Two-scale numerical modelling of flow in porous media with fine-scale heterogeneity

Elliot Carr
Mathematical Sciences
Queensland University of Technology

Maths Seminar Series
5 June 2015

Part I:

Unsaturated water flow in a two-dimensional binary medium

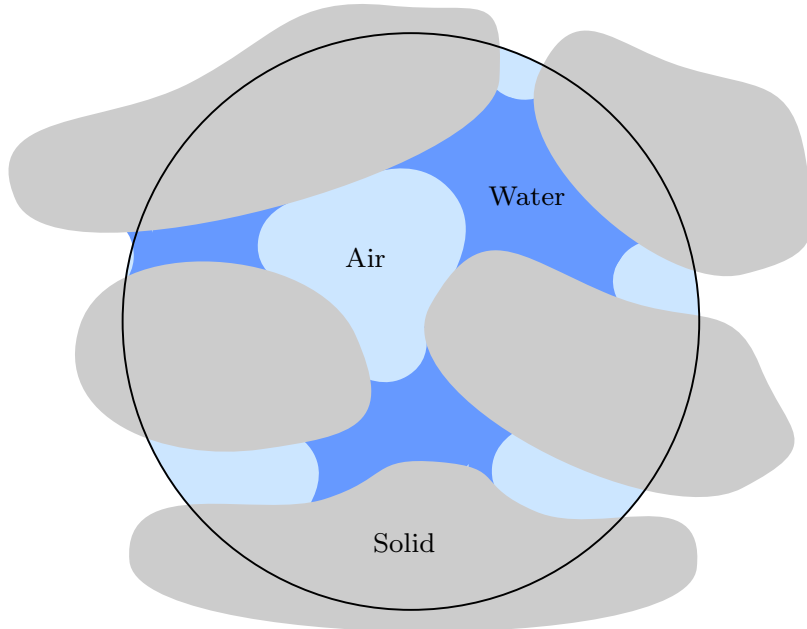
Joint work with Ian Turner (QUT) and Patrick Perré (ECP)

Unsaturated water flow

- Immiscible two-phase (air and water) flow
- Darcy's Law (phase α):

$$\boldsymbol{q}_\alpha = -\frac{k_{\mathrm{r},\alpha}}{\mu_\alpha} K \left(\nabla p_\alpha - \rho_\alpha \boldsymbol{g}\right)$$

- Capillary pressure $p_c = p_a - p_w$



- Mass conservation (phase α):

$$\frac{\partial \left(\rho_\alpha \phi S_\alpha\right)}{\partial t} + \nabla \cdot \left(\rho_\alpha \boldsymbol{q}_\alpha\right) = 0$$

where $S_w + S_a = 1$

Richards' equation

- Model assumptions:
  (i) air phase is at a constant and atmospheric pressure
  (ii) water phase is incompressible and of constant density
- Single equation for the water saturation:

$$\frac{\partial \theta}{\partial t}(h) + \nabla \cdot (-K(h)(\nabla h + \nabla y)) = 0$$

where $h = p_c/(\rho_w g)$ is the capillary pressure head, $\theta = \phi S_w$ is the moisture content and K is the hydraulic conductivity.

- Closure relationships [**van Genuchten (1980)**]:

$$\theta(h) = \theta_{\text{res}} + (\theta_{\text{sat}} - \theta_{\text{res}}) S_e(h)$$

$$K(h) = K_{\text{sat}} \sqrt{S_e} \left(1 - \left(1 - S_e^{1/m}\right)^m\right)^2$$

$$S_e(h) = (1 + (-\alpha h)^n)^{-m}$$

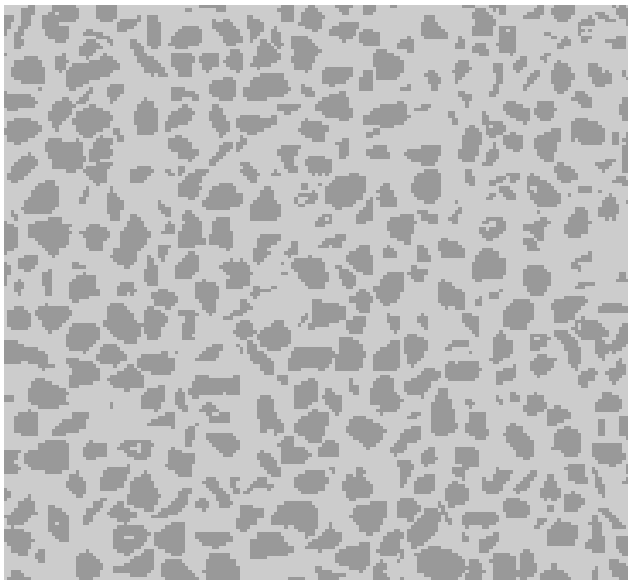
with hydraulic parameters θ_{res}, θ_{sat}, α, n and m.

Fine-scale model

- **Binary medium**: Domain comprised of two sub-domains Ω_a (connected) and Ω_b (inclusions):

$$\theta(h) = \begin{cases} \theta_a(h) & \text{in } \Omega_a \\ \theta_b(h) & \text{in } \Omega_b \end{cases}$$

$$K(h) = \begin{cases} K_a(h) & \text{in } \Omega_a \\ K_b(h) & \text{in } \Omega_b \end{cases}$$



Heterogeneous domain

Ω_a Ω_b

- Richards' equation in a binary medium ($i = a, b$):

$$\frac{\partial \theta_i}{\partial t}(h_i) + \nabla \cdot \left(-K_i(h_i)\left(\nabla h_i + \nabla y\right)\right) = 0$$

- Computational cost of direct numerical simulation is prohibitively expensive when the domain exhibits small-scale heterogeneity.

Two-scale computational model

Macroscopic equation: $h_a(x,t)$ $(x \in \Omega)$

$$\frac{\partial \theta_{\text{eff}}}{\partial t}(h_a) + \nabla \cdot (\mathbf{K}_{\text{eff}}(h_a)\,(\nabla h_a + \nabla y)) = S$$

Microscopic equation: $h_{b,i}(x,t)$ $(x \in \mathcal{C}_{i,b})$

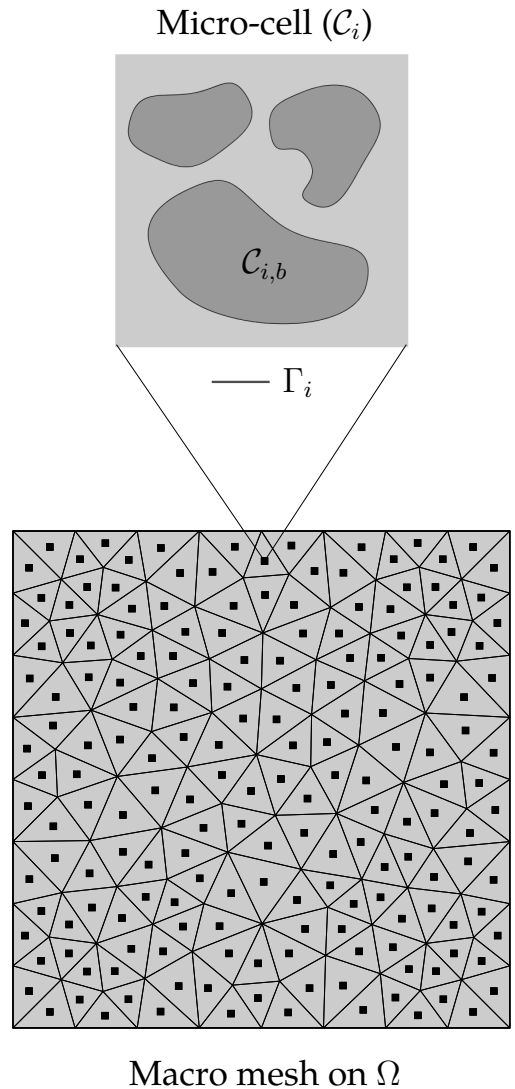
$$\frac{\partial \theta_b}{\partial t}(h_{b,i}) + \nabla \cdot (K_b(h_{b,i})\,(\nabla h_{b,i} + \nabla y)) = 0$$

Coupling between scales:

(i) Microscopic boundary condition:
$h_{b,i}(x,t) = h_a(x,t)$ $(x \in \Gamma_i)$

(ii) Source term S represents the exchange of water between the two sub-domains and is recovered numerically from the micro-cell problems

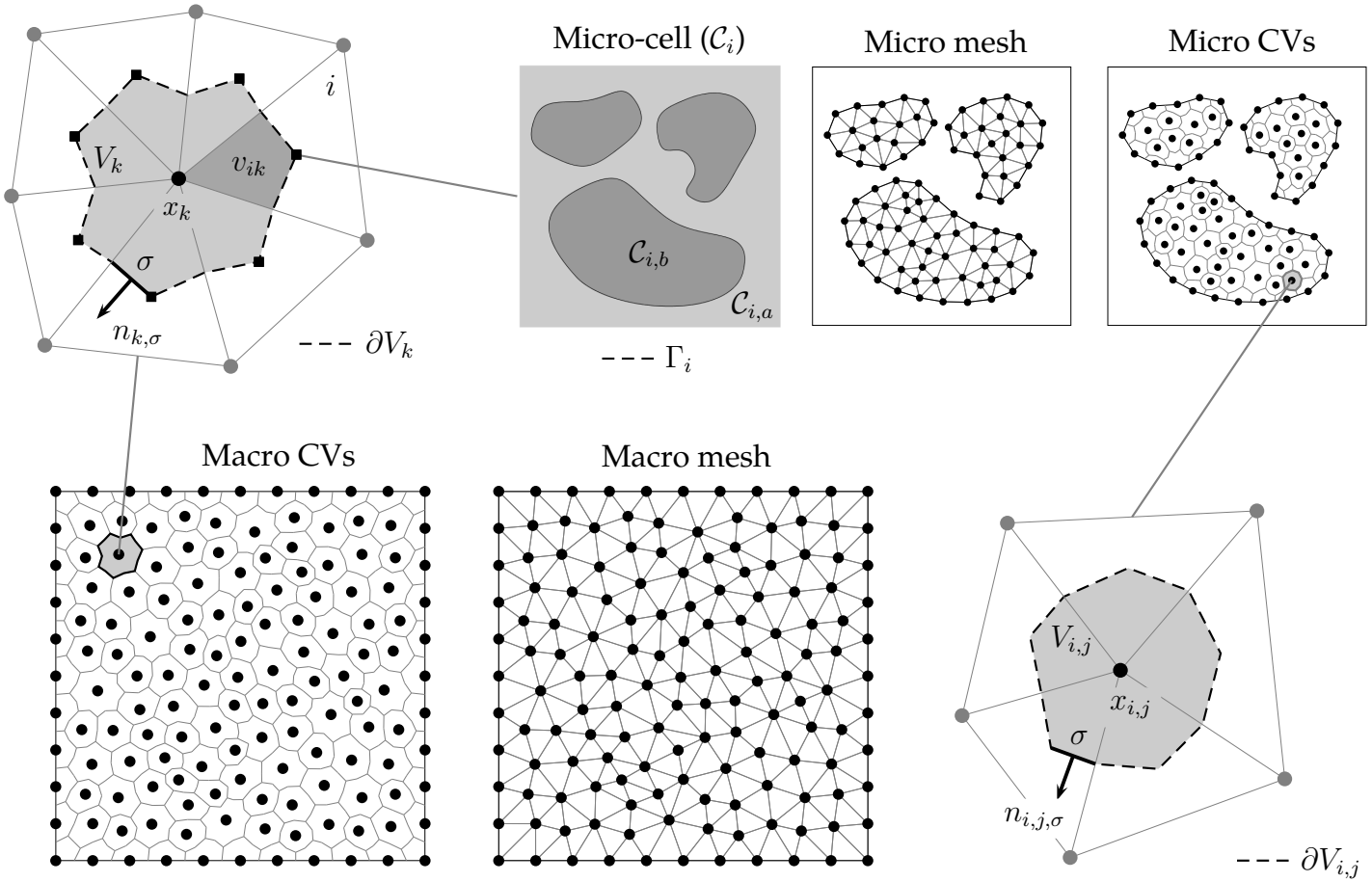
Szymkiewicz and Lewandowska (2006)
Carr and Turner (2014), Carr et al. (2015)



Macro mesh on Ω

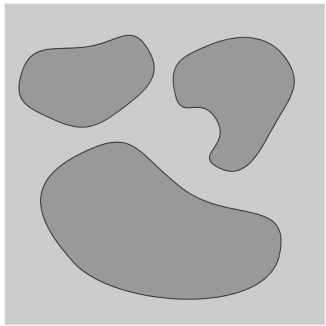
Spatial discretisation

Control Volume Finite Element discretisation [Carr et al. (2015)]



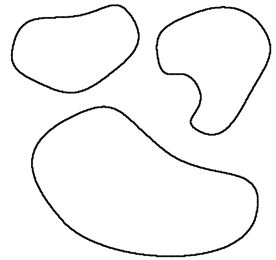
Imaged-based meshing of micro-cells

Image-based mesh generation using GMSH [Carr and Turner (2014)]

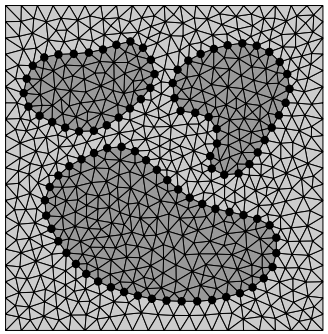


Bitmap image

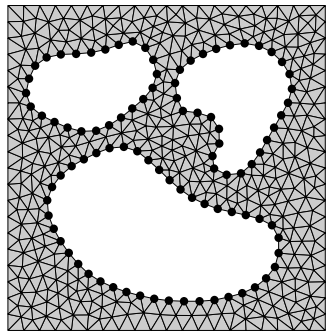
Binary image



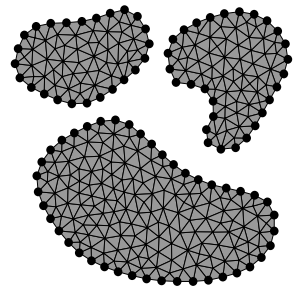
Edge detection



Full micro-cell ($\mathcal{C}_i$)



$\mathcal{C}_{i,a}$ only



$\mathcal{C}_{i,b}$ only

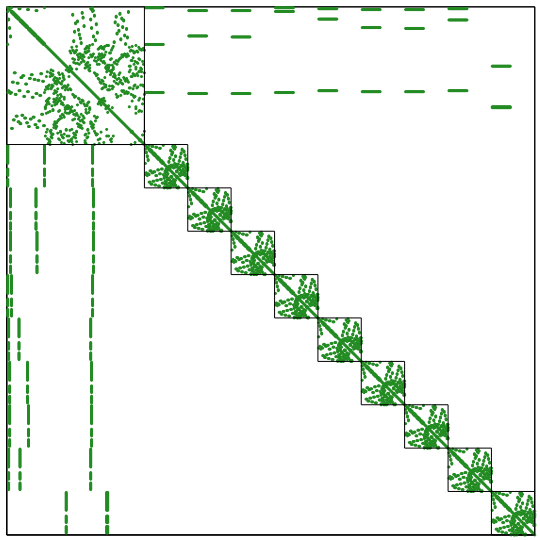
Semi-discrete system of ODEs

- Spatial discretisation can be expressed in the form [Carr et al. (2015)]

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{g}(\boldsymbol{u}), \quad \boldsymbol{u}(0) = \boldsymbol{u}_0 \in \mathbb{R}^N,$$

  where $N =$ n_macro_nodes + n_macro_elements $\times$ n_micro_nodes.

