Use Of Dynamically Adapted Basis Functions To Accelerate Multiscale Simulation Of Complex Geomodels

Øystein S. Klemetsdal[1,2] Olav Møyner[1,2] Knut-Andreas Lie[1,2]

[1]Department of Mathematical Sciences, NTNU, Norway
[2]Department of Mathematics and Cybernetics, SINTEF Digital

ECMOR XVI – 16th European Conference on the Mathematics of Oil Recovery
September 3–6, 2018, Barcelona, Spain

Introduction

- Linear systems in reservoir simulation are typically ill-conditioned and challenging to solve
  → need for iterative solvers with efficient preconditioners[1]
- Constrained pressure-residual method (CPR)[2]: physics-based preconditioner
  - inexpensive pressure estimate used in initial guess for solution to full system
- Multiscale methods have been applied as CPR pressure solver[3]
- Herein: improve convergence of linear solver by applying multiple multiscale operators that
  - target specific features in the geological model;
  - resolve dynamic couplings between pressure and other variables

[1]Lacroix et al., 2003, [2]Wallis, 1983; Wallis et al., 1985; Gries et al., 2014, [3]Cusini et al (2015)

Governing equations

- Conservation of mass for component β:

$$\frac{\partial}{\partial t}\left(\phi\left[\rho_w S_w X_{w\beta} + \rho_o S_o X_{o\beta} + \rho_v S_v X_{v\beta}\right]\right) + \nabla\cdot\left(\rho_w X_{w\beta}\vec{v}_w + \rho_o X_{o\beta}\vec{v}_o + \rho_v X_{v\beta}\vec{v}_v\right) = q_\beta$$

Mass balance

$$\vec{v}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha}\mathbf{K}\left(\nabla p_\alpha - \rho_\alpha g \nabla z\right)$$

Darcy's law

- Fugacity balance:

$$f_\beta^o(p, T, X_{o1}, \ldots, X_{oN_c}) = f_\beta^v(p, T, X_{v1}, \ldots, X_{vN_c})$$

- Closure relations:

Mass fractions sum to 1

Phases fill PV

$$S_w + S_o + S_v = 1, \quad \sum_{\beta=1}^{N_c} X_{\alpha\beta} = 1, \quad \alpha = w, o, v$$

$$p_o = p_w + P_{cow}(S_w, S_o), \quad p_v = p_o + P_{cvo}(S_o, S_v)$$

Capillary pressure

Discretization

- Introduce grid $\{\Omega_i\}_{i=1}^N$, backward Euler for temporal discretization, integrate over each cell $\longrightarrow$ Discrete conservation of mass for component β in cell i:

$$F_\beta^i = A_\beta^i + \sum_{j \in \mathcal{N}(i)} G_\beta^{i,j} - Q_\beta^i = 0$$

Accumulation
Flux
Sources/sinks

- Write residuals and variables on vector form $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, where

$$\mathbf{F} = (\mathbf{F}_1, \dots, \mathbf{F}_{N_c}) = (F_1^1, \dots, F_1^N, \dots, F_{N_c}^1, \dots, F_{N_c}^N) = (\mathbf{F}_p, \mathbf{F}_s)$$
$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_c}) = (p_1, \dots, p_N, x_2^1, \dots, x_2^N, \dots, x_{N_c}^1, \dots, x_{N_c}^N) = (\mathbf{x}_p, \mathbf{x}_s)$$

Discretization

- Introduce grid $\{\Omega_i\}_{i=1}^{N}$, backward Euler for temporal discretization, integrate over each cell
  $\rightarrow$ Discrete conservation of mass for component β in cell i:

$$F_\beta^i = A_\beta^i + \sum_{j \in \mathcal{N}(i)} G_\beta^{i,j} - Q_\beta^i = 0$$

Accumulation | Flux | Sources/sinks

- Write residuals and variables on vector form $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, where

$N_c \times N$ residuals

$$\mathbf{F} = (\mathbf{F}_1, \dots, \mathbf{F}_{N_c}) = \underbrace{(F_1^1, \dots, F_1^N, \dots, F_{N_c}^1, \dots, F_{N_c}^N)}_{N_c \times N \text{ residuals}} = (\mathbf{F}_p, \mathbf{F}_s)$$

