A Newton-like Method based on Model Reduction Techniques for Implicit Numerical Methods

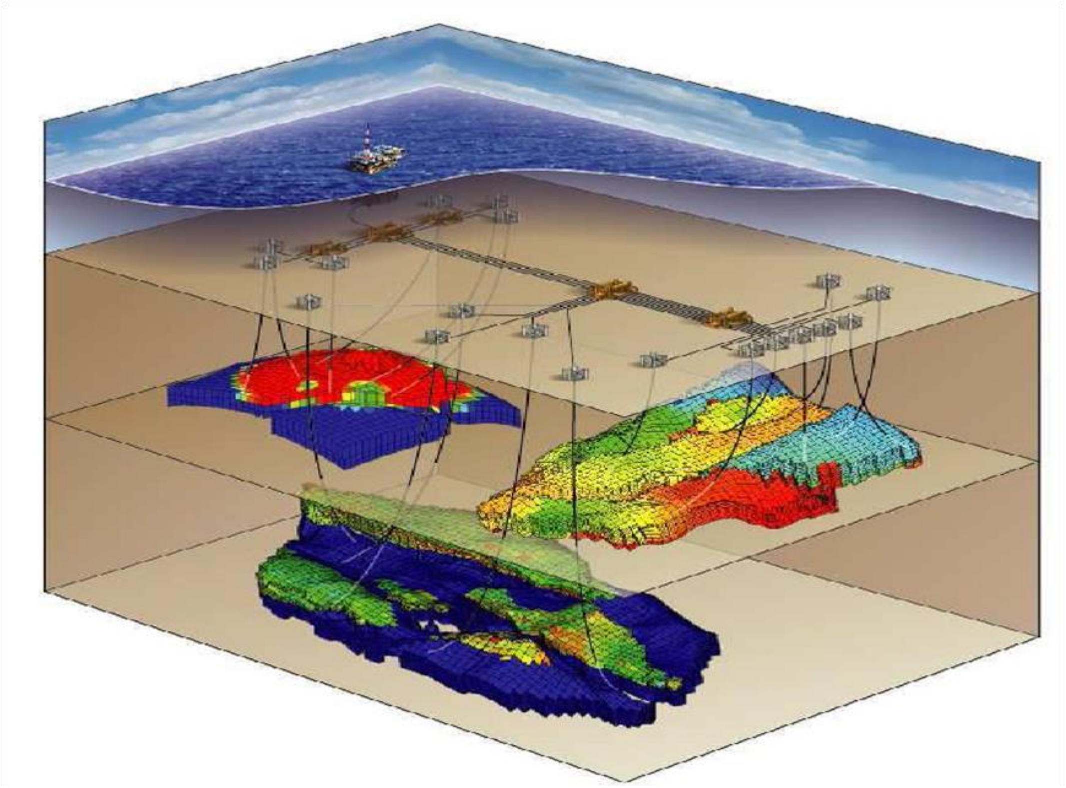
Tobias K. S. Ritschel

Department of Applied Mathematics and Computer Science,
Technical University of Denmark

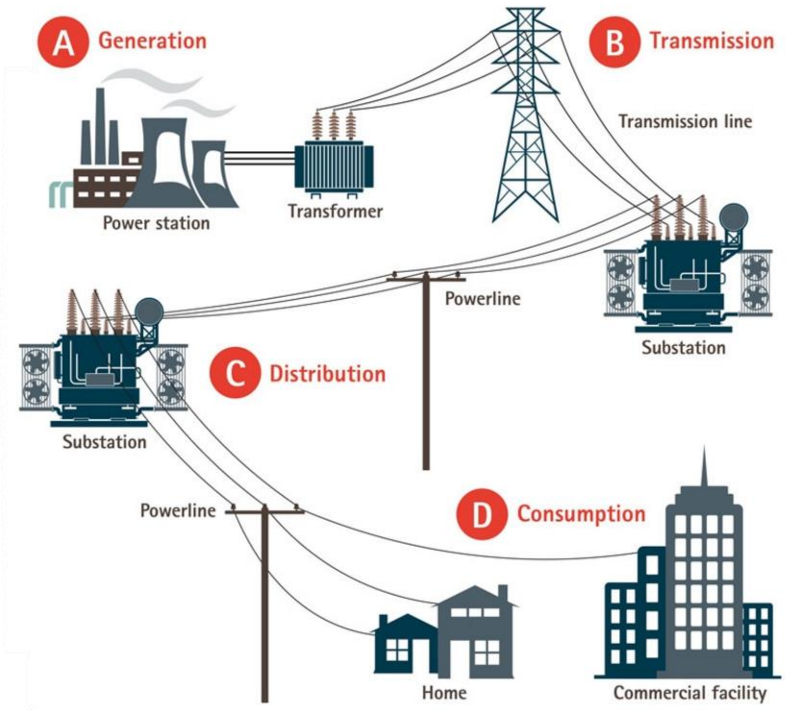
14:20 – 14:40, FrB1.1, June 16, 2023
European Control Conference, Bucharest, Romania