- **Example:** Sparsity pattern of the Jacobian matrix of $\boldsymbol{g}(\boldsymbol{u})$ (zoomed in)



Time stepping: exponential integrator

Exponential Rosenbrock-Euler Method [Carr et al. (2011, 2013)]:

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{g}(\boldsymbol{u}) \quad \xrightarrow{\text{Linearise}} \quad \frac{d\boldsymbol{u}}{dt} = \boldsymbol{g}_n + \boldsymbol{J}_n(\boldsymbol{u} - \boldsymbol{u}_n)$$

where $\boldsymbol{g}_n = \boldsymbol{g}(\boldsymbol{u}_n)$ and $\boldsymbol{J}_n = \boldsymbol{J}(\boldsymbol{u}_n)$, and solve exactly over a single time step:

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \tau_n \varphi(\tau_n \boldsymbol{J}_n)\boldsymbol{g}_n$$

where $\varphi(\boldsymbol{A}) = \boldsymbol{A}^{-1}(e^{\boldsymbol{A}} - \boldsymbol{I})$.

- Explicit scheme
- Krylov subspace methods for computing $\varphi(\tau_n \boldsymbol{J}_n)\boldsymbol{g}_n$ converge rapidly without preconditioning, and require only matrix-vector products with $\boldsymbol{J}_n$:

$$\boldsymbol{J}_n \boldsymbol{v} \approx \frac{\boldsymbol{g}(\boldsymbol{u}_n + \varepsilon \|\boldsymbol{v}\|_2 \boldsymbol{v}) - \boldsymbol{g}(\boldsymbol{u}_n)}{\varepsilon \|\boldsymbol{v}\|_2}, \quad \varepsilon \approx 10^{-8}$$

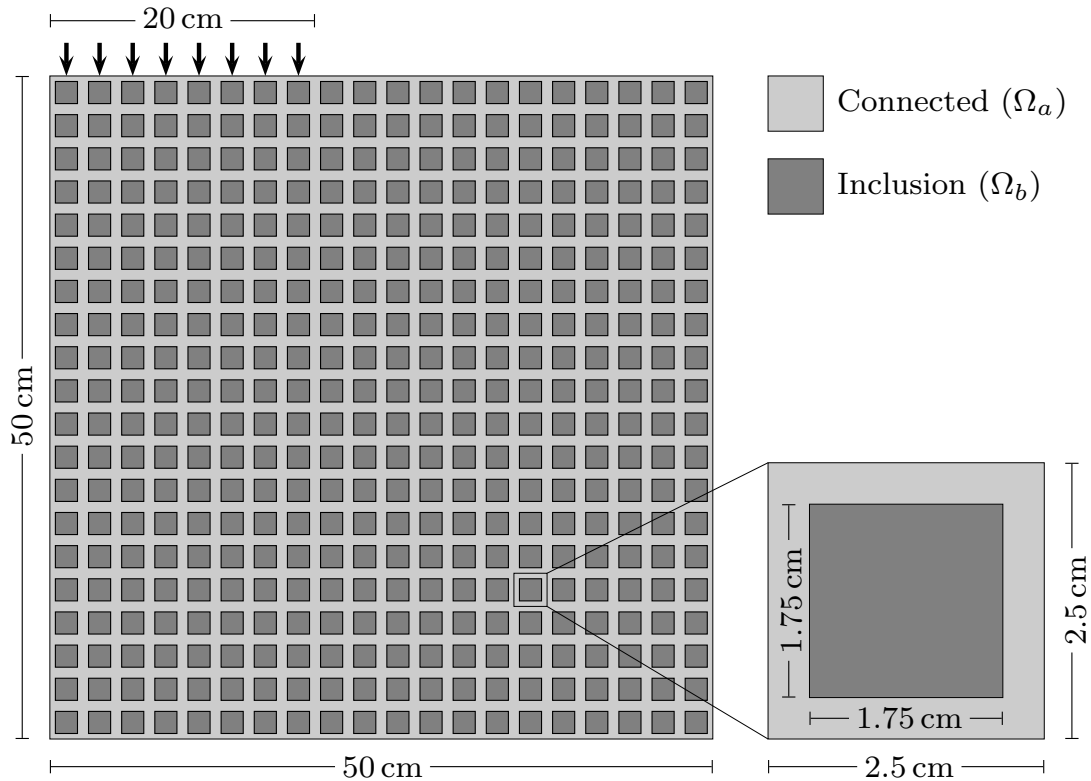
Code implementation details

- Both fine-scale and two-scale codes developed in C++
- Linear algebra operations performed using BLAS and LAPACK libraries:
  (a) Intel MKL (HPC Platform)
  (b) Accelerate framework (Macbook Pro)
- ODE right-hand side function $\boldsymbol{g}(\boldsymbol{u})$ implemented in parallel using OpenMP

```
#include <omp.h>
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel for
for (k=0; k<n_macro_elements; k++)
{
  ...
}
```

- Code accommodates both triangular and quadrilateral elements at both scales

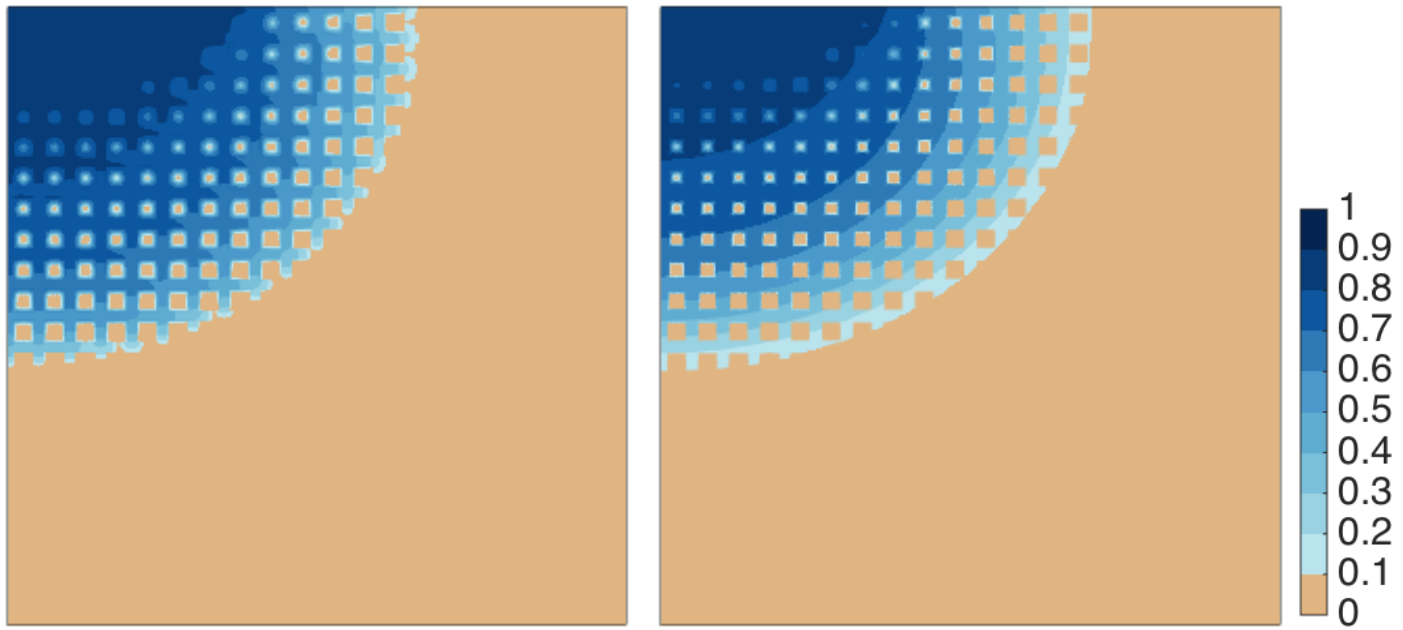
Model validation



Fine-scale: $208,676$ triangular elements and $104,859$ nodes [$N = 104,859$]
Two-scale: 20×20 grid and 441 nodes (macro mesh)
264 triangular elements and 153 nodes (micro mesh) [$N = 61,641$]

Model validation

Saturation contours after 25 hours:

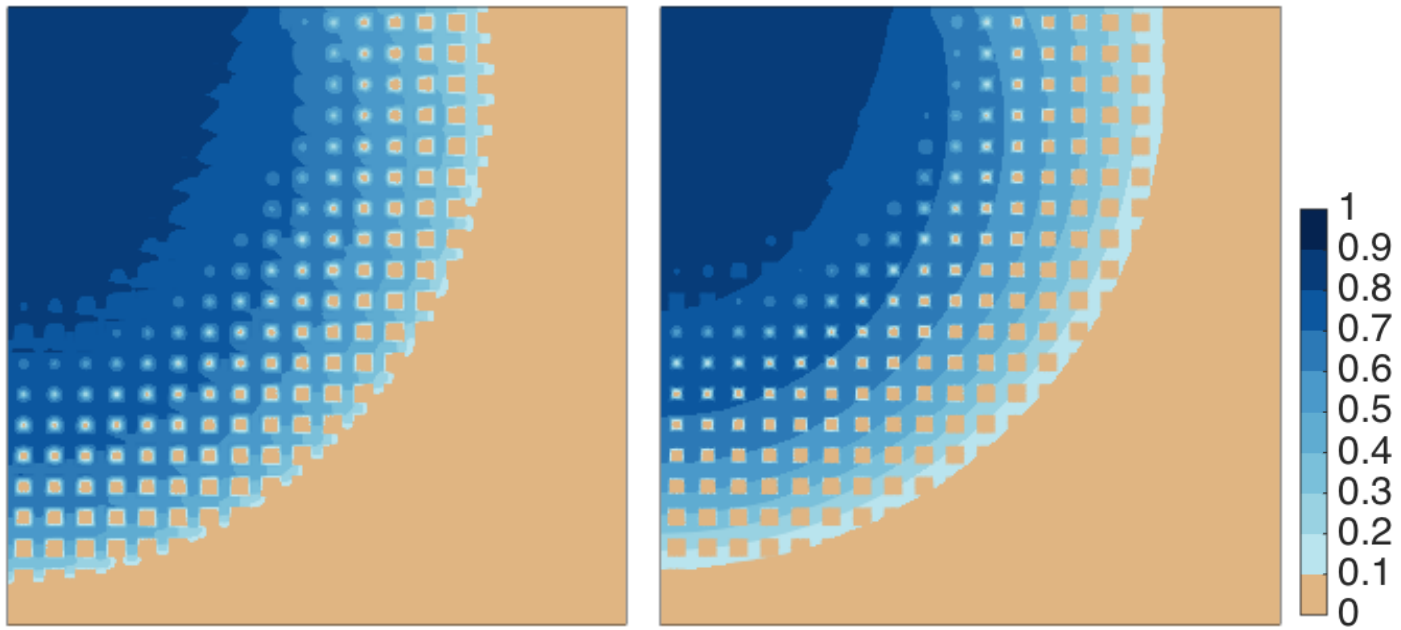


Fine-scale model
(Runtime: 2 hours, 32 mins)

Two-scale model
(Runtime: 1 min)

Model validation

Saturation contours after 50 hours:

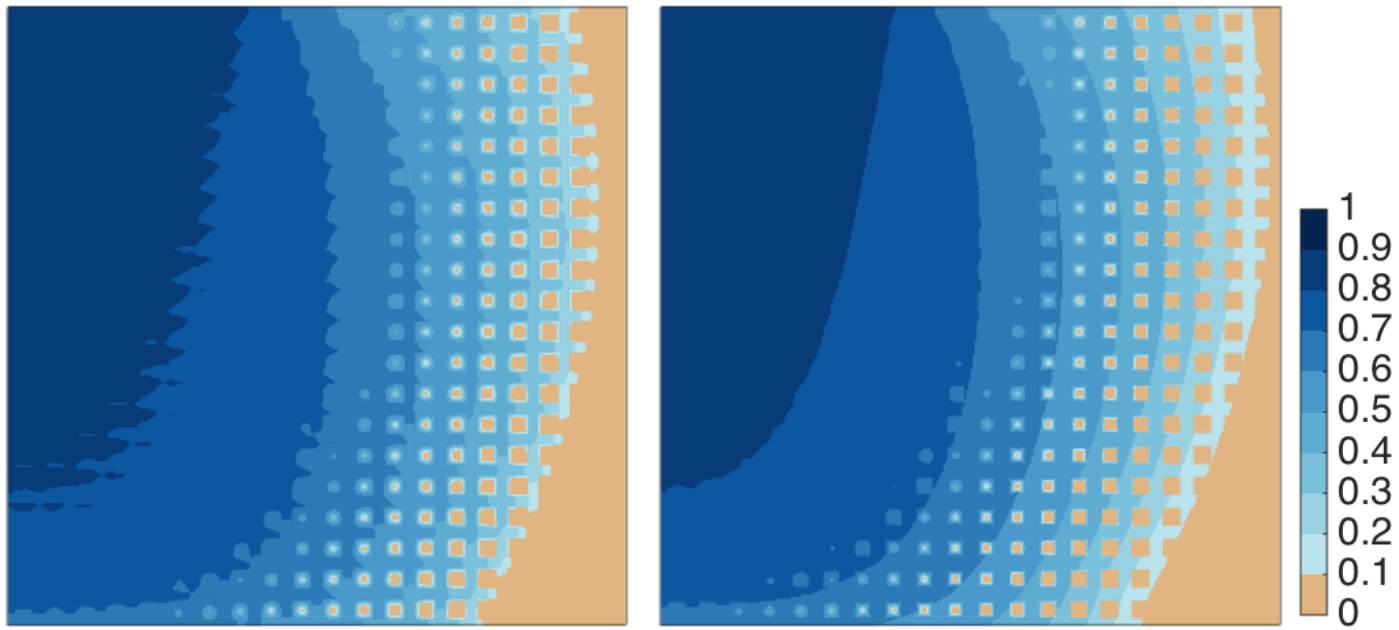


Fine-scale model
(Runtime: 2 hours, 32 mins)

Two-scale model
(Runtime: 1 min)

Model validation

Saturation contours after 100 hours:



Fine-scale model
(Runtime: 2 hours, 32 mins)

Two-scale model
(Runtime: 1 min)

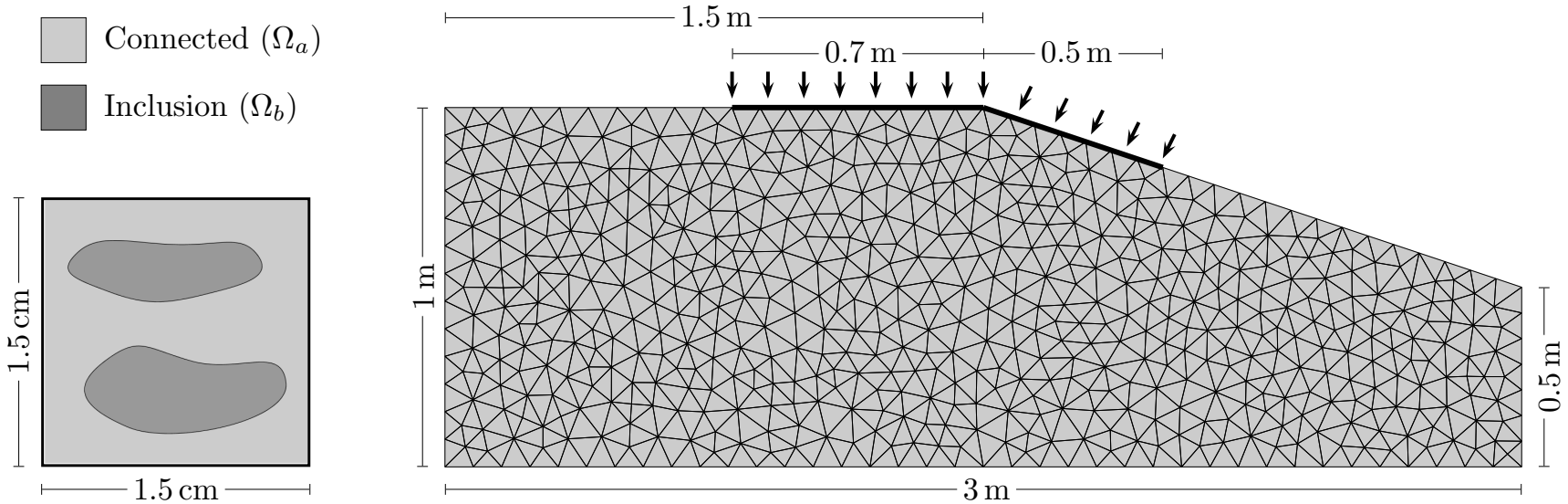
Model validation

Saturation contours after 400 hours:

Fine-scale model
(Runtime: 2 hours, 32 mins)

Two-scale model
(Runtime: 1 min)

Application to a heterogeneous aquifer



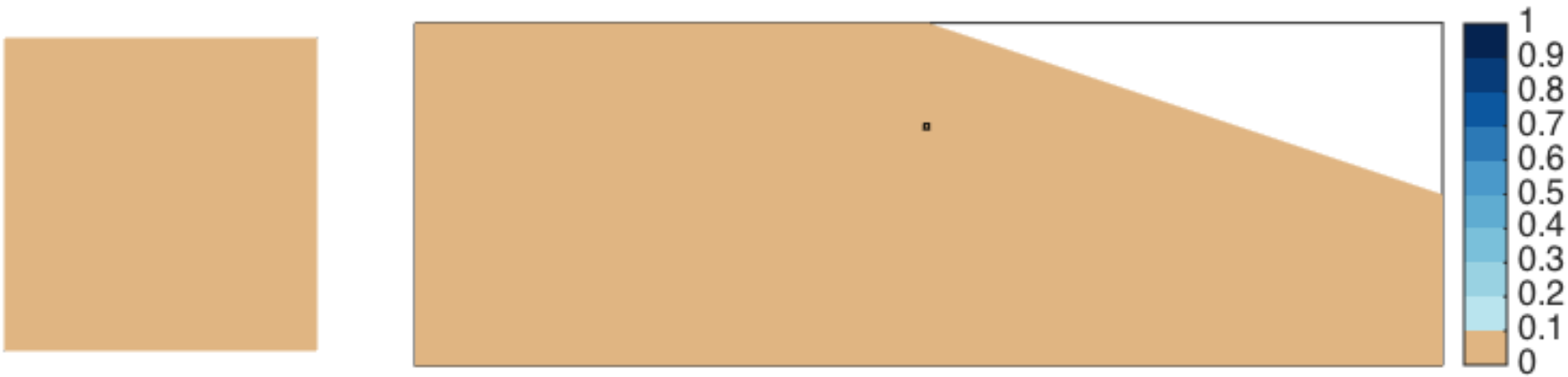
Macro mesh: 1116 elements, 608 nodes

Micro meshes: 583 elements, 339 nodes

Total number of unknowns: $608 + 1116 \times 339 = 378{,}932$

Application to a heterogeneous aquifer

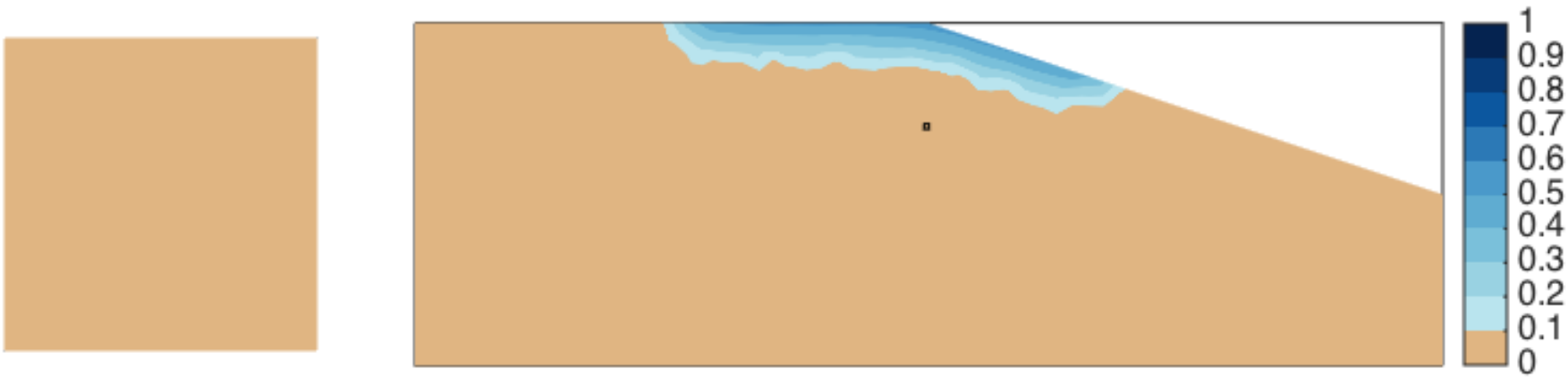
Saturation contours after 0 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

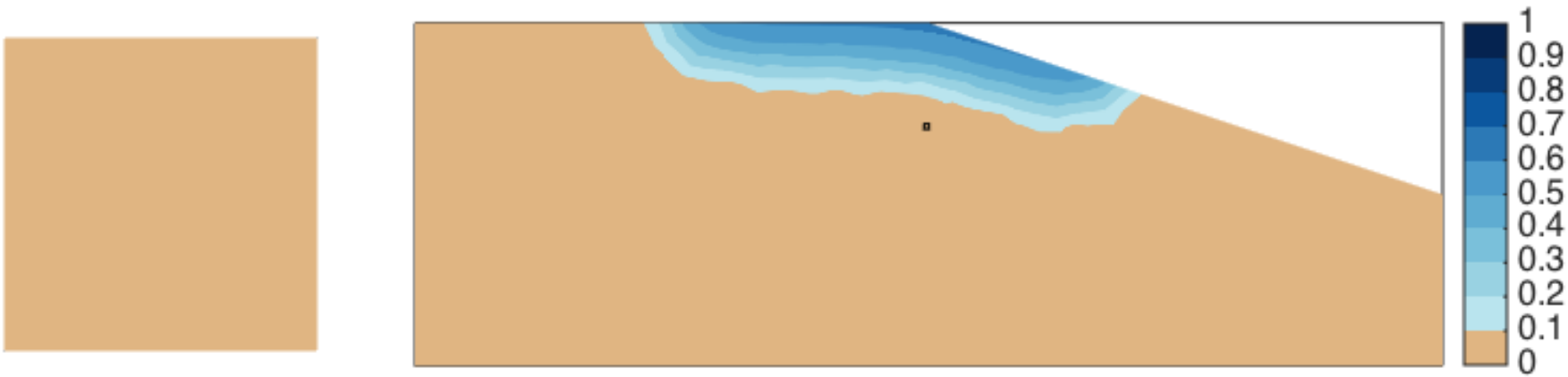
Saturation contours after 1 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

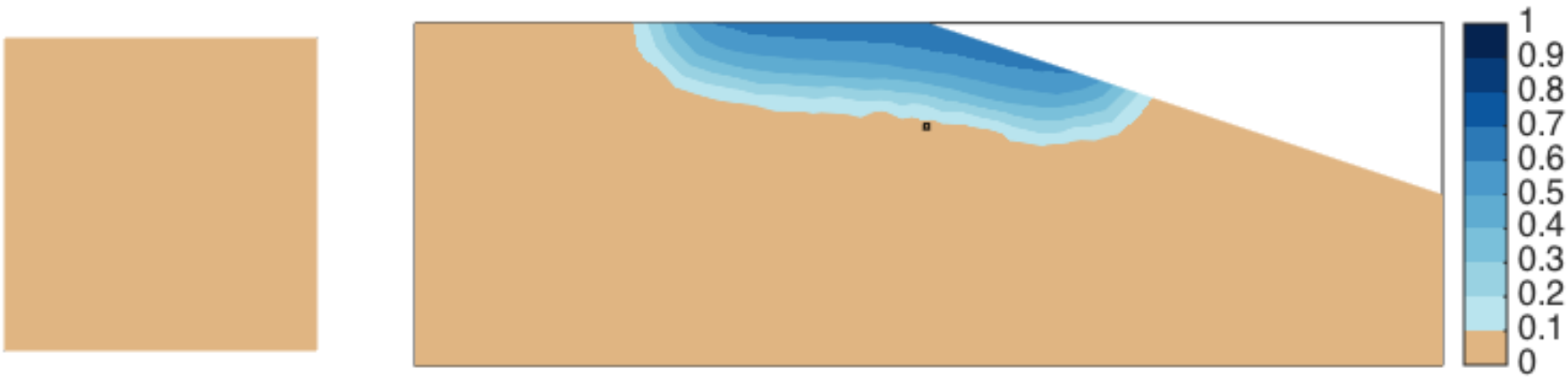
Saturation contours after 2 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