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_c}) = (\underbrace{p_1, \dots, p_N}_{N \text{ pressure variables}}, \underbrace{x_2^1, \dots, x_2^N, \dots, x_{N_c}^1, \dots, x_{N_c}^N}_{(N_c - 1) \times N \text{ secondary variables}}) = (\mathbf{x}_p, \mathbf{x}_s)$$

N pressure variables

$(N_c - 1) \times N$
secondary variables

Preconditioning: constrained pressure-residual (CPR)

- Linearize and neglect higher-order terms $\longrightarrow$ Newton-Raphson method:

$$-\underbrace{\begin{bmatrix} \mathbf{J}_{pp} & \mathbf{J}_{ps} \\ \mathbf{J}_{sp} & \mathbf{J}_{ss} \end{bmatrix}}_{\text{Jacobian matrix}} \underbrace{\begin{bmatrix} \Delta\mathbf{x}_p \\ \Delta\mathbf{x}_s \end{bmatrix}}_{\text{Newton update}} = \underbrace{\begin{bmatrix} \mathbf{F}_p \\ \mathbf{F}_s \end{bmatrix}}_{\text{Residual}}$$

Jacobian matrix $(\mathbf{J}_{\alpha,\beta})_{i,j} = \partial \mathbf{F}^i_\alpha / \partial \mathbf{x}^j_\beta$

Newton update

Residual

- Typically solved iteratively, convergence rate reduced by
  - Mixed elliptic/hyperbolic character $\longrightarrow$ pressure is a strong variable
  - Large aspect ratios and variations in rock properties $\longrightarrow$ ill-conditioned systems

  Effective preconditioner crucial
- Constrained pressure residual method (CPR)
  1. Decouple system so that $(\mathbf{J}^*_{pp})^{-1}\mathbf{J}^*_{ps}$ is "small"
     - Obtain inexpensive estimate to pressure update $\Delta\mathbf{x}_p$
  2. Solve full system using $(\Delta\mathbf{x}_p, \mathbf{0})$ as initial guess

Preconditioning: constrained pressure-residual (CPR)

- Linearize and neglect higher-order terms → Newton-Raphson method:

Almost elliptic (parabolic) pressure equation $-\mathbf{J}^*_{pp}\Delta\mathbf{x}_p = \mathbf{F}^*_p$

$$-\underbrace{\begin{bmatrix} \mathbf{J}^*_{pp} & \mathbf{J}^*_{ps} \\ \mathbf{J}_{sp} & \mathbf{J}_{ss} \end{bmatrix}}_{\text{Jacobian matrix}} \underbrace{\begin{bmatrix} \Delta\mathbf{x}_p \\ \Delta\mathbf{x}_s \end{bmatrix}}_{\text{Newton update}} = \underbrace{\begin{bmatrix} \mathbf{F}^*_p \\ \mathbf{F}_s \end{bmatrix}}_{\text{Residual}}$$

Jacobian matrix $(\mathbf{J}_{\alpha,\beta})_{i,j} = \partial \mathbf{F}^i_\alpha / \partial \mathbf{x}^j_\beta$

Newton update

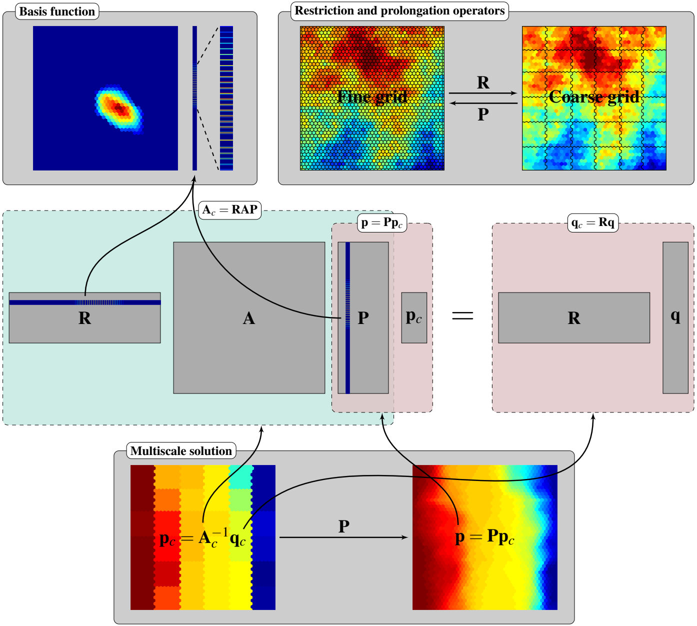
Residual

- Typically solved iteratively, convergence rate reduced by
  - Mixed elliptic/hyperbolic character → pressure is a strong variable
  - Large aspect ratios and variations in rock properties → ill-conditioned systems