Large-scale systems

Transport processes



Coupled/networked systems



Typical discretization (e.g., finite volume) => stiff ODEs/DAEs

Stiffness depends on subsystems

Numerical methods and model reduction

Model reduction of linear systems

Original model

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$\dot{x} = A \; x + B \; u$$

$$y = C \; x + D \; u$$

Reduced model

$$\dot{\hat{x}}(t) = A_r\hat{x}(t) + B_r u(t)$$
$$\hat{y}(t) = C_r\hat{x}(t) + D_r u(t)$$

$$x(t) \approx V\hat{x}(t)$$

$$\dot{\hat{x}} = A_r \; \hat{x} + B_r \; u$$

$$\hat{y} = C_r \; \hat{x} + D_r \; u$$

Model reduction of nonlinear systems

$$\begin{aligned}
\dot{x}(t) &= f(x(t), u(t), d(t), p) \\
u(t) &= u_k, \quad t \in [t_k, t_{k+1}[\\
d(t) &= d_k, \quad t \in [t_k, t_{k+1}[
\end{aligned}$$

$$\begin{aligned}
V\dot{\hat{x}}(t) &= f(V\hat{x}(t), u(t), d(t), p) \\
W^T V\dot{\hat{x}}(t) &= W^T f(V\hat{x}(t), u(t), d(t), p)
\end{aligned}$$

$$\begin{aligned}
\dot{\hat{x}}(t) &= \hat{f}(\hat{x}(t), u(t), d(t), p) \\
\hat{f}(\hat{x}(t), u(t), d(t), p) &= (W^T V)^{-1} W^T f(V\hat{x}(t), u(t), d(t), p)
\end{aligned}$$

- For explicit methods: No gain in computation time
- Difficult to obtain an accurate approximation

Euler's implicit method and Newton's method

$$
\begin{aligned}
x_{k+1} - x_k &= f(x_{k+1}, u_k, d_k, p)\Delta t_k \\
R_k(x_{k+1}) &= R_k(x_{k+1}; x_k, u_k, d_k, p) \\
&= x_{k+1} - x_k - f(x_{k+1}, u_k, d_k, p)\Delta t_k = 0
\end{aligned}
$$

$$
\begin{aligned}
R_k(x_{k+1}^{(\ell)}) + \frac{\partial R_k}{\partial x_{k+1}}(x_{k+1}^{(\ell)})\Delta x_{k+1}^{(\ell)} &= 0 \\
\frac{\partial R_k}{\partial x_{k+1}}(x_{k+1}) = \mathrm{I} - \frac{\partial f}{\partial x}(x_{k+1}, u_k, d_k, p)\Delta t_k
\end{aligned}
$$

$$
\begin{aligned}
x_{k+1}^{(\ell+1)} &= x_{k+1}^{(\ell)} + \Delta x_{k+1}^{(\ell)} \\
A_k^{(\ell)} \Delta x_{k+1}^{(\ell)} &= b_k^{(\ell)}
\end{aligned}
$$

$$
\|R_k(x_{k+1}^{(\ell)})\| < \tau
$$

$$
\begin{aligned}
A_k^{(\ell)} &= \frac{\partial R_k}{\partial x_{k+1}}(x_{k+1}^{(\ell)}) \\
b_k^{(\ell)} &= -R_k(x_{k+1}^{(\ell)})
\end{aligned}
$$

Euler's implicit method and Newton's method

$$\Delta x_{k+1}^{(\ell)} \approx V_k \Delta \hat{x}_{k+1}^{(\ell)}$$
$$W_k^T A_k^{(\ell)} V_k \Delta \hat{x}_{k+1}^{(\ell)} = W_k^T b_k^{(\ell)}$$

$$x_{k+1}^{(\ell+1)} = x_{k+1}^{(\ell)} + V_k \Delta \hat{x}_{k+1}^{(\ell)}$$
$$\hat{A}_k^{(\ell)} \Delta \hat{x}_{k+1}^{(\ell)} = \hat{b}_k^{(\ell)}$$

$$x_{k+1}^{(\ell+1)} = x_{k+1}^{(\ell)} + \Delta x_{k+1}^{(\ell)}$$
$$A_k^{(\ell)} \Delta x_{k+1}^{(\ell)} = b_k^{(\ell)}$$

$$\|R_k(x_{k+1}^{(\ell)})\| < \tau$$

$$X = U \Sigma R$$
$$V = W$$

$$\hat{A}_k^{(\ell)} = W_k^T A_k^{(\ell)} V_k$$
$$\hat{b}_k^{(\ell)} = W_k^T b_k^{(\ell)}$$

$$A_k^{(\ell)} = \frac{\partial R_k}{\partial x_{k+1}}(x_{k+1}^{(\ell)})$$
$$b_k^{(\ell)} = -R_k(x_{k+1}^{(\ell)})$$

- Columns of U corresponding to singular values larger than 100*2^-53 times the largest