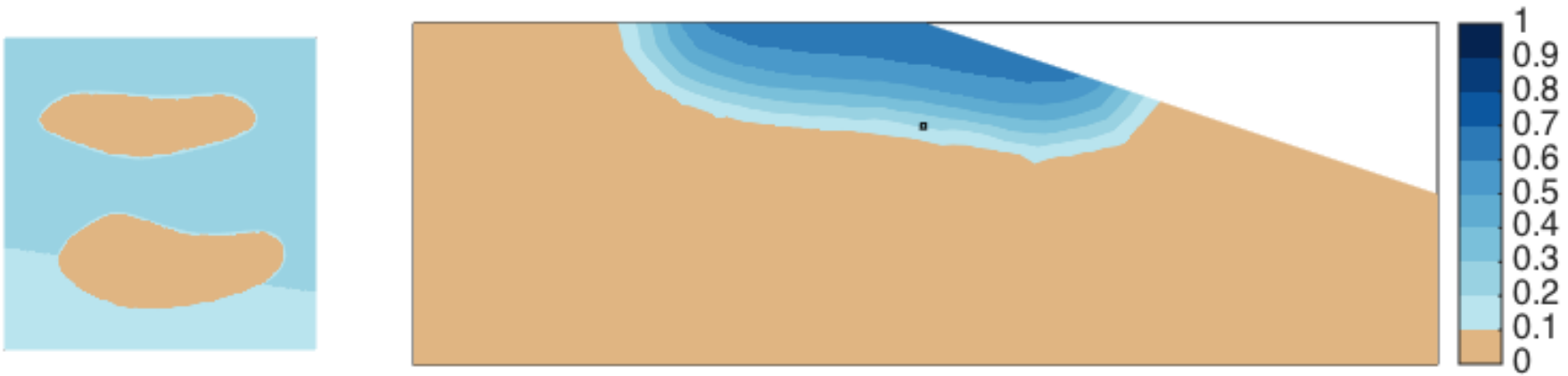
Saturation contours after 3 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

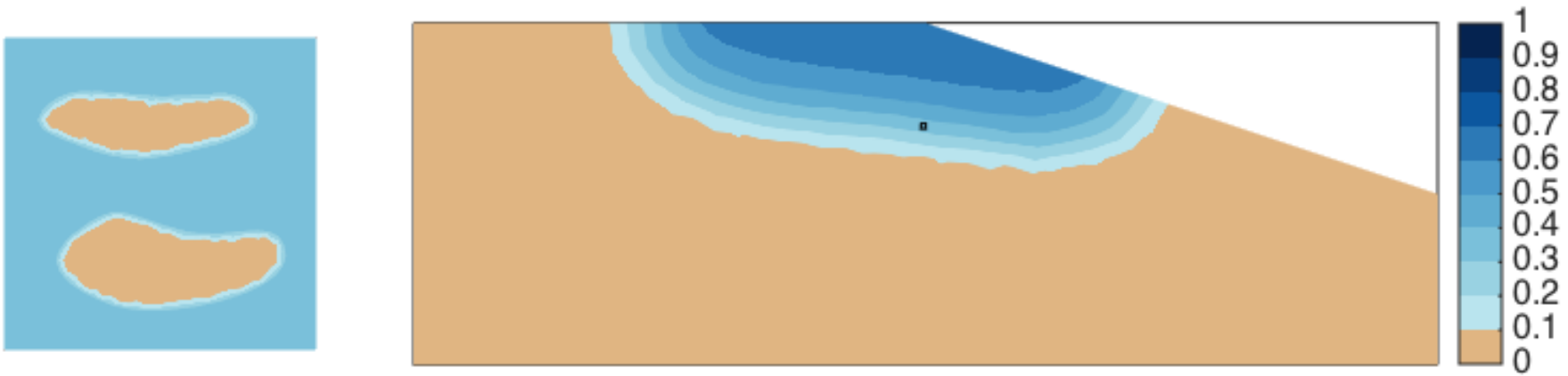
Saturation contours after 4 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

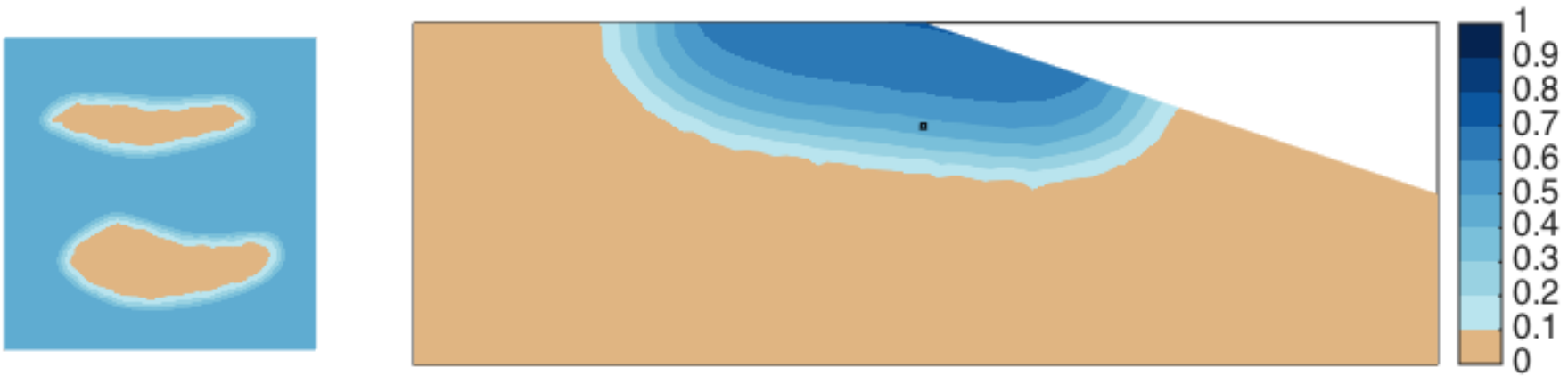
Saturation contours after 5 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

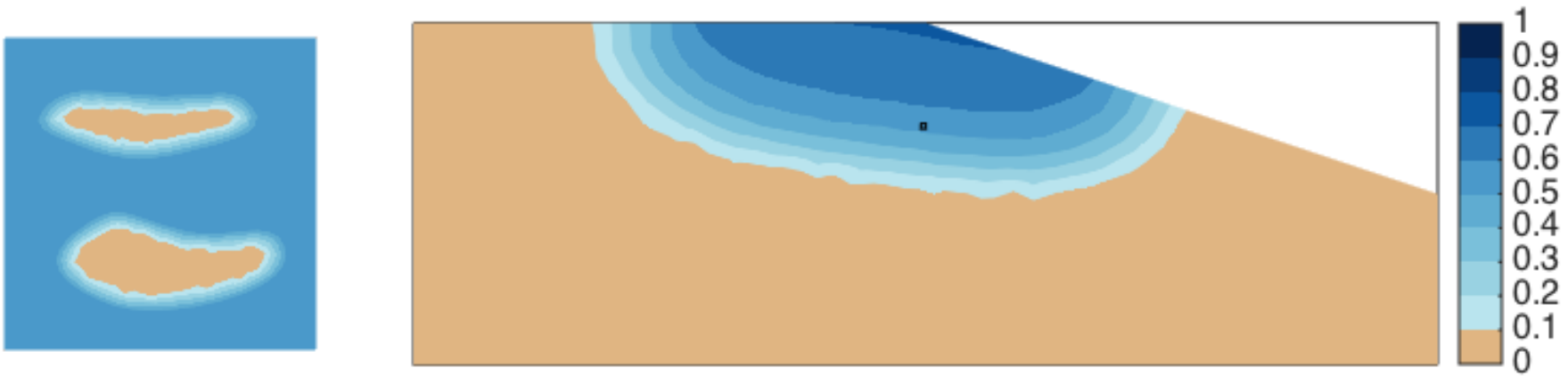
Saturation contours after 6 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

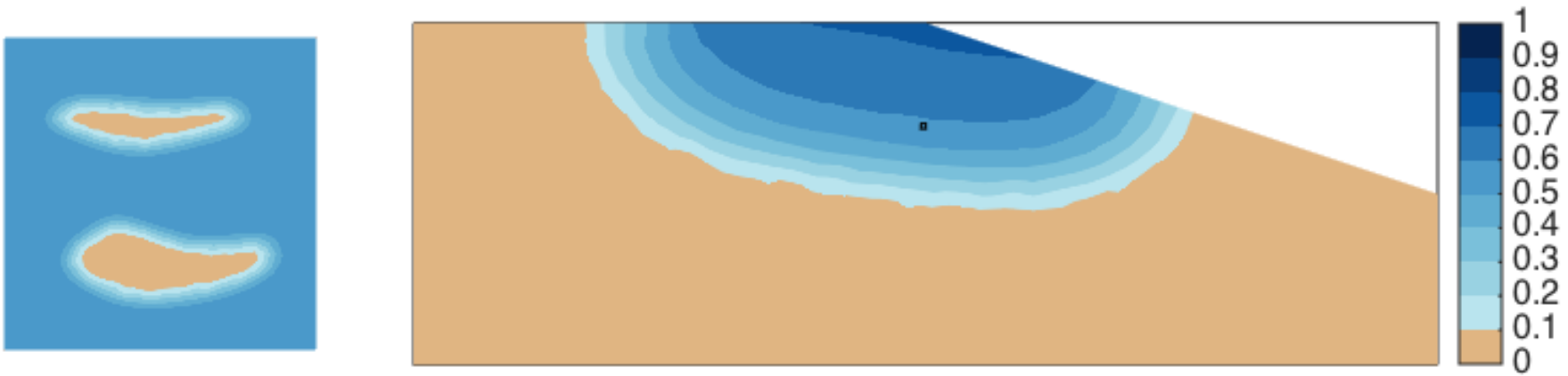
Saturation contours after 7 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

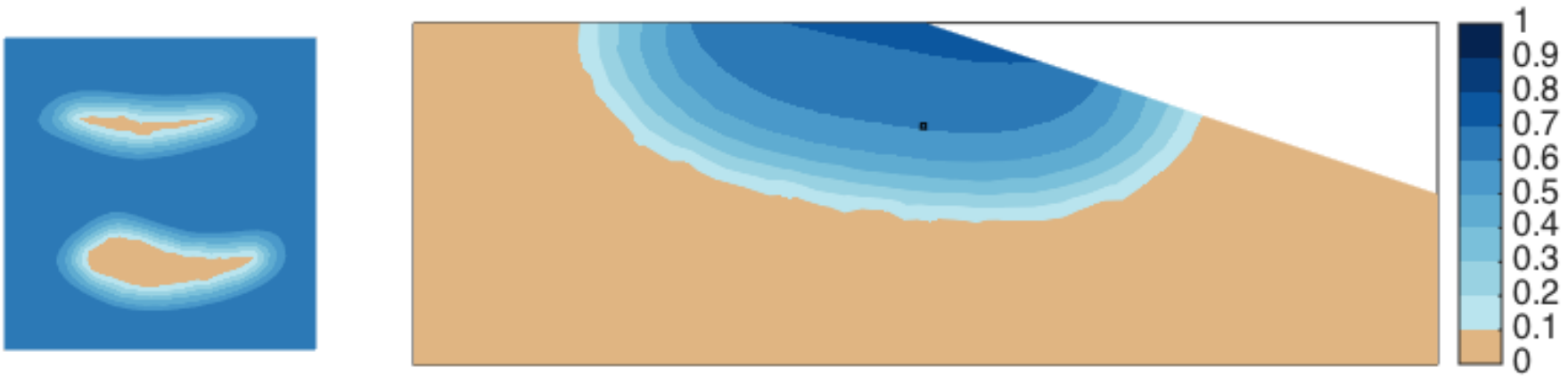
Saturation contours after 8 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

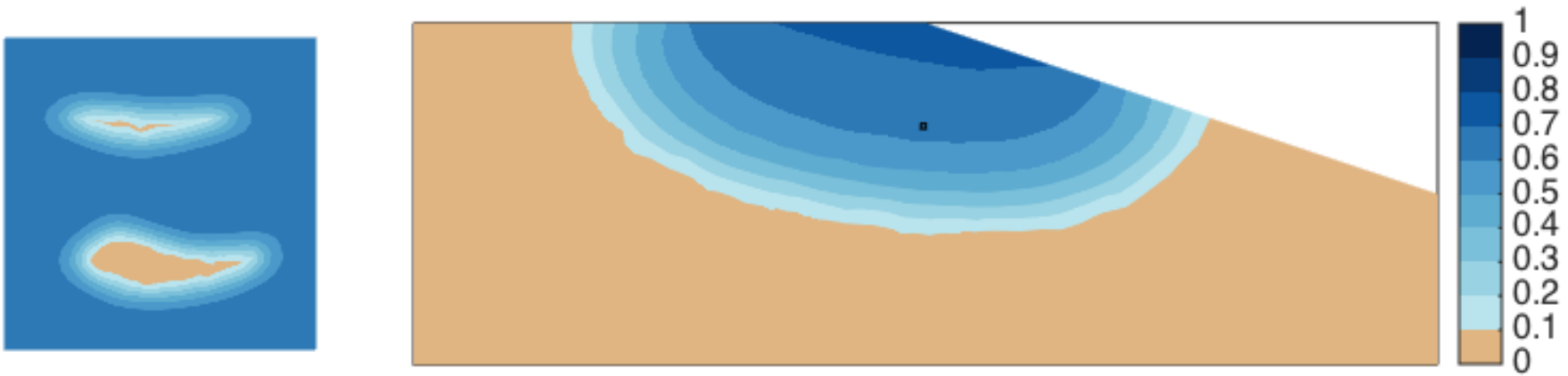
Saturation contours after 9 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