  Effective preconditioner crucial

- Constrained pressure residual method (CPR)
  1. Decouple system so that $(\mathbf{J}^*_{pp})^{-1}\mathbf{J}^*_{ps}$ is "small"
     - Obtain inexpensive estimate to pressure update $\Delta\mathbf{x}_p$
  2. Solve full system using $(\Delta\mathbf{x}_p, \mathbf{0})$ as initial guess

Multiscale methods
(Hou and Wu, 1997; Efendiev and Hou, 2009)



Iterative multiscale multibasis method

- Multiscale methods typically resolve global low-frequency errors quite effectively
- Contain local high-frequency errors due to localization introduced to define basis functions
- Iterative framework[1]:

Smoothing operation (ILU(k))

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + S(\mathbf{A}, \mathbf{q} - \mathbf{A}\mathbf{x}^{k}),$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \mathbf{P}\mathbf{A}_c^{-1}\mathbf{R}(\mathbf{q} - \mathbf{A}\mathbf{x}^{k+1/2})$$

Multiscale operation

[1] Hajibeygi et al., 2008; Wang et al., 2014

Iterative multiscale multibasis method

- Multiscale methods typically resolve global low-frequency errors quite effectively
- Contain local high-frequency errors due to localization introduced to define basis functions
- Iterative framework[1] with multiple multiscale operators:

Smoothing operation (ILU(k))

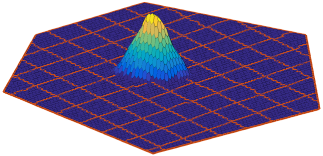
$$\mathbf{x}^{k+(2\ell-1)/2N_P} = \mathbf{x}^{k+(\ell-1)/N_P} + S^{\ell}(\mathbf{A}, \mathbf{q} - \mathbf{A}\mathbf{x}^{k+(\ell-1)/N_P}),$$
$$\mathbf{x}^{k+\ell/N_P} = \mathbf{x}^{k+(2\ell-1)/2N_P} + \mathbf{P}^{\ell}(A_c^{\ell})^{-1}\mathbf{R}^{\ell}(\mathbf{q} - \mathbf{A}\mathbf{x}^{k+(2\ell-1)/2N_P}).$$

Multiscale operation

- Each multiscale operator can target specific *feature* in geomodel
  → Feature-enhanced iterative multiscale multibasis method[2]

[1] Hajibeygi et al.; 2008, Wang et al., 2014, [2] Lie et al., 2016

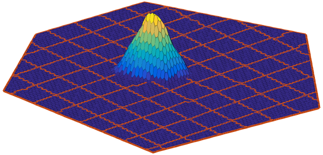
Iterative multiscale multibasis method



General partition (always included)

- Similar block sizes (e.g. rectilinear partition)
- MsRSB basis functions (Møyner and Lie, 2015)
- Resolves global pressure field

Iterative multiscale multibasis method

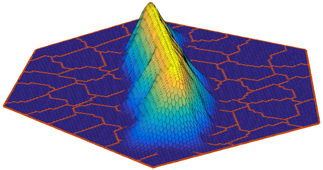


General partition (always included)

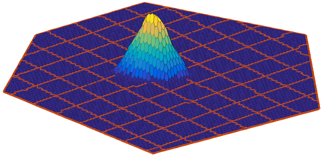
- Similar block sizes (e.g. rectilinear partition)
- MsRSB basis functions (Møyner and Lie, 2015)
- Resolves global pressure field

Static geomodel features

- Permeability/fractures/wells etc.
- Resolves local errors introduced by geological features
- MsRSB or specialized basis functions



Iterative multiscale multibasis method

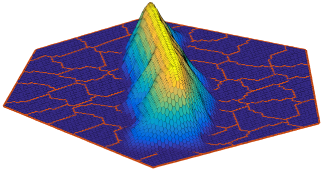


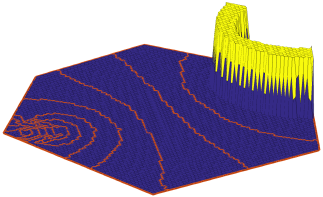
General partition (always included)

- Similar block sizes (e.g. rectilinear partition)
- MsRSB basis functions (Møyner and Lie, 2015)
- Resolves global pressure field