Euler's implicit method and Newton's method

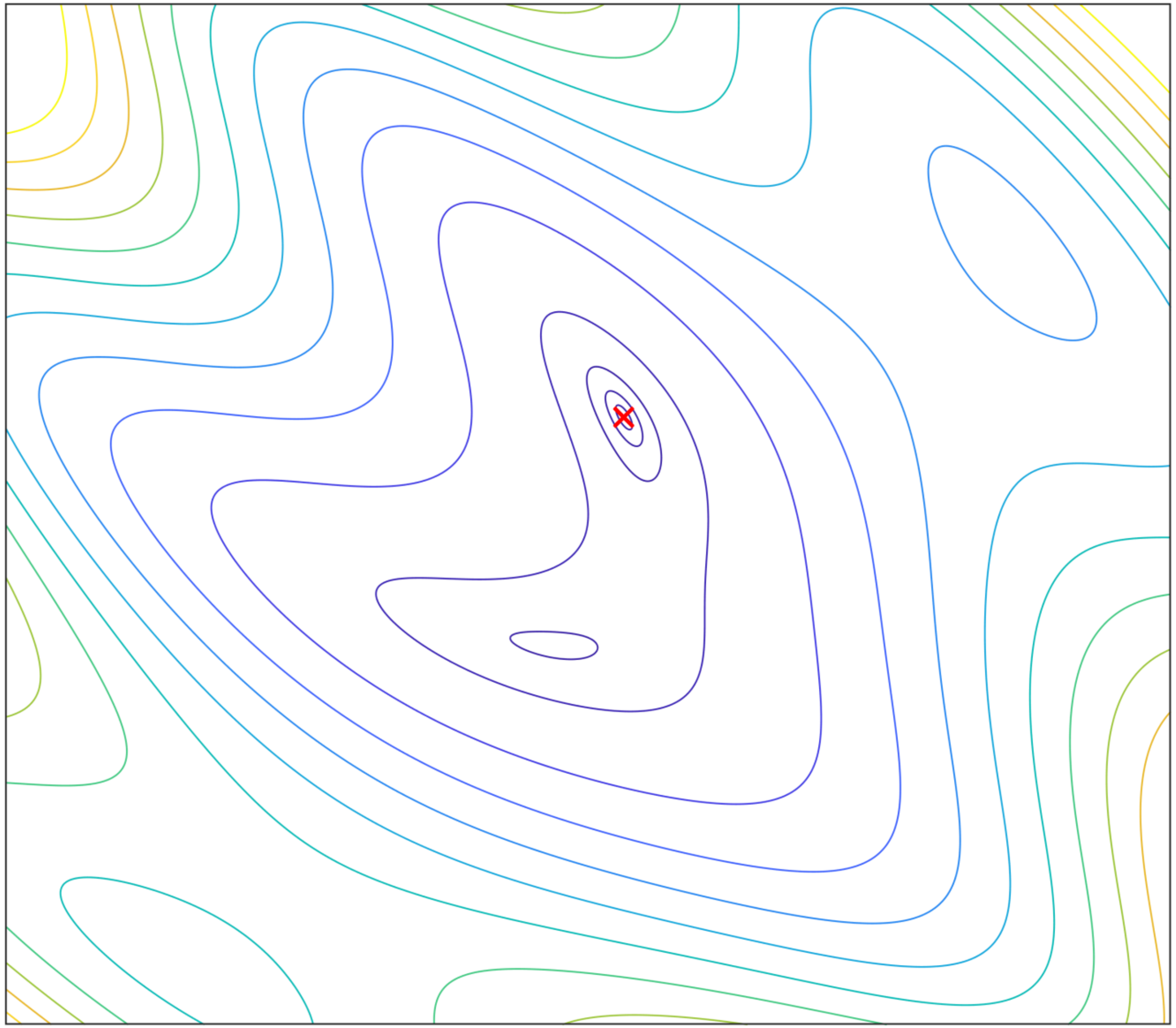
$$x_{k+1}^{(\ell+1)} = x_{k+1}^{(\ell)} + V_k \Delta \hat{x}_{k+1}^{(\ell)}$$
$$\hat{A}_k^{(\ell)} \Delta \hat{x}_{k+1}^{(\ell)} = \hat{b}_k^{(\ell)}$$

$$x_{k+1}^{(\ell+1)} = x_{k+1}^{(\ell)} + \Delta x_{k+1}^{(\ell)}$$
$$A_k^{(\ell)} \Delta x_{k+1}^{(\ell)} = b_k^{(\ell)}$$

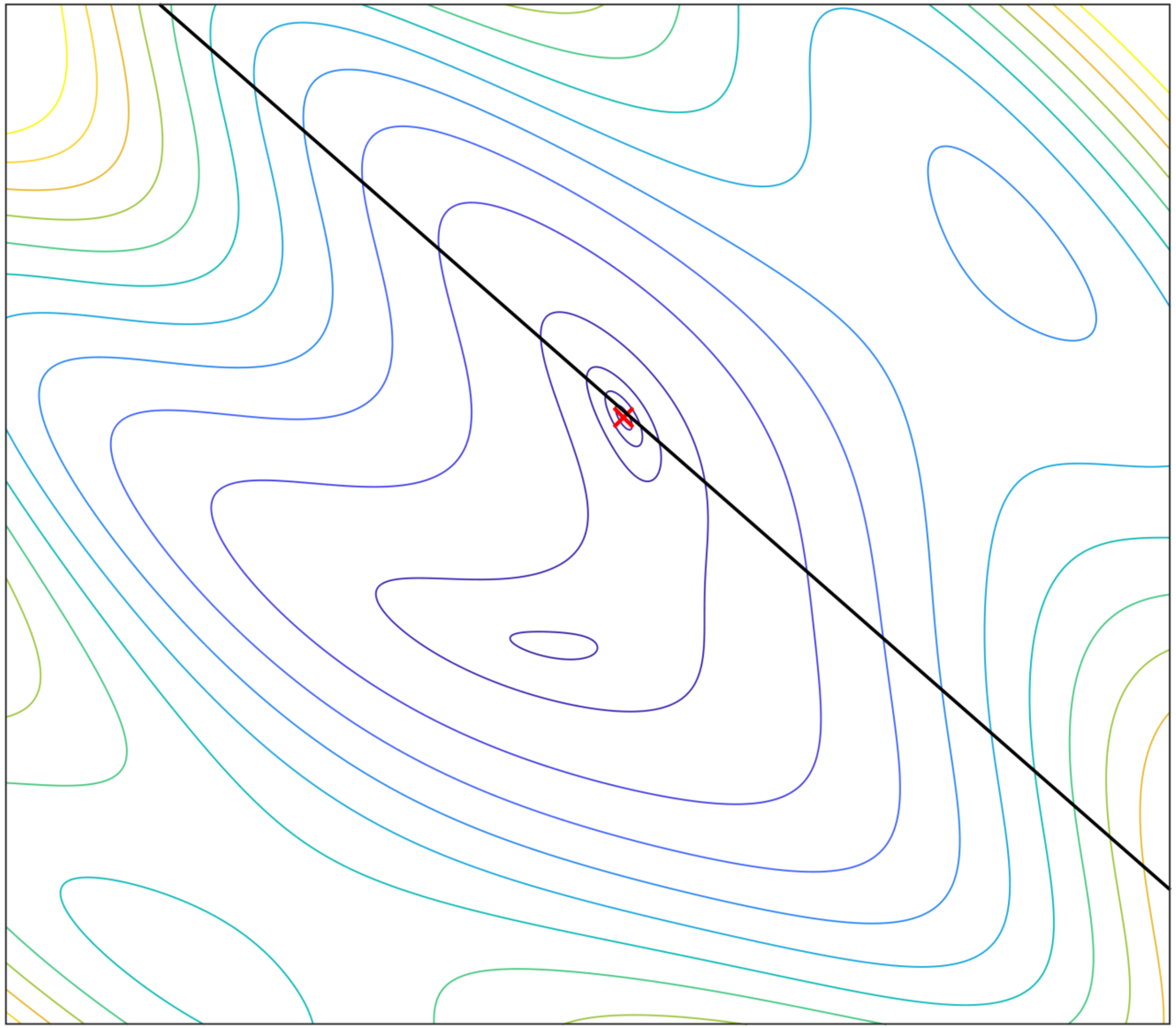
$$\hat{A} \quad \hat{x} \quad = \quad \hat{b}$$