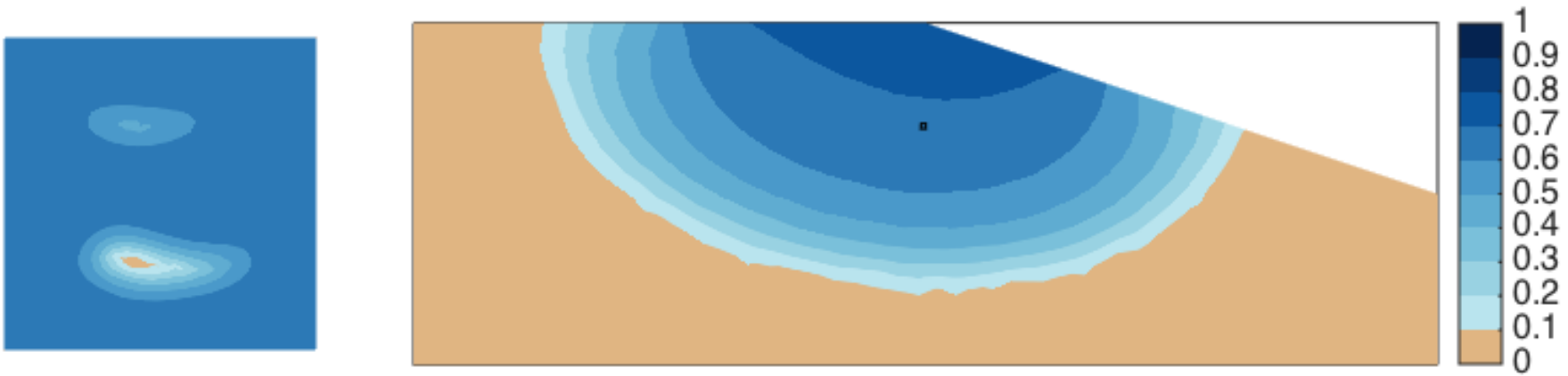
Saturation contours after 10 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

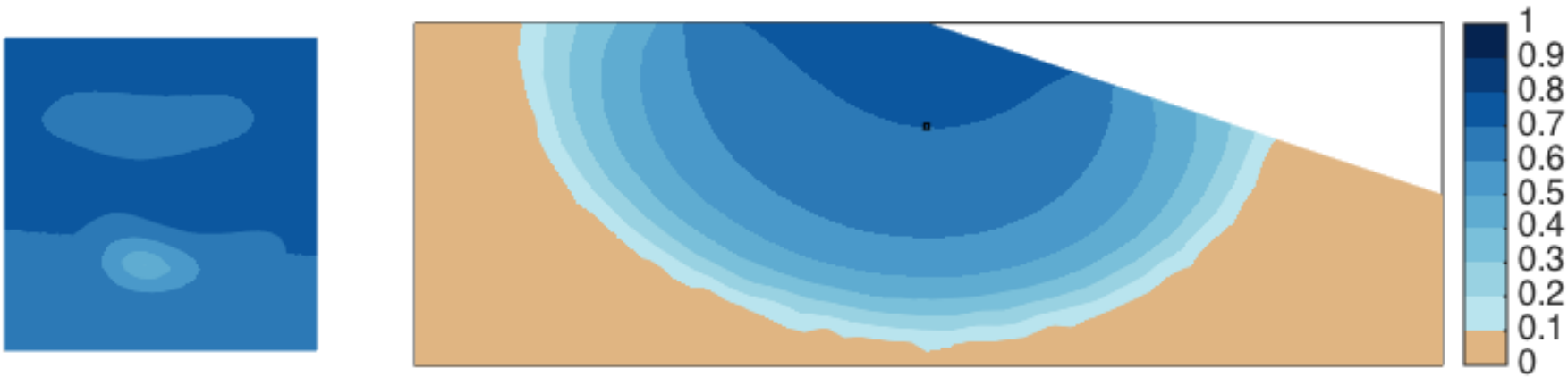
Saturation contours after 15 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

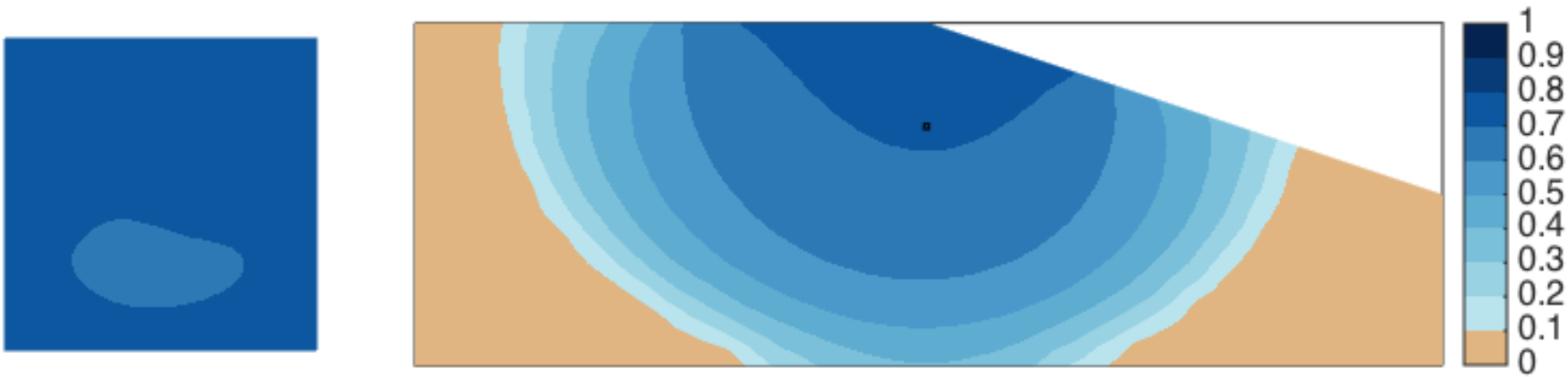
Saturation contours after 20 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

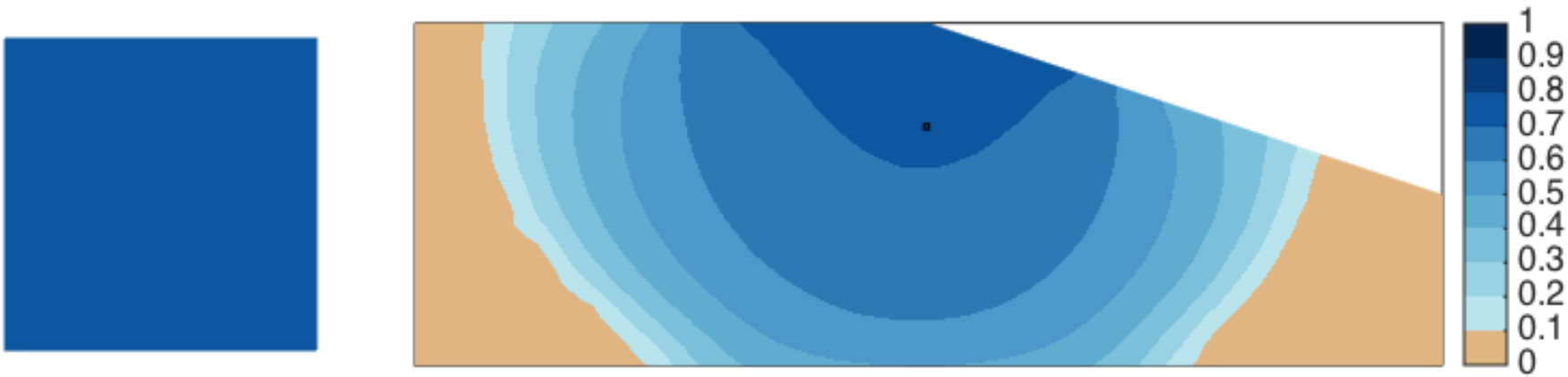
Saturation contours after 25 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

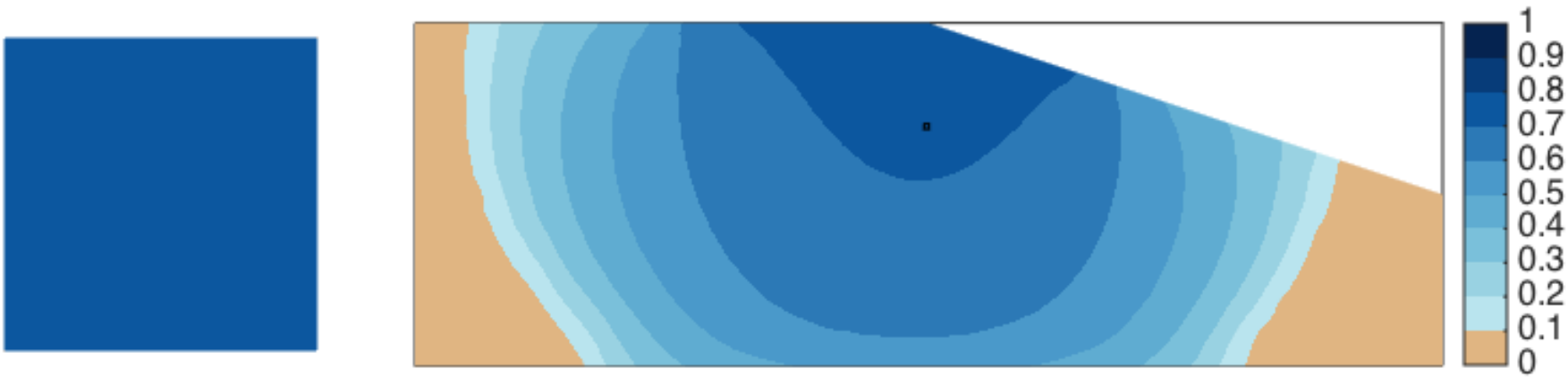
Saturation contours after 30 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

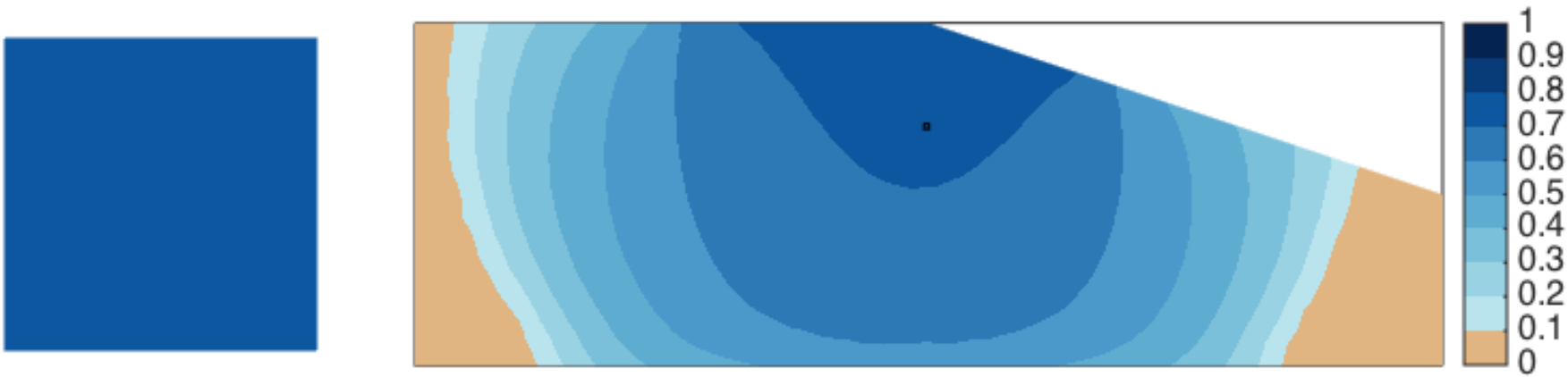
Saturation contours after 35 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

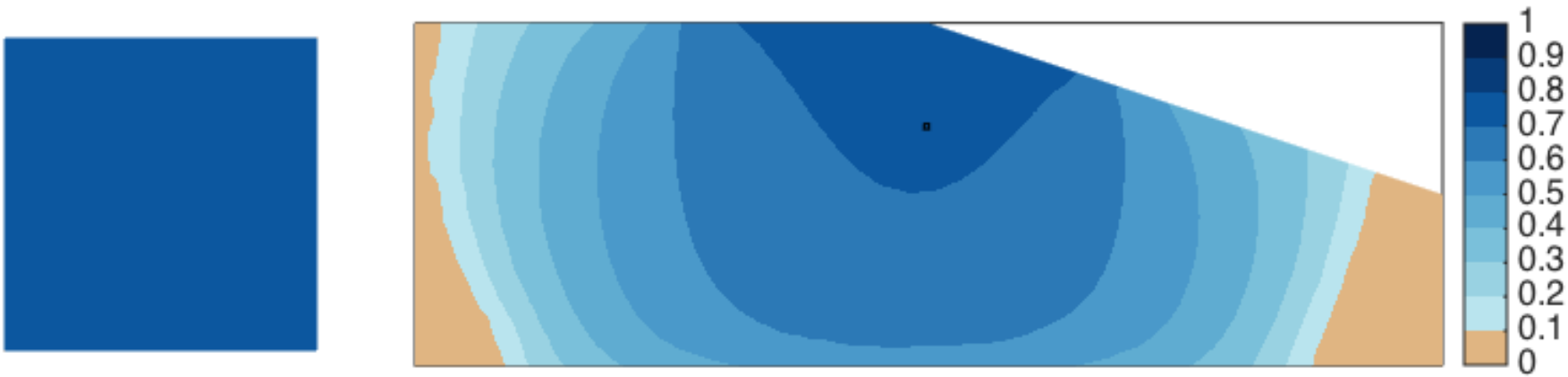
Saturation contours after 40 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