Static geomodel features

- Permeability/fractures/wells etc.
- Resolves local errors introduced by geological features
- MsRSB or specialized basis functions

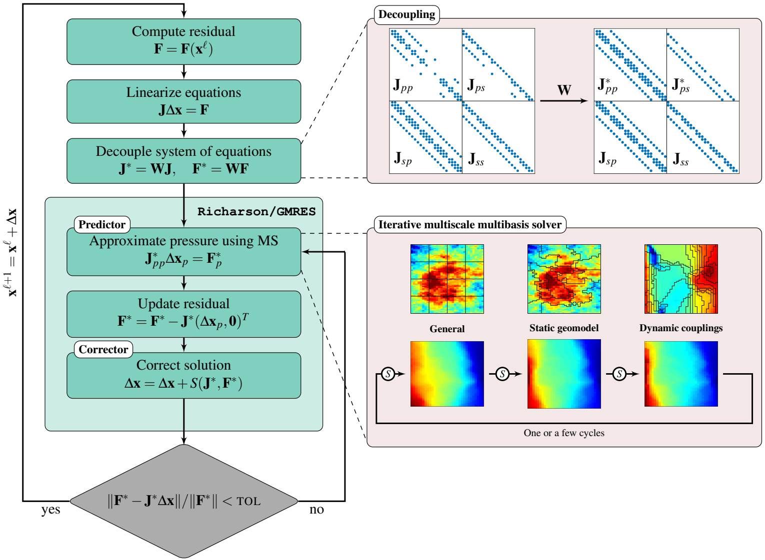




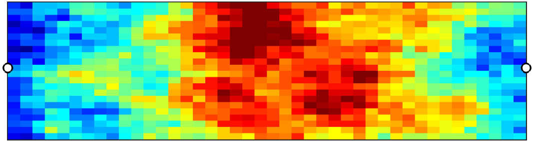
Dynamic couplings

- Partitions based on e.g, $\Delta p \rightarrow$ dynamic
- Local pressure changes along saturation fronts etc.
- Constant basis functions

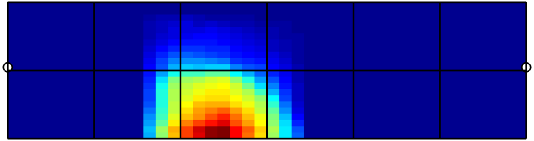
Solution procedure



Example 1: Coupling strength



Permebaility

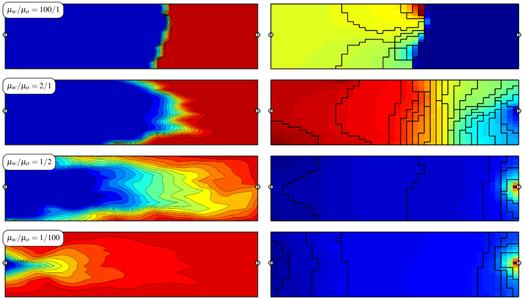


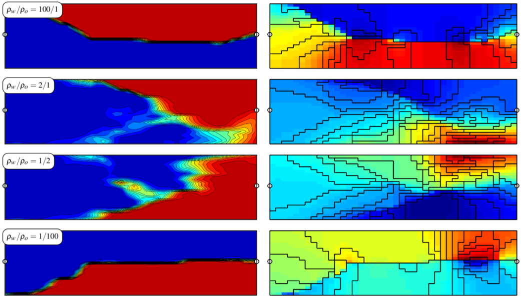
General rectilinear partition

- Incompressible two-phase model with single-component aqueous and liquid phase
- Geological model: 42×22 subset of Layer 10 of SPE 10 Model 2[1]
- Initially filled with liquid phase, inject 1 PV of aqueous phase
- General basis: Rectilinear 6×2 with MsRSB basis functions
- Look at effect of dynamic basis functions as we vary viscosity and density ratios