$$A \quad x \quad = \quad b$$

Subspace methods



Subspace methods



Simulation based on Newton-like method

Algorithm 1: Simulation based on Newton-like method

Input: x_0, $\{\Delta t_k\}_{k=0}^{N-1}$, $\{u_k\}_{k=0}^{N-1}$, $\{d_k\}_{k=0}^{N-1}$, p, N, N_b, N_h, τ

Output: $\{x_k\}_{k=0}^{N}$

for $k = 0, \ldots, N-1$ **do**
- **if** $k \geq N_b$ **then**
  - Set $k_h = \max\{0, k - N_h + 1\}$;
  - Set $X_k = \begin{bmatrix} x_{k_h} & \cdots & x_k \end{bmatrix}$;
  - Compute V_k and W_k using POD of X_k;
  - Compute x_{k+1} using Algorithm 2;
- **else**
  - Compute x_{k+1} using Newton's method;
- **end**

end

$$N_b = \ln(n_x)$$

$$N_h = \sqrt[3]{n_x}$$

Algorithm 2: Newton-like method

Input: x_k, V_k, W_k, τ

Output: x_{k+1}

Set $\ell = 0$;

Set $x_{k+1}^{(0)} = x_k$;

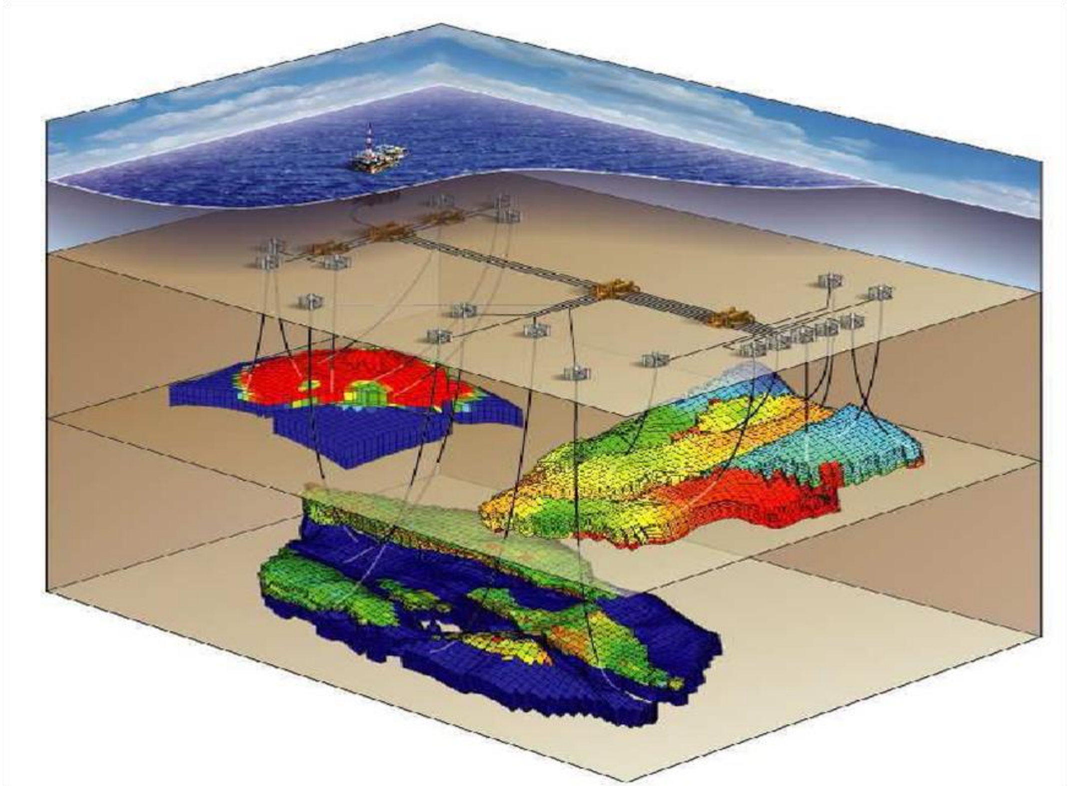
while $\|R_k(x_{k+1}^{(\ell)})\| \geq \tau$ **do**
- **if** $\ell > 0$ *and* $\|\Delta \hat{x}_{k+1}^{(\ell-1)}\| < \tau$ **then**
  - Solve the full order system (9) for $\Delta x_{k+1}^{(\ell)}$;
  - Compute $x_{k+1}^{(\ell+1)}$ using (5);
  - Set $\|\Delta \hat{x}_{k+1}^{(\ell)}\|$ larger than τ;
- **else**
  - Solve the reduced system (18) for $\Delta \hat{x}_{k+1}^{(\ell)}$;
  - Compute $x_{k+1}^{(\ell+1)}$ using (17);
- **end**
- Increment ℓ by 1;

