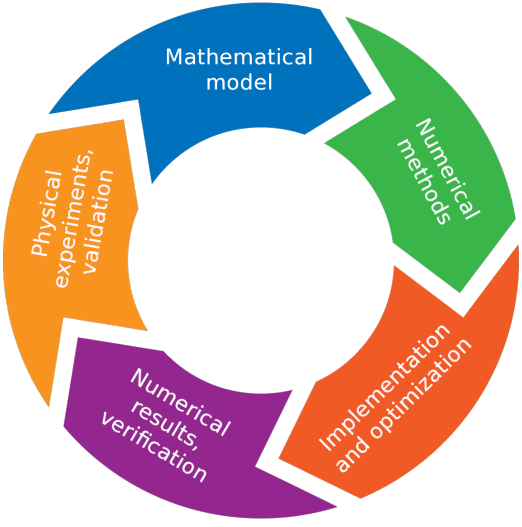
Development of Numerical Solvers in Template Numerical Library and Their Applications

Jakub Klinkovský, Tomáš Oberhuber, Radek Fučík, Pavel Eichler, et al.

Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague

Current Problems in Numerical Analysis
October 18, 2024

Overview


Mathematical model
Numerical methods
Implementation and optimization
Numerical results, verification
Physical experiments, validation

Template Numerical Library

- TNL = Template Numerical Library **(ŠNEK = Šablonová NumErická Knihovna)**
- Open-source **high performance computing** library written in C++
- **Unified high-level interface** for modern parallel architectures – CPUs, GPUs, MPI
- **Building blocks for numerical solvers:**
  - common parallel algorithms (reduction, scan, sort)
  - algorithms and data structures for linear algebra
  - numerical meshes, tools for pre-processing and post-processing

Web: `https://tnl-project.org/` GitLab: `https://gitlab.com/tnl-project/tnl`

T. Oberhuber, J. Klinkovský, R. Fučík, *TNL: Numerical library for modern parallel architectures*. Acta Polytechnica. 2021, 61(SI), 122-134. ISSN 1805-2363. DOI: 10.14311/AP.2021.61.0122.

TNL Modules

The TNL project comprises several **domain-specific modules** that target mainly computational fluid dynamics:

- **TNL-MHFEM** – implementation of the **mixed-hybrid finite element method**
  - main application: compositional multi-phase flow in porous media
  - can solve general advection-diffusion-reaction PDEs
- **TNL-LBM** – implementation of the **lattice Boltzmann method**
  - validated on incompressible Navier-Stokes, and non-Newtonian flow