[1]Christie and Blunt, 2001

Example 1: Coupling strength

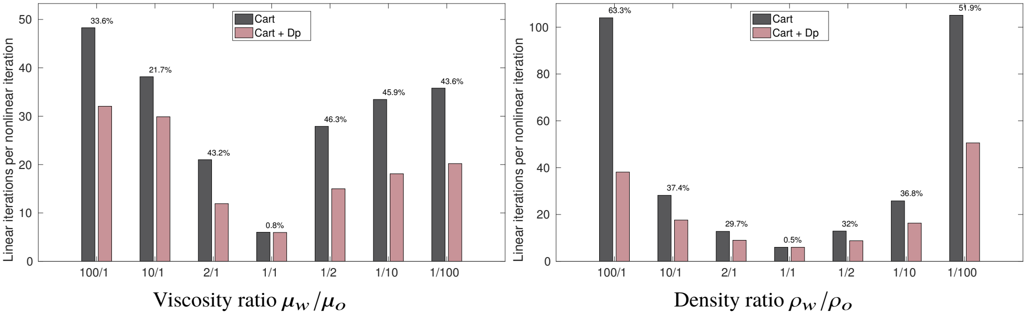




- Varying viscosity ratio μ_w/μ_o
- No capillary or gravity effects included

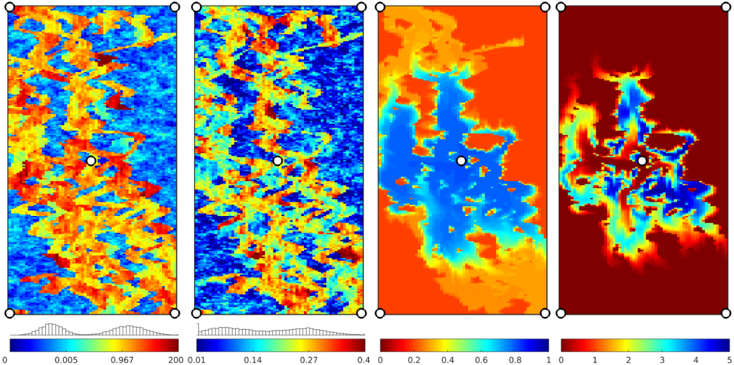
- Varying density ratio ρ_w/ρ_o
- No capillary effects, equal viscosities

Example 1: Coupling strength



- Average number of linear iterations per nonlinear iteration
- Adding dynamic basis is beneficial in all cases except unit density and viscosity ratios
- Difference of 0.8% and 0.5% comes from the extra smoothing iteration

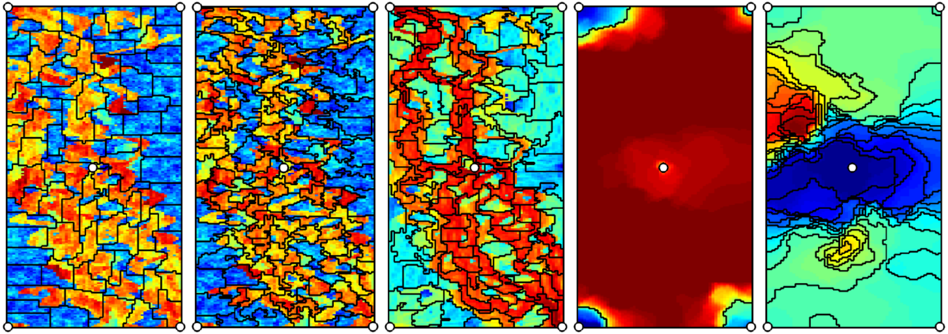
Example 2: Polymer injection



(a) Permeability (100 md) (b) Porosity (c) S_w, 1200 days (d) c, 1200 days (kg/m^3)

- Two-phase, three-component: Polymer injection in Layer 52 of SPE 10 Model 2
- Layer consist of high-permeable fluvial channels
- Injection over 2000 days, polymer slug injected from 400 to 800 days

Example 2: Polymer injection

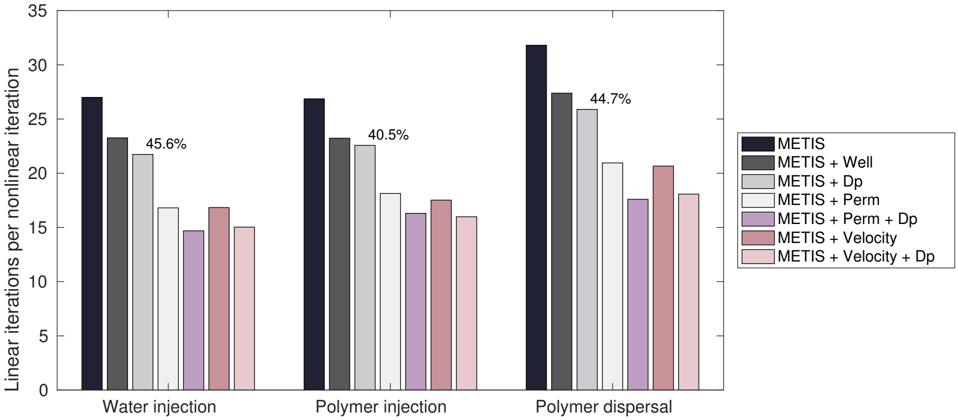


(a) METIS (80) (b) Permeability (150) (c) Velocity (136) (d) Wells (6) (e) Δp (117)