end

Set $x_{k+1} = x_{k+1}^{(\ell)}$;

Numerical example

Numerical example of CO_2 storage



$$
\begin{aligned}
\partial_t C_w &= -\nabla \cdot \mathbf{N}^w, \\
\partial_t C_k &= -\nabla \cdot \mathbf{N}_k + Q_k, \\
\mathbf{N}_k &= x_k \mathbf{N}^o \\
\mathbf{N}^\alpha &= \rho^\alpha \mathbf{u}^\alpha \\
\mathbf{u}^\alpha &= -\frac{k_r^\alpha}{\mu^\alpha}\mathbf{K}(\nabla P - \bar{\rho}^\alpha g \nabla z) \\
Q_k &= x_k^{\mathrm{inj}} Q^o
\end{aligned}
$$

$$
P = \frac{RT}{v^o - b_m} - \frac{a_m}{(v^o + \epsilon b_m)(v^o + \sigma b_m)}
$$

$$
\begin{aligned}
\dot{x}(t) &= f(x(t), u(t), d(t), p) \\
&= F(\tilde{G}(x(t), p), u(t), d(t), p)
\end{aligned}
$$

Numerical example of CO_2 storage

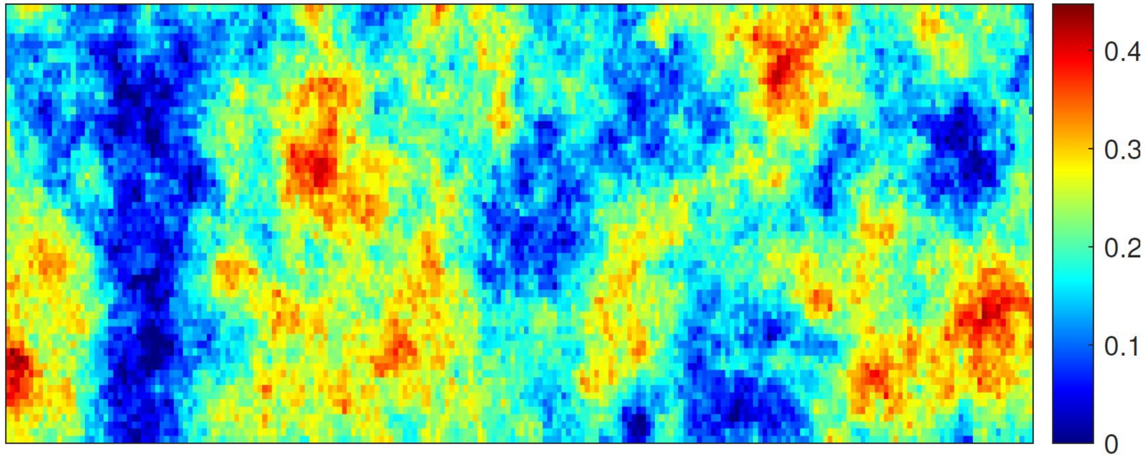


Fig. 1. Porosity field.

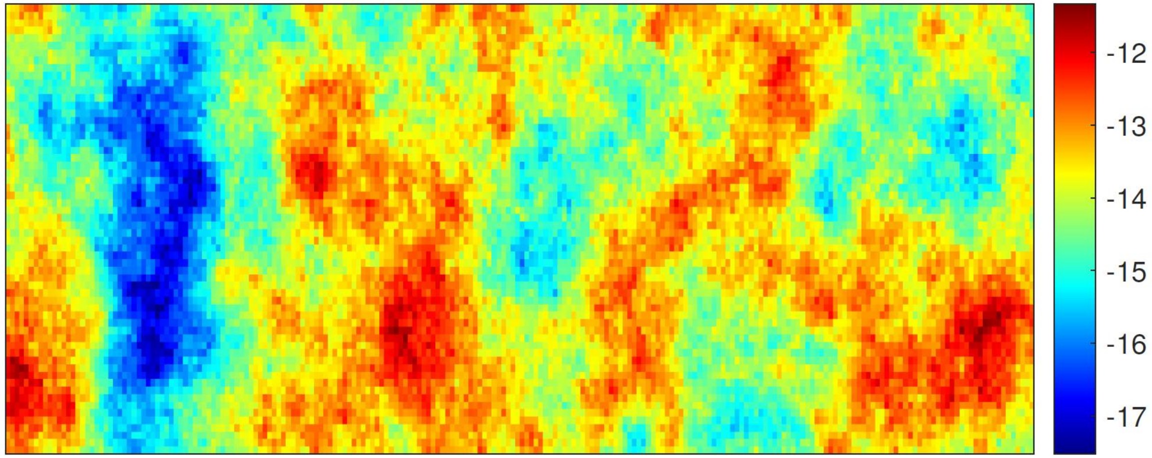


Fig. 2. Logarithm of the permeability field in m^2.

