plotProgress', true);
```