- General partition: MsRSB, partition generated using METIS[1]
- Static partitions based on permeability and velocity using agglomeration of grid cells[2]
- Well partition with specialized basis functions[3]
- Dynamic partition based on Δp from previous timestep

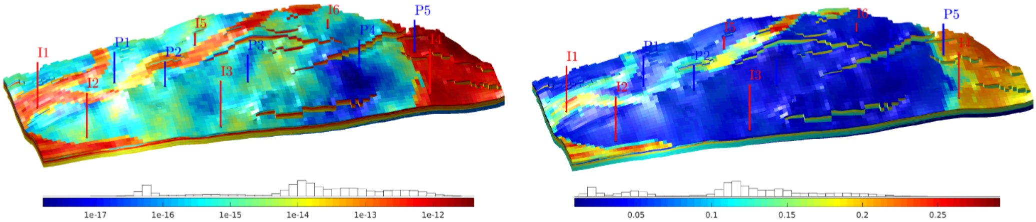
[1]Karypis and Kumar, 1998 [2]Hauge et al., 2012 [3]Lie et al., 2017

Example 2: Polymer injection



- Average number of linear iterations per nonlinear iteration
- Beneficial with partitions honoring channeled structure
- Significant reduction by adding well partition (only 6 coarse cells)

Example 3: Field model (SAIGUP)

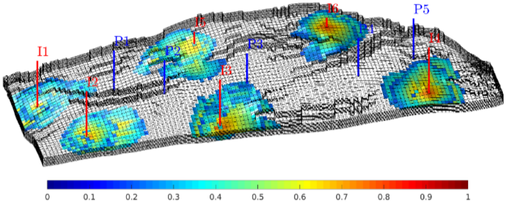


(a) Permeability (md) (b) Porosity

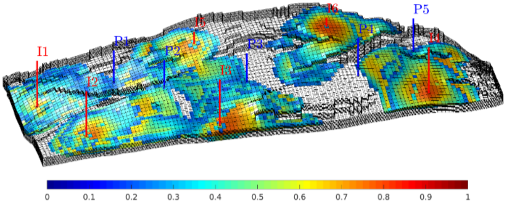
- Shallow-marine oil reservoir, modeled in the SAIGUP study[1]
- Spans lateral area of $\sim 9 \times 3$ km^2, $40 \times 120 \times 20$ corner-point grid, several major faults
- Simulate WAG injection using four-phase four-pseudo-component model
  - 0.8 PV of water + 0.8 PV of solvent gas/water cycles + 0.8 PV of water

[1]Manzocchi et al., 2008

Example 3: Field model (SAIGUP)