Numerical example of CO_2 storage

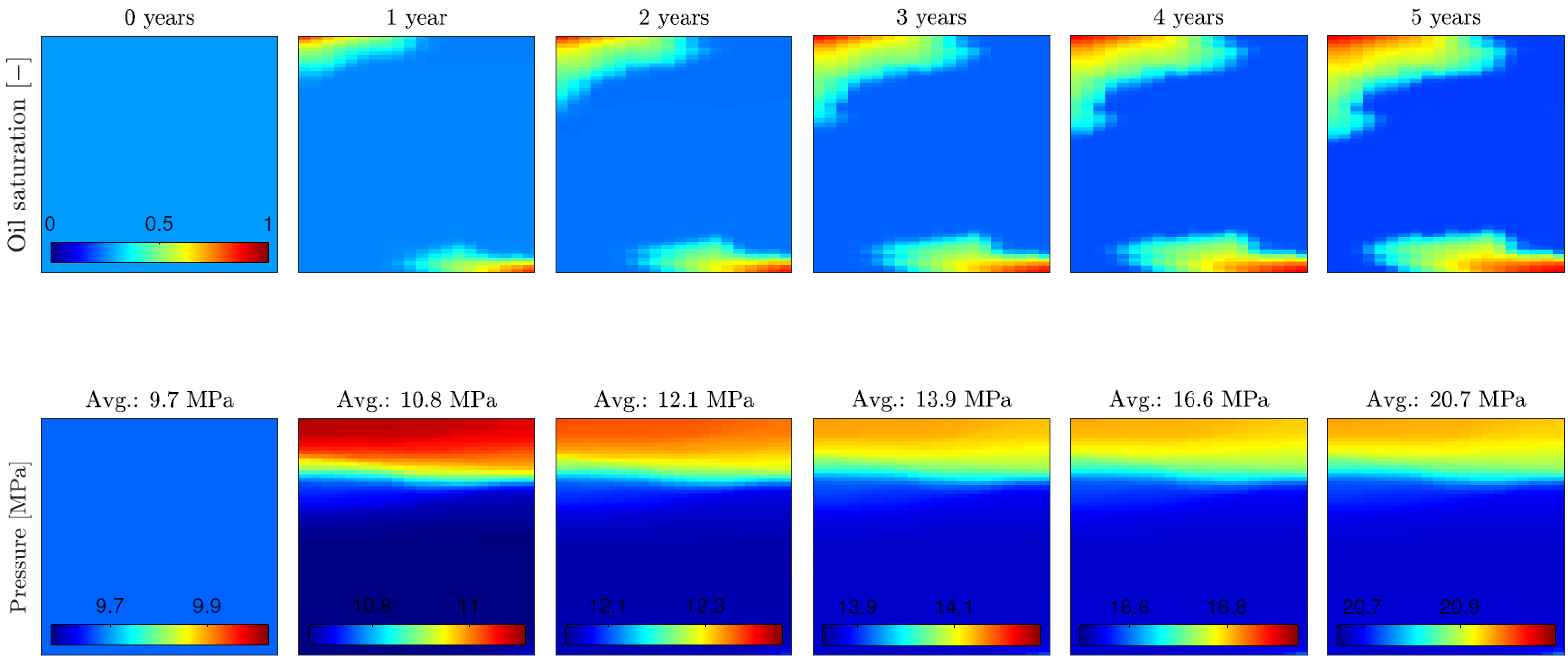
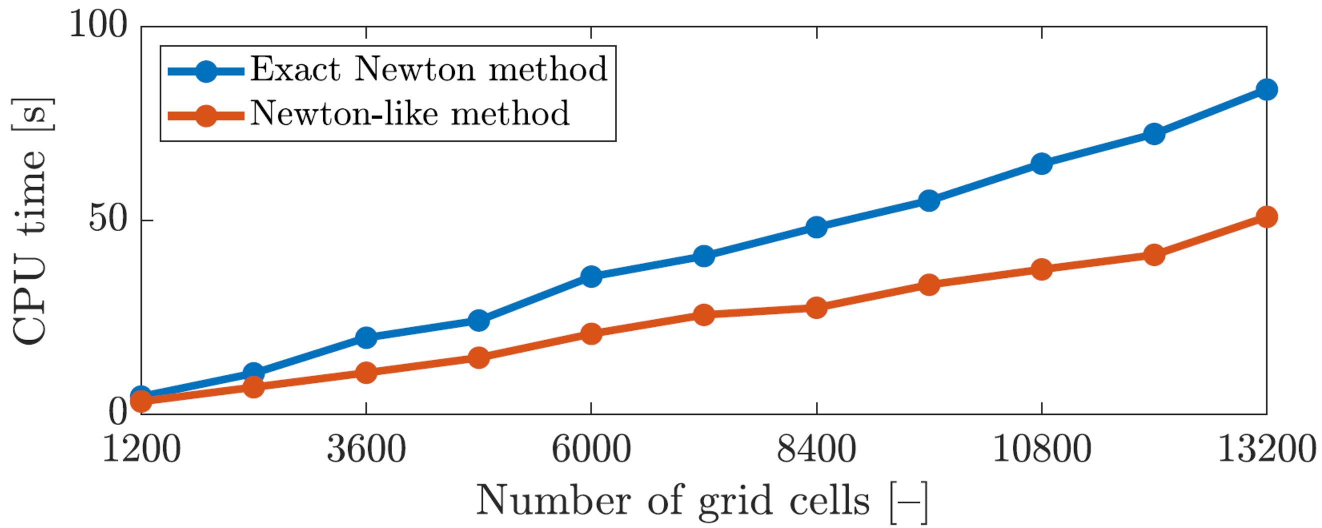
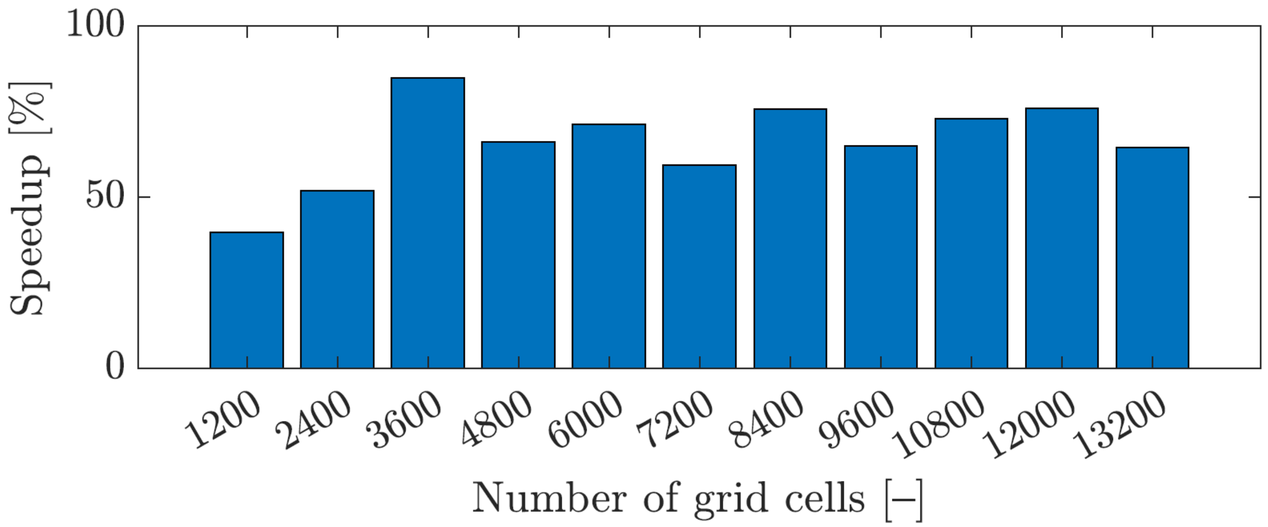