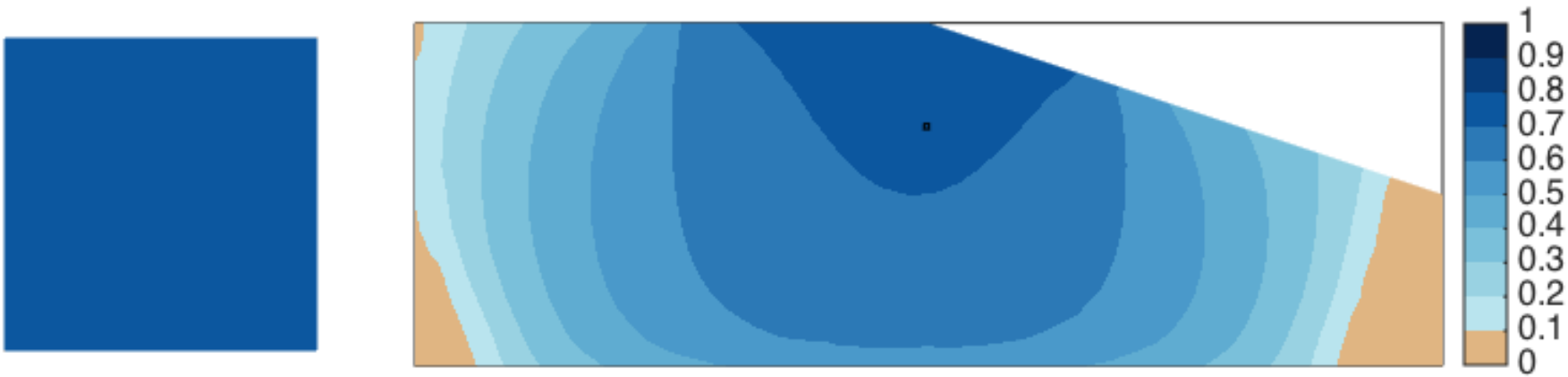
Saturation contours after 45 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

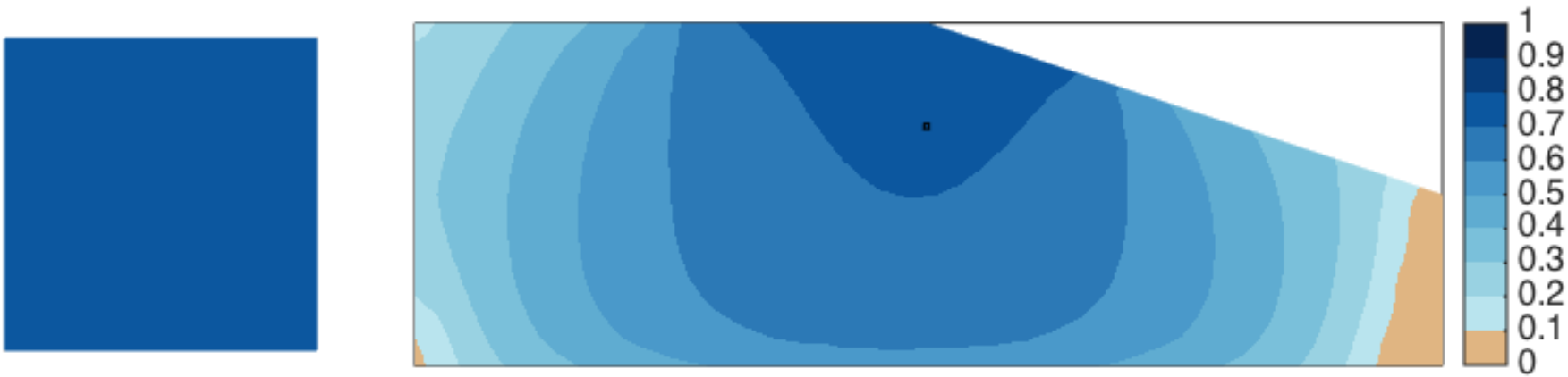
Saturation contours after 50 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

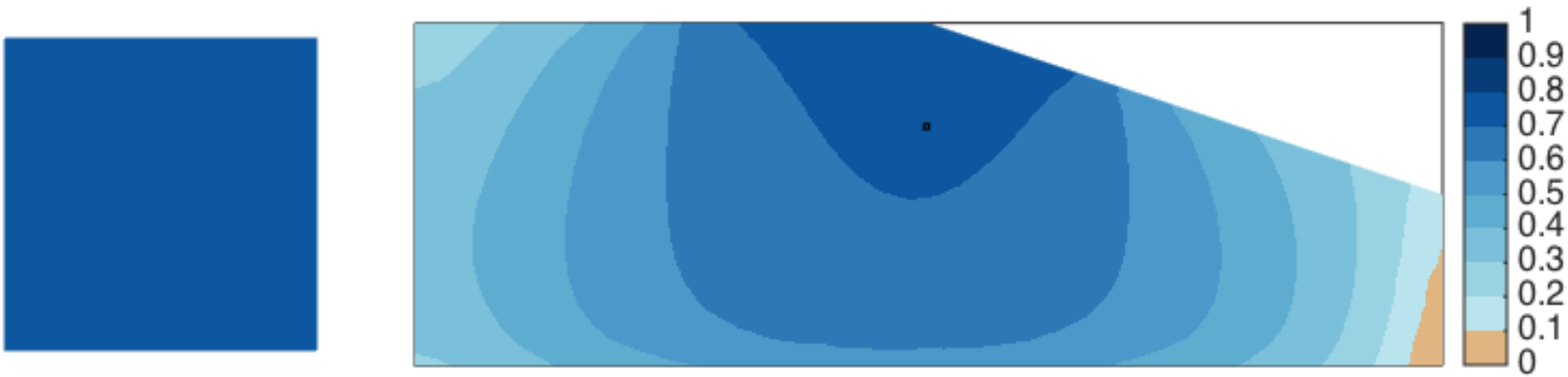
Saturation contours after 60 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

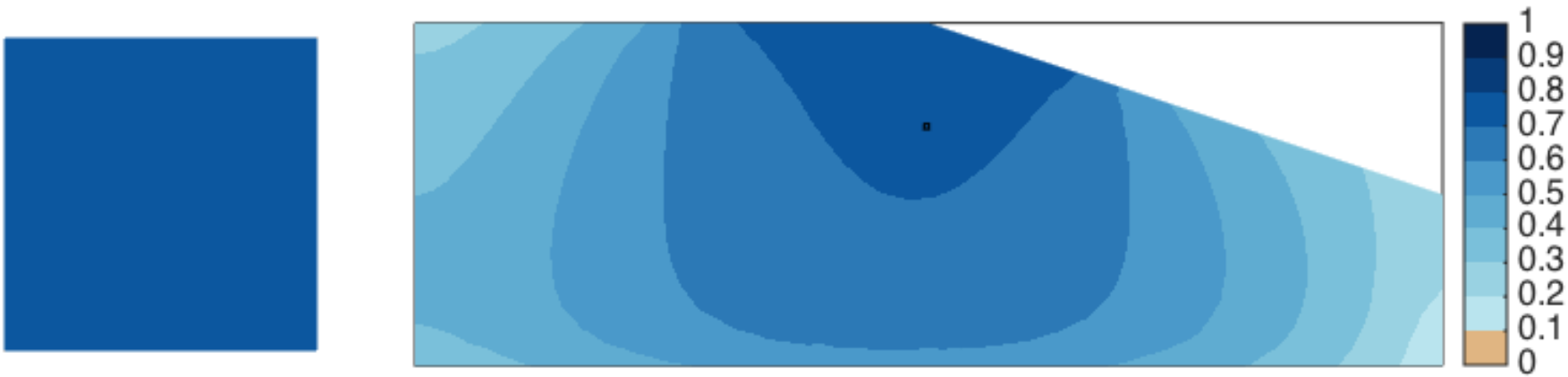
Saturation contours after 70 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

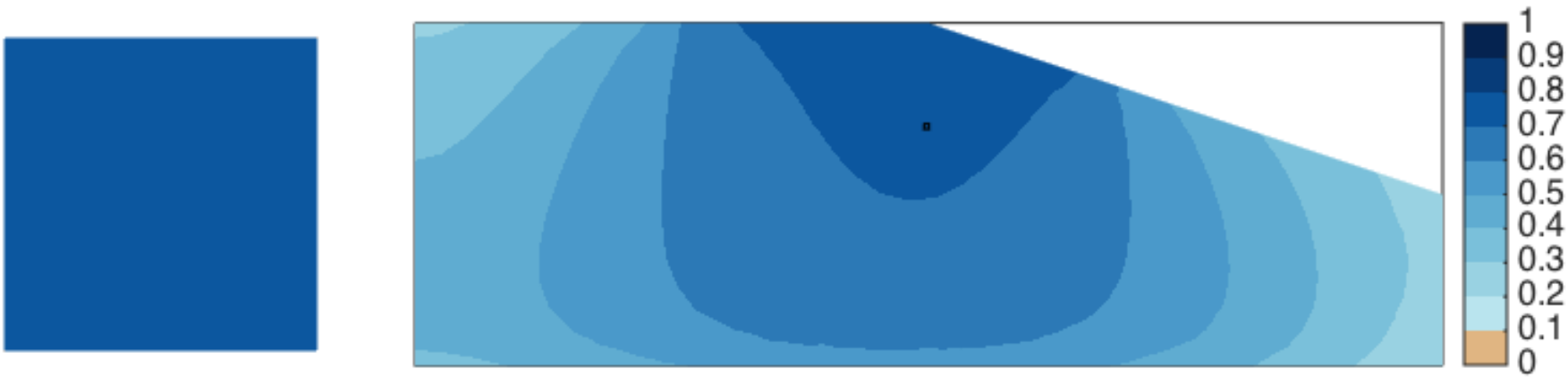
Saturation contours after 80 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

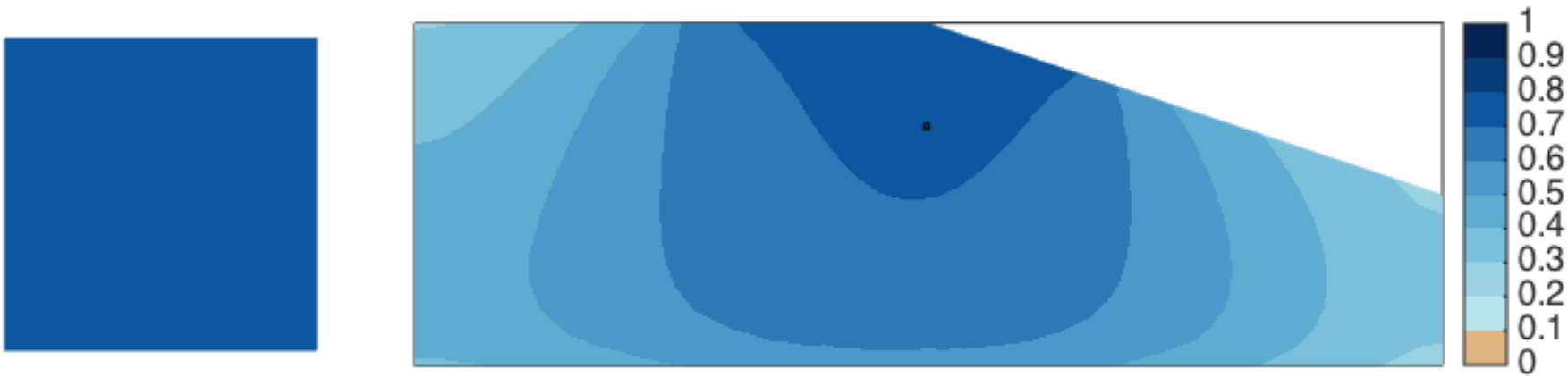
Saturation contours after 90 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

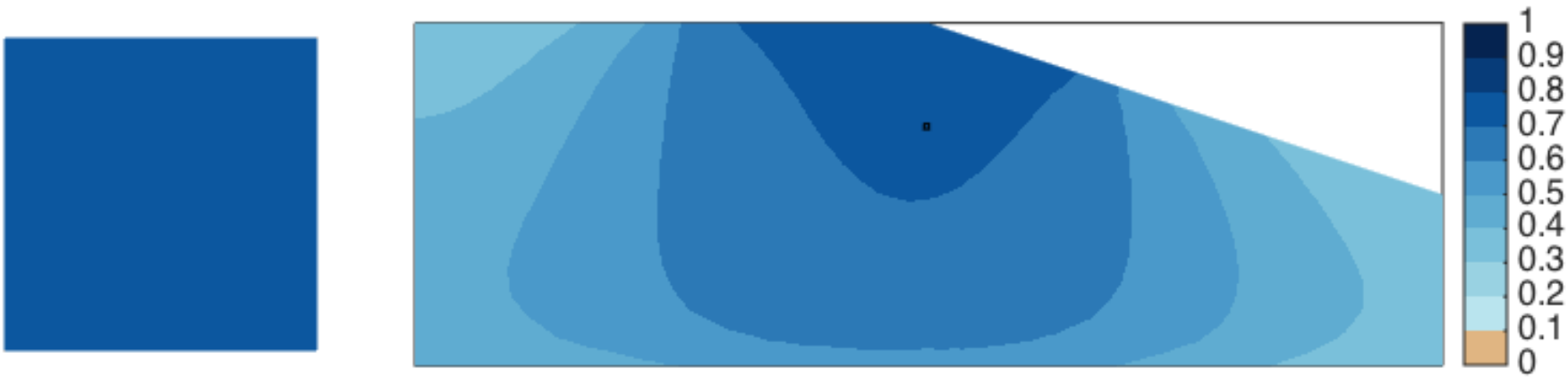
Saturation contours after 100 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

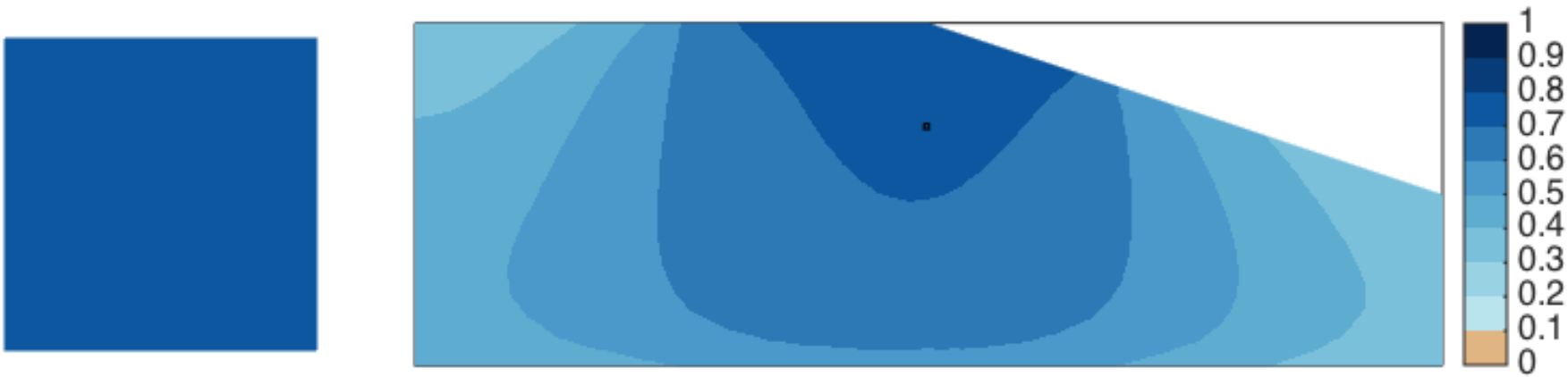
Saturation contours after 200 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

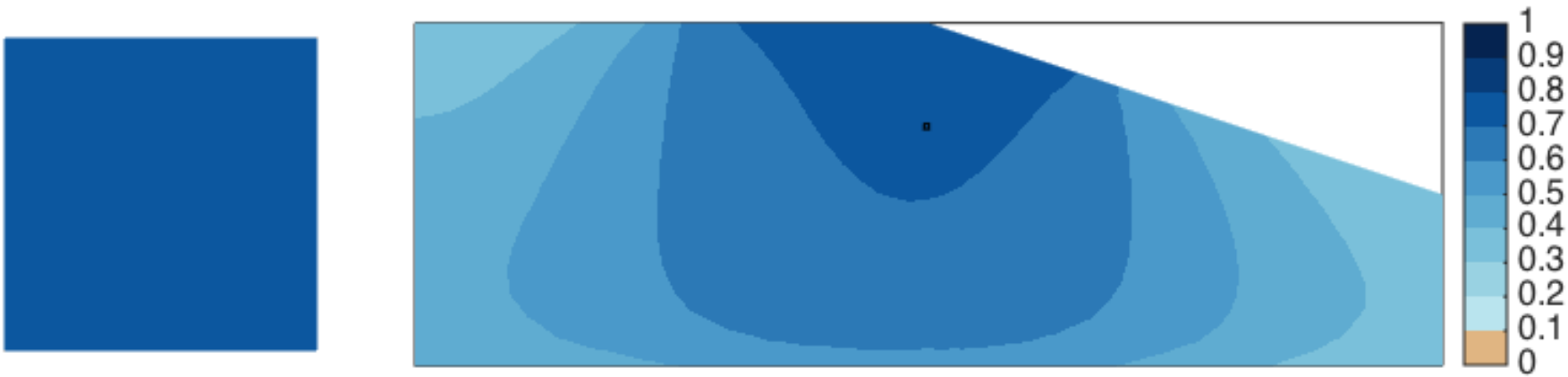
Saturation contours after 300 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Application to a heterogeneous aquifer

Saturation contours after 400 hours:



	Number of OpenMP threads				
	1	2	4	8	12
Runtime (min:sec)	40:58	22:05	11:55	07:44	05:31
Speedup	1.00	1.86	3.44	5.30	7.43

Summary

Pros:

- ✓ Ability to capture fine-scale detail in heterogeneous problem
- ✓ Produces good qualitative agreement with fine-scale model
- ✓ Significant reduction in simulation time (compared with fine-scale model)
- ✓ Numerically feasible for problems with very small-scale heterogeneities

Cons:

- ✗ Restricted to heterogeneous domains comprised of two sub-domains where one is connected and the other forms disconnected/isolated inclusions

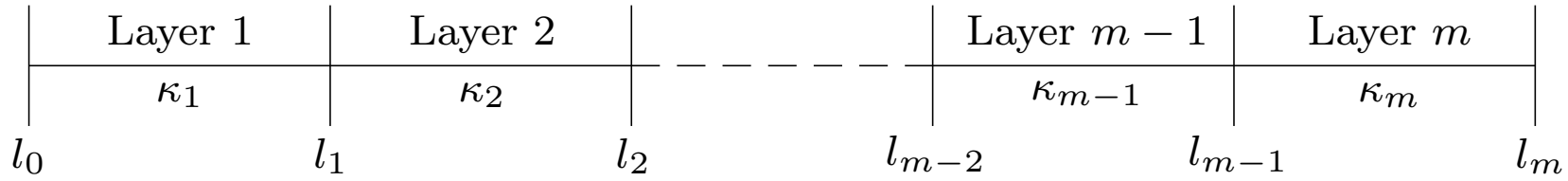
The next stage of this research will investigate two-scale approaches for arbitrary heterogeneous media...

Part II:
One-dimensional diffusion in a multilayer composite slab

Joint work with Ian Turner (QUT)

Fine-scale model

- Diffusion in a composite slab consisting of m layers:



- Diffusion equation defined on each layer ($i = 1, \ldots, m$):

$$\frac{\partial u_i}{\partial t} = \kappa_i \frac{\partial^2 u_i}{\partial x^2} \qquad l_{i-1} < x < l_i$$

- Initial condition $u_i(x, 0) = f(x)$ and external boundary conditions:

$$a_L u_1(l_0, t) + b_L \frac{\partial u_1}{\partial x}(l_0, t) = c_L \qquad a_R u_m(l_m, t) + b_R \frac{\partial u_m}{\partial x}(l_m, t) = c_R$$

- Suitable internal boundary conditions at the interfaces ($i = 1, \ldots, m-1$), e.g.,

$$u_i(l_i, t) = u_{i+1}(l_i, t) \qquad \kappa_i \frac{\partial u_i}{\partial x}(l_i, t) = \kappa_{i+1} \frac{\partial u_{i+1}}{\partial x}(l_i, t)$$

- Interested in the case when m is very large

One two-scale idea...

Example: Composite slab consisting of 100 layers

Macroscopic equation

$$\frac{\partial U}{\partial t} = \frac{\partial F}{\partial x} \qquad l_0 < x < l_m$$

subject to the initial condition $U(x,0) = f(x)$ and external boundary conditions:

$$a_L U(l_0,t) + b_L \frac{\partial U}{\partial x}(l_0,t) = c_L$$

$$a_R U(l_m,t) + b_R \frac{\partial U}{\partial x}(l_m,t) = c_R$$

is evolved on a coarse grid with unknown macroscopic flux F recovered from solution of the fine-scale model.

One two-scale idea...

Example: Composite slab consisting of 100 layers

Macroscopic equation

$$\frac{\partial U}{\partial t} = \frac{\partial F}{\partial x} \qquad l_0 < x < l_m$$

subject to the initial condition $U(x,0) = f(x)$ and external boundary conditions:

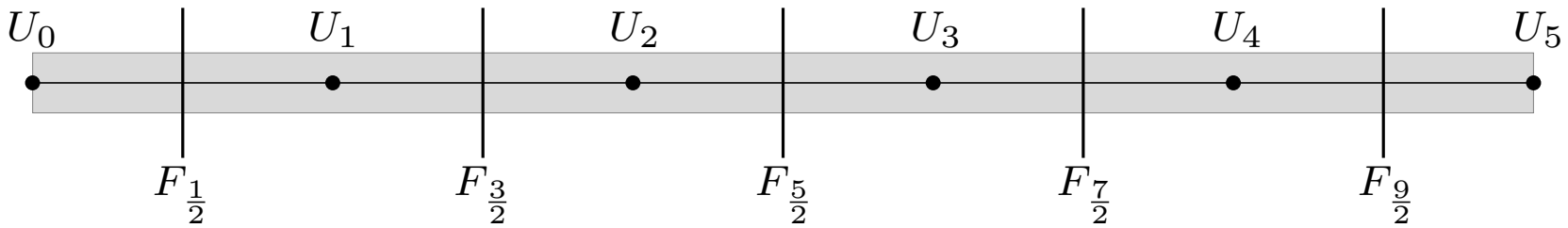
$$a_L U(l_0,t) + b_L \frac{\partial U}{\partial x}(l_0,t) = c_L$$

$$a_R U(l_m,t) + b_R \frac{\partial U}{\partial x}(l_m,t) = c_R$$

is evolved on a coarse grid with unknown macroscopic flux F recovered from solution of the fine-scale model.

One two-scale idea...

Example: Composite slab consisting of 100 layers



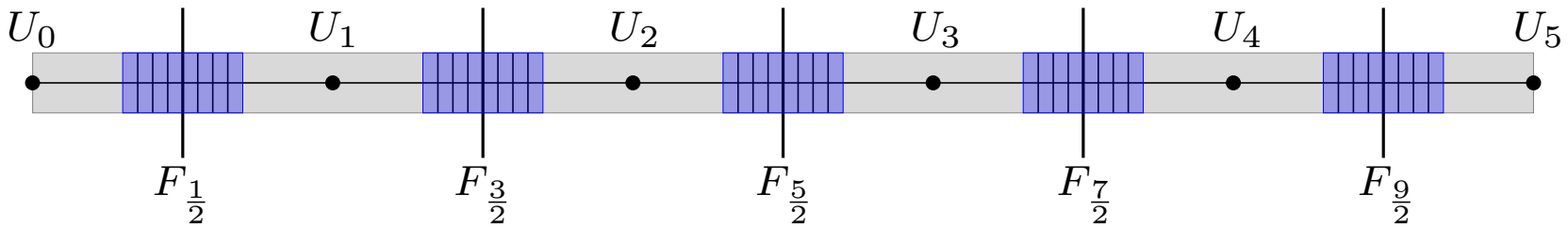
- Control volume discretisation in space:

$$\frac{dU_k}{dt} = \frac{F_{k+\frac{1}{2}} - F_{k-\frac{1}{2}}}{x_{k+\frac{1}{2}} - x_{k-\frac{1}{2}}}$$

- Micro-cells are centered around the control volume boundaries
- Macroscopic fluxes $F_{k+\frac{1}{2}}$ and $F_{k-\frac{1}{2}}$ recovered from the fine-scale solution on the micro-cells (average of fine-scale fluxes)
- Micro-cell boundary conditions depend on the macroscopic field U (what form should these take?)

One two-scale idea...

Example: Composite slab consisting of 100 layers



- Control volume discretisation in space:

$$\frac{dU_k}{dt} = \frac{F_{k+\frac{1}{2}} - F_{k-\frac{1}{2}}}{x_{k+\frac{1}{2}} - x_{k-\frac{1}{2}}}$$

- Micro-cells are centered around the control volume boundaries
- Macroscopic fluxes $F_{k+\frac{1}{2}}$ and $F_{k-\frac{1}{2}}$ recovered from the fine-scale solution on the micro-cells (average of fine-scale fluxes)
- Micro-cell boundary conditions depend on the macroscopic field U (what form should these take?)

Analytic solution for multilayer diffusion

- Exact solution in each layer can be expressed as [Hickson et al. (2009)]:

$$u_i(x,t) = w_i(x) + \sum_{n=0}^{\infty} c_n e^{-t\lambda_n^2} \phi_{i,n}(x)$$

- Eigenvalues and eigenfunctions satisfy:

$$\kappa_i \frac{d^2\phi_{i,n}}{dx^2} + \lambda_n^2 \phi_{i,n} = 0$$

subject to external boundary conditions:

$$a_L \phi_{1,n}(l_0) + b_L \frac{d\phi_{1,n}}{dx}(l_0) = 0\,, \qquad a_R \phi_{m,n}(l_m) + b_R \frac{d\phi_{m,n}}{dx}(l_m) = 0$$

and internal boundary conditions ($i = 1, \ldots, m-1$):

$$\phi_{i,n}(l_i) = \phi_{i+1,n}(l_i)\,, \qquad \kappa_i \frac{d\phi_{i,n}}{dx}(l_i) = \kappa_{i+1} \frac{d\phi_{i+1,n}}{dx}(l_i)$$

Analytic solution for multilayer diffusion

- The eigenfunctions take the form [**Ozisik (1968)**]:

$$\phi_{i,n}(x) = \alpha_{i,n} \sin\left(\frac{\lambda_n}{\sqrt{\kappa_i}}x\right) + \beta_{i,n} \cos\left(\frac{\lambda_n}{\sqrt{\kappa_i}}x\right)$$

  and the eigenvalues λ_n $(n = 0, 1, \ldots)$ are the positive roots of the transcendental equation:

$$\det(A(\lambda_n)) = 0$$

  where A is a $2m \times 2m$ matrix.

- For a large number of layers (large m), finding the eigenvalues is infeasible:
  - ✗ computing the determinant is numerically unstable for large matrices
  - ✗ run the risk of missing eigenvalues

- I couldn't get it working beyond 10 layers

Semi-analytic solution

Again, the solution is split into two parts:

$$u_i(x,t) = w_i(x) + v_i(x,t)$$

where $w_i(x)$ is the steady state solution and $v_i(x,t)$ satisfies:

$$\frac{\partial v_i}{\partial t} = \kappa_i \frac{\partial^2 v_i}{\partial x^2} \quad l_{i-1} < x < l_i$$

subject to the initial, internal and external boundary conditions:

$$v_i(x,0) = f(x) - w_i(x) \equiv \tilde{f}_i(x)\,, \quad i = 1,\ldots,m$$

$$a_L v_1(l_0,t) + b_L \frac{\partial v_1}{\partial x}(l_0,t) = 0$$

$$a_R v_m(l_m,t) + b_R \frac{\partial v_m}{\partial x}(l_m,t) = 0$$

$$v_i(l_i,t) = v_{i+1}(l_i,t)\,, \quad i = 1,\ldots,m-1$$

$$\kappa_i \frac{\partial v_i}{\partial x}(l_i,t) = \kappa_{i+1} \frac{\partial v_{i+1}}{\partial x}(l_i,t) \equiv g_i(t)\,, \quad i = 1,\ldots,m-1$$

Semi-analytic solution

Take Laplace transforms, and assume that the transformed solution can be represented as an orthogonal eigenfunction expansion:

$$\overline{v}_i(x,s) = \sum_{n=0}^{\infty} \langle \overline{v}_i, \phi_{i,n} \rangle \phi_{i,n}(x)$$

where the eigenvalues and eigenfunctions satisfy:

$$\frac{d^2\phi_{i,n}}{dx^2} + \lambda_{i,n}^2 \phi_{i,n} = 0$$

subject to boundary conditions local to each layer:

- First layer ($i = 1$): $a_L \phi_{1,n}(l_0) + b_L \frac{d\phi_{1,n}}{dx}(l_0) = 0$ and $\frac{d\phi_{1,n}}{dx}(l_1) = 0$
- Middle layers ($i = 2, \ldots, m-1$): $\frac{d\phi_{i,n}}{dx}(l_{i-1}) = 0$ and $\frac{d\phi_{i,n}}{dx}(l_i) = 0$
- Last layer ($i = m$): $\frac{d\phi_{m,n}}{dx}(l_{m-1}) = 0$ and $a_R \phi_{m,n}(l_m) + b_R \frac{d\phi_{m,n}}{dx}(l_m) = 0$

Semi-analytic solution

- Eigenvalues are roots of simple transcendental equations:
  - First layer ($i = 1$): $\lambda_{1,n}$ are positive roots of
$$a_L \lambda \sin(\lambda(l_1 - l_0)) = -b_L \cos(\lambda(l_1 - l_0))$$
  - Middle layers ($i = 2, \ldots, m-1$): $\lambda_{i,n}$ are non-negative roots of
$$\sin(\lambda(l_i - l_{i-1})) = 0$$
  - Last layer ($i = m$): $\lambda_{m,n}$ are positive roots of
$$a_R \lambda \sin(\lambda(l_m - l_{m-1})) = -b_R \cos(\lambda(l_m - l_{m-1}))$$
- Inverting the Laplace transformed solution, the solution within each layer is expressed in terms of inverse Laplace transforms involving the interface functions. For example, in the first layer [**Carr and Turner (2015)**]:
$$u_1(x,t) = w_1(x) + \sum_{n=0}^{\infty} \left[c_{1,n} e^{-t\kappa_1 \lambda_{1,n}^2} + \mathcal{L}^{-1} \left\{ \frac{\overline{g}_1(s)}{s + \kappa_1 \lambda_{1,n}^2} \right\} \phi_{1,n}(l_1) \right] \phi_{1,n}(x)$$

Semi-analytic solution

- Inverse Laplace transforms are computed numerically:

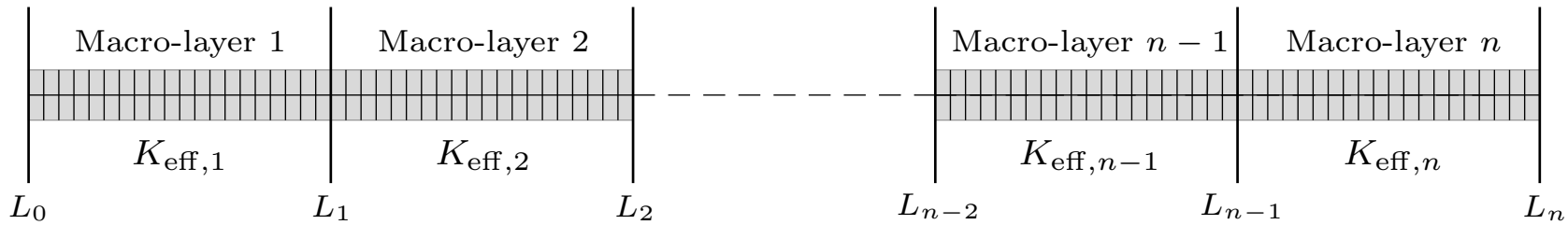
$$\mathcal{L}^{-1}\left\{\frac{\overline{g}_i(s)}{s+\kappa_i\lambda_n^2}\right\} \approx -2\Re\left\{\sum_{k=1}^{N/2} c_{2k-1}\frac{\overline{g}_i(z_{2k-1}/t)}{z_{2k-1}+\kappa_i\lambda_n^2 t}\right\},$$

where c_{2k-1} and z_{2k-1} are the residues and poles of the best $(N-1,\ N)$ rational approximation to e^z on the negative real line computed using the Carathéodory-Fejér method [**Trefethen et al. (2006)**].

- Advantages of this approach:
  - ✓ Eigenvalues are roots of simple transcendental equations that are easy to solve numerically
  - ✓ Avoids solving a complicated transcendental equation for the eigenvalues involving the matrix determinant
  - ✓ Scales well to a large number of layers (tested up to 10,000 layers with no problems)

Application to macroscopic modelling

- **Fine-scale model**: composite slab consisting of m layers
- **Macroscopic model**: composite slab consisting of n layers ($n \ll m$):



$$\frac{\partial U_k}{\partial t} = K_{\text{eff},k} \frac{\partial^2 U_k}{\partial x^2} \qquad L_{k-1} < x < L_k$$

with effective diffusivity (harmonic average) satisfying:

$$\frac{L_k - L_{k-1}}{K_{\text{eff},k}} = \sum_{i \in S_j} \frac{l_i - l_{i-1}}{\kappa_i}$$

where S_j is the set of micro-layers comprising macro-layer k.

Application to macroscopic modelling

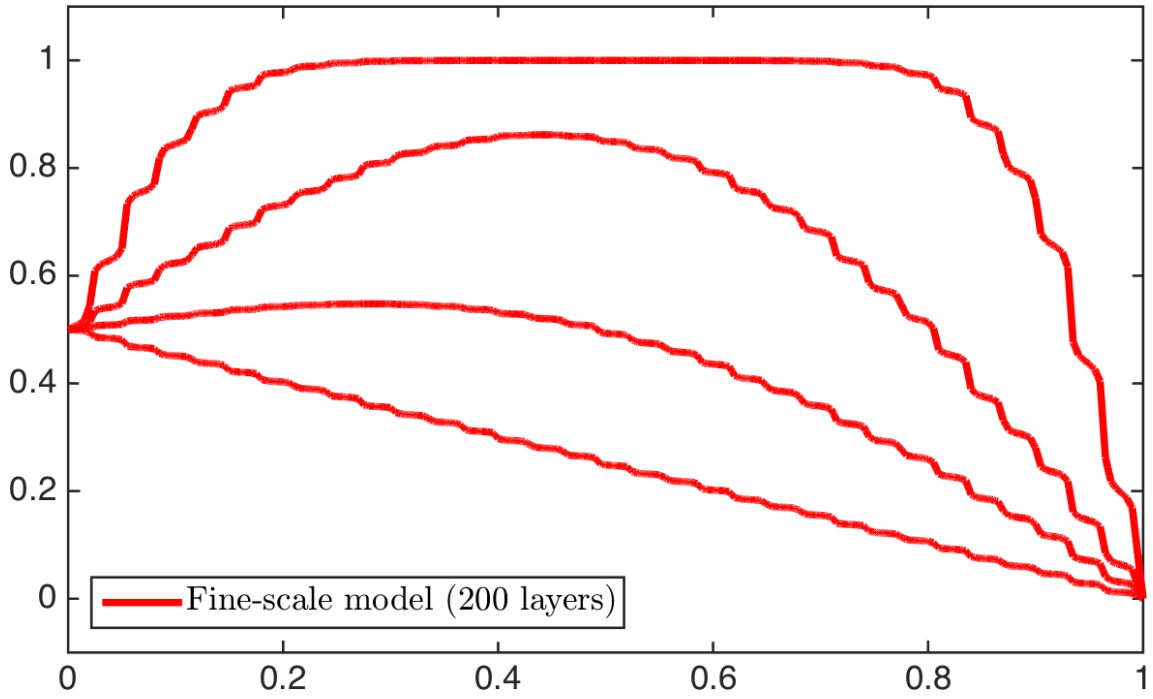


Figure: Micro diffusivities $\kappa_i = 1.1 + \sin(i)$ $(i = 1, \ldots, 200)$

- Fine-scale model (200 layers): 20 secs runtime
- Macroscopic model (10 layers): 2 secs runtime

Application to macroscopic modelling

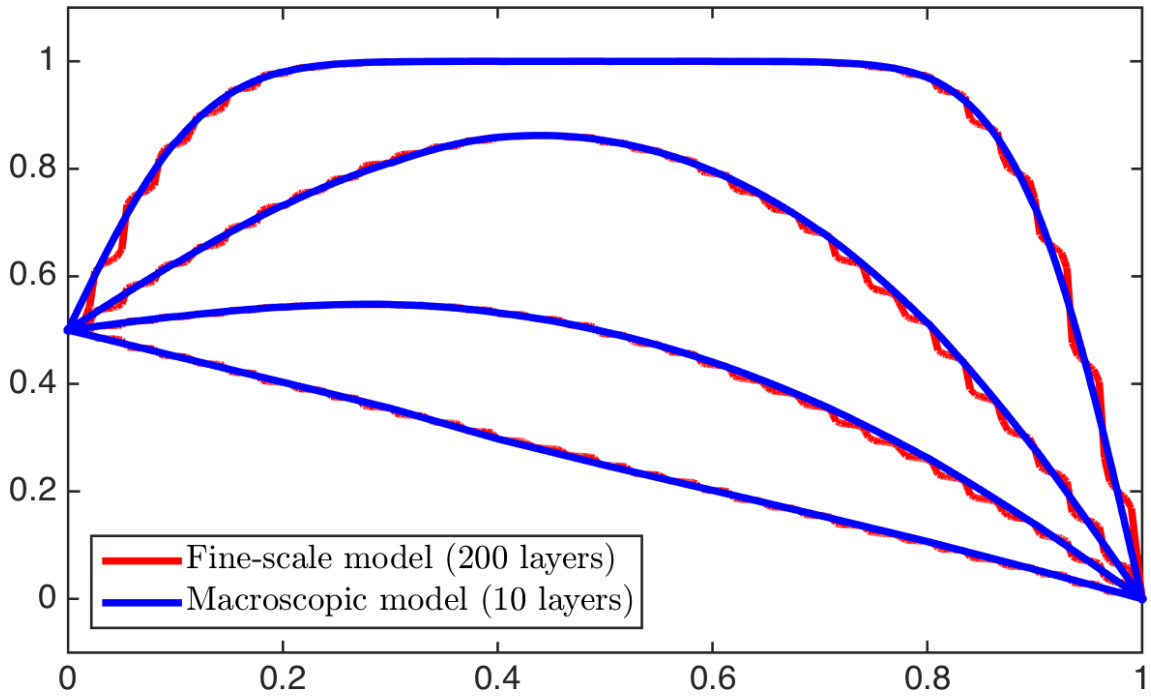


Figure: Micro diffusivities $\kappa_i = 1.1 + \sin(i)$ $(i = 1, \ldots, 200)$

- Fine-scale model (200 layers): 20 secs runtime
- Macroscopic model (10 layers): 2 secs runtime

Conclusions and Future Outlook

Two-scale models:

- are useful for problems involving fine-scale heterogeneities
- can produce excellent agreement with fine-scale solutions with a significant reduction in simulation time

Future work:

- Extend the two-scale model for unsaturated water flow in a binary medium to a full two-phase formulation (e.g., air and water or water and oil)
- Complete work on two-scale modelling for multilayer diffusion
- Overall goal will be to eventually combine these models to form a generic two-scale model for multiphase transport/flow problems in arbitrary heterogeneous porous media

Thank you!

References

E. J. Carr and I. W. Turner. Two-scale computational modelling of water flow in unsaturated soils containing irregular-shaped inclusions. *Int. J. Numer. Meth. Eng.*, 98:157–173, 2014.

E. J. Carr and I. W. Turner. *Upcoming paper*, 2015.

E. J. Carr, T. J. Moroney, and I. W. Turner. Efficient simulation of unsaturated flow using exponential time integration. *Appl. Math. Comput.*, 217(14):6587–6596, 2011.

E. J. Carr, I. W. Turner, and P. Perré. A variable-stepsize Jacobian-free exponential integrator for simulating transport in heterogeneous porous media: Application to wood drying. *J. Comput. Phys.*, 233: 66–82, 2013.

E. J. Carr, I. W. Turner, and P. Perré. *Upcoming paper*, 2015.

R. I. Hickson, S. I. Barry, and G. N. Mercer. Critical times in multilayer diffusion. Part 1: Exact solutions. *Int. J. Heat Mass Tran.*, 52:5776–5783, 2009.

M. Necati Ozisik. *Boundary value problems of heat conduction*. International Textbook Company, Scranton, Pennsylvania, 1968.

A. Szymkiewicz and J. Lewandowska. Unified macroscopic model for unsaturated water flow in soils of bimodal porosity. *Hydrolog. Sci. J.J*, 51(6):1106–1124, 2006.

L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer. Talbot quadratures and rational approximations. *BIT Numer. Math.*, 46:653–670, 2006.

M. T. van Genuchten. A closed–form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am. J.*, 44(5):892–898, 1980.