(a) Water saturation after initial water injection

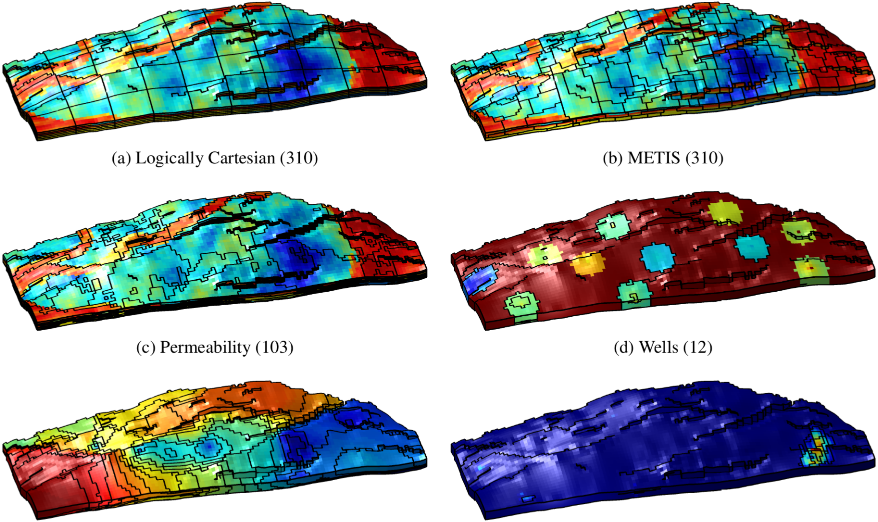


(b) Water saturation after WAG + final water injection

- Shallow-marine oil reservoir, modeled in the SAIGUP study[1]
- Spans lateral area of $\sim 9 \times 3$ km^2, $40 \times 120 \times 20$ corner-point grid, several major faults
- Simulate WAG injection using four-phase four-pseudo-component model
  - 0.8 PV of water + 0.8 PV of solvent gas/water cycles + 0.8 PV of water

[1]Manzocchi et al., 2008

Example 3: Field model (SAIGUP)



(a) Logically Cartesian (310)

(b) METIS (310)

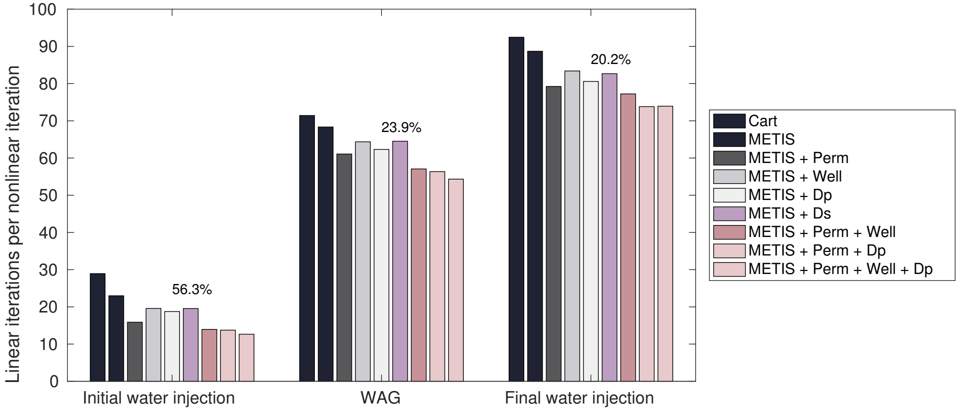
(c) Permeability (103)

(d) Wells (12)

(e) Δp (78)

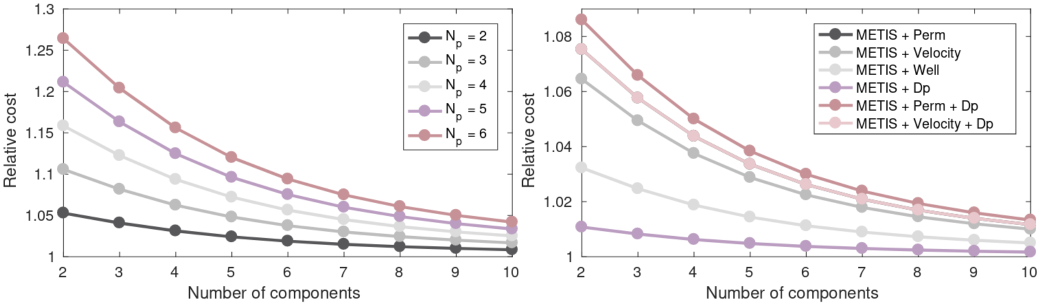
(f) S_s (179)

Example 3: Field model (SAIGUP)



- Significant reduction during initial water injection
- With solvent: Strong coupling between solvent gas saturation and reservoir fluid mobility $\longrightarrow$ two-stage CPR preconditioner not as effective.

Computational efficiency



(a) Theoretical: equal overlap $b = d = 15$ (b) Actual numbers from Example 2 (Layer 52, SPE10)

- Cost of using N_p multiscale operators over using just one:

$$c(N_p) = N\left[N_p(4d + 2b + 4) + N_c(d + 1) + N_c(2dN_c + 1)\right]$$

- Relatively small cost compared to solving the full system (max 10 – 15% for realistic scenarios)