Fig. 3. Simulation of CO_2 injection over 5 years (1,200 grid cells). Top row: Oil saturation. Bottom row: Pressure in MPa.

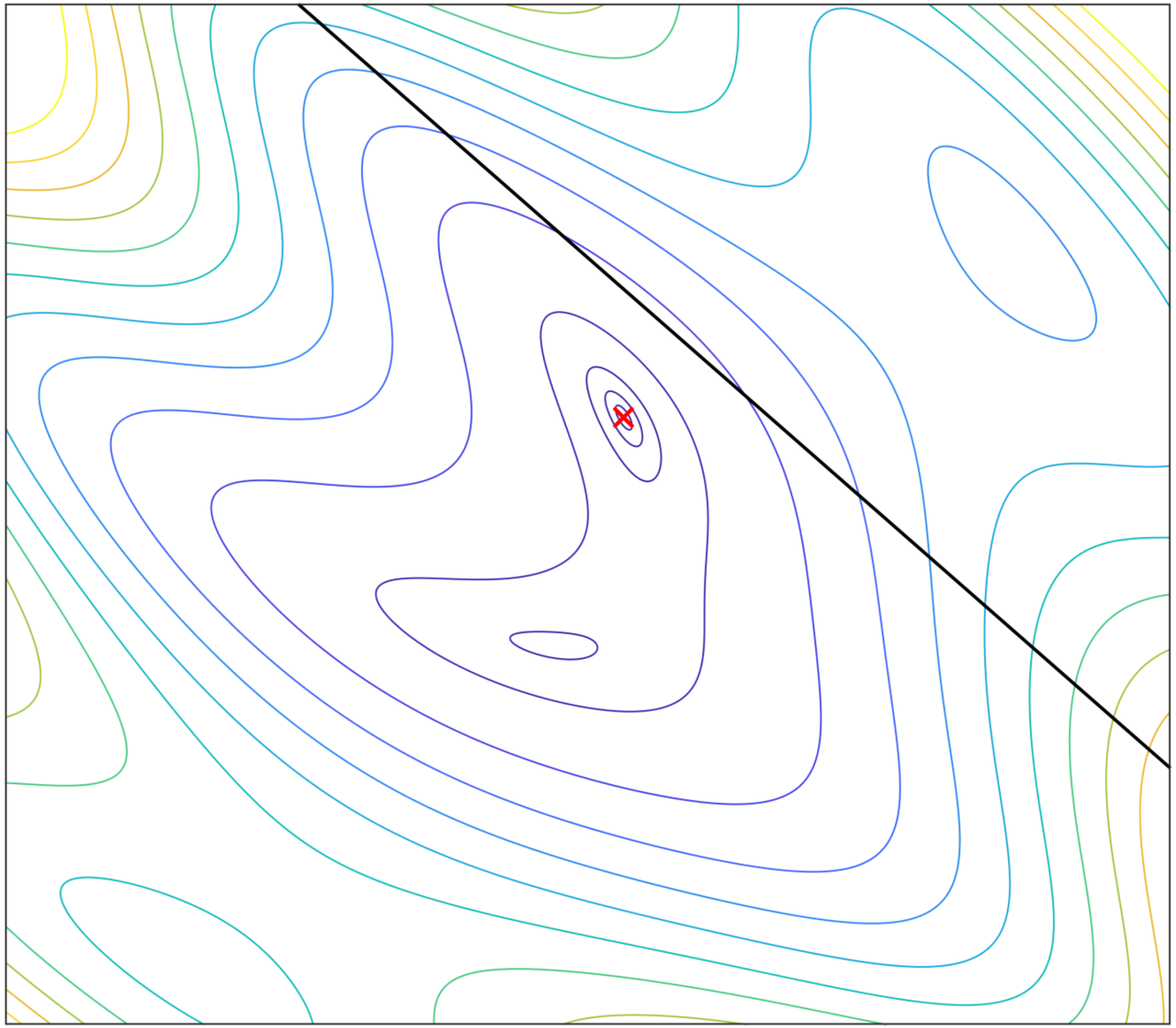
Numerical example of CO_2 storage



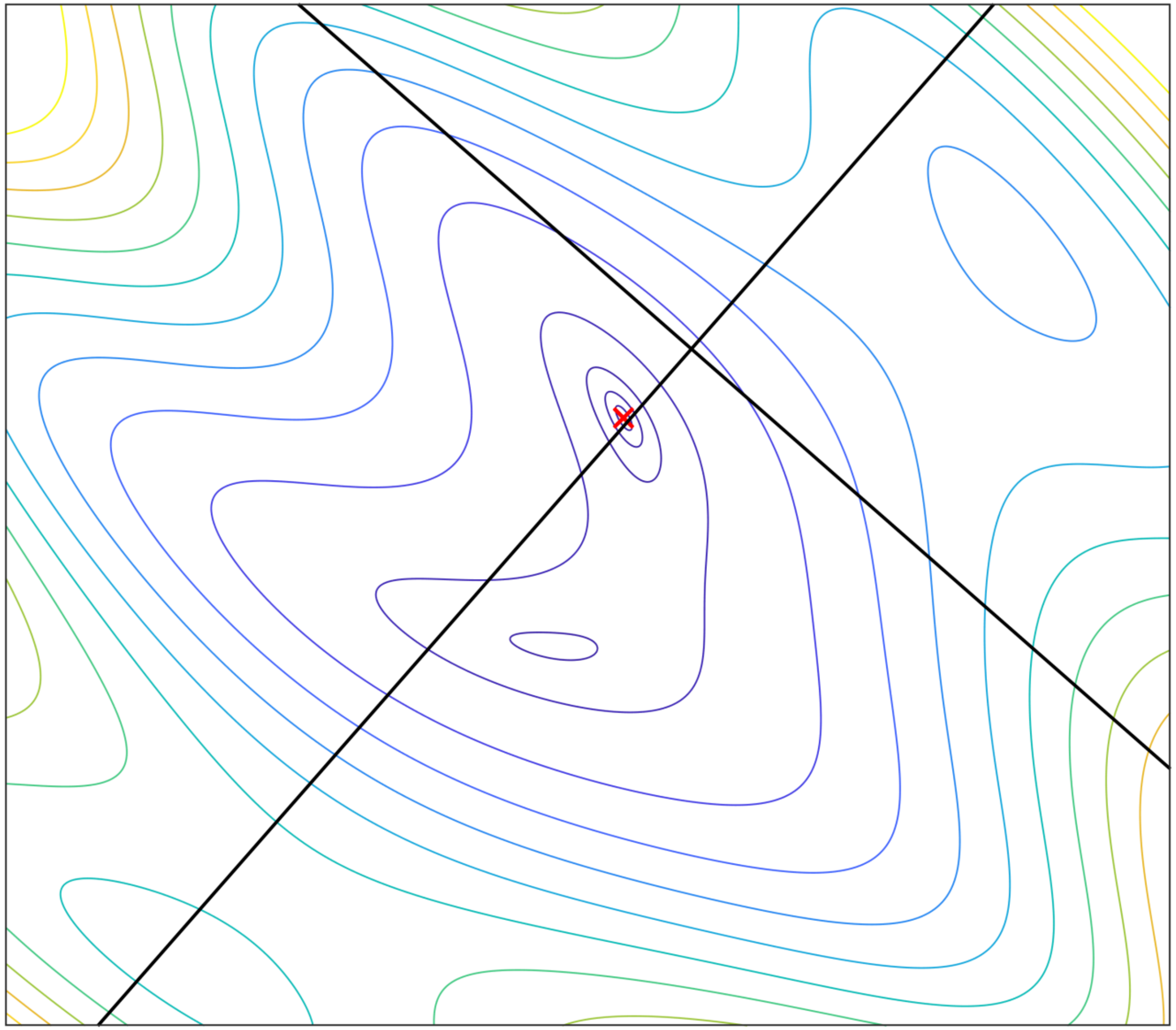


Future work

Iterative improvement of subspace



Iterative improvement of subspace



- Increase subspace
- Cycle subspaces

Other directions

Consider direct methods for numerical optimal control (single-shooting, multiple-shooting, fully simultaneous methods)

Create initial guess with Euler's explicit method

Consider the outputs to be the (norm of) the residual function or a linearization of it

Transport processes:
Exploit that most states only change in short periods of time

Summary

Combine numerical methods
and model reduction techniques

Apply to numerical solution of IVPs

Preliminary results:
Further developments are necessary

Thank you