Conclusion

- Feature-adapted multiscale method used as pressure solver in CPR preconditioner for a fully-implicit simulator
- Combination of general uniform, static geomodel, and dynamic partitions adapting to pressure update/saturation
  - Honors geological features, near-well regions, dynamic couplings
- Significant reduction in number of linear iterations observed
  - 10 – 60% reduction compared to using CPR preconditioner with a single multiscale operator
- Experiments indicate that it is beneficial with
  - static partitions honoring large permeability contrasts and/or near-well regions;
  - partitions adapting to pressure updates whenever these are located along propagating fluid fronts and/or discontinuous across fluid phase interfaces

Acknowledgements

All simulations have been done using the
MATLAB Reservoir Simulation Toolbox (MRST)

`mrst.no`

The authors were supported by the Research Council of Norway under grant no. 244361, and VISTA, which is a basic research programme funded by Equinor and conducted in close collaboration with The Norwegian Academy of Science and Letters

Extra – Discretization

$$A_\beta^i = \sum_\alpha A_{\alpha,\beta}^i, \quad A_{\alpha,\beta}^i = \left(\phi\rho_\alpha S_\alpha X_{\alpha\beta}\right)_i^{n+1} - \left(\phi\rho_\alpha S_\alpha X_{\alpha\beta}\right)_i^n \qquad \text{(Accumulation)}$$

$$G_\beta^{i,j} = \sum_\alpha G_{\alpha,\beta}^{i,j} \quad G_{\alpha,\beta}^{i,j} = \frac{\Delta t}{|\Omega_i|}|\Gamma_{ij}|\left(\rho_\alpha X_{\alpha\beta}\vec{v}_\alpha \cdot \vec{n}\right)_{ij}^{n+1} \qquad \text{(Flux)}$$

$$Q_\beta^i = \frac{\Delta t}{|\Omega_i|}(q_\beta)_i^{n+1} \qquad \text{(Sources/sinks)}$$

Extra – Decoupling

- Decoupling: find weights $\mathbf{w}_\beta$ for $k = 2\ldots, N_c$ so that

$$\text{IMPES: } \sum_{\beta=1}^{N_c} \mathbf{W}_\beta \frac{\partial \mathbf{A}_\beta}{\partial \mathbf{x}_k} = \mathbf{0}, \quad \text{quasi-IMPES: } \sum_{\beta=1}^{N_c} \mathbf{W}_\beta \operatorname{diag}\left(\frac{\partial \mathbf{F}_\beta}{\partial \mathbf{x}_k}\right) = \mathbf{0}$$

1. Solve for $\mathbf{w}_1 \ldots \mathbf{w}_\beta$

$$\begin{bmatrix} \mathbf{M}_{2,1}^T & \ldots & \mathbf{M}_{2,N_c}^T \\ \vdots & \ddots & \vdots \\ \mathbf{M}_{N_c,1}^T & \ldots & \mathbf{M}_{N_c,N_c}^T \\ \mathbf{I} & \ldots & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \vdots \\ \mathbf{w}_\beta \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}, \quad \mathbf{M}_{k,\beta} = \begin{cases} \frac{\partial \mathbf{A}_\beta}{\partial \mathbf{x}_k} & \text{IMPES} \\ \operatorname{diag}\left(\frac{\partial \mathbf{F}_\beta}{\partial \mathbf{x}_k}\right) & \text{quasi-IMPES} \end{cases}$$

2. Premultiply $\mathbf{J}$ and $\mathbf{F}$ by

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & & \ldots & \mathbf{W}_{N_c} \\ \mathbf{0} & \mathbf{I} & \ldots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & & \ldots & \mathbf{I} \end{bmatrix}, \quad \text{where} \quad \mathbf{W}_\beta = \operatorname{diag}(\mathbf{w}_\beta)$$

Extra – Computational efficiency

- Cost of using N_p multiscale operators (assuming all partitions are equal)

$$N_p \mathcal{O}\Big(\underbrace{(Nd + N + 2Nd + N)}_{\text{Smooth}} + \underbrace{(Nd + N + bN)}_{\text{Restrict}} + \underbrace{M^p}_{\text{Solve for } \mathbf{p}_c} + \underbrace{(bN + N)}_{\text{Prolongate and update}}\Big)$$

  - d: Upper bound on number of nonzero elements in rows of $\mathbf{A}$, $d \ll N$
  - M: Number of coarse cells in partition
  - b: Maximum number of basis functions with support in a single cell for a Galerkin restriction, $1 < b < M$

- Cost of using N_p multiscale operators over using just one:

$$c(N_p) \approx N\big[N_p(4d + 2b + 4) + N_c(d + 1) + N_c(2dN_c + 1)\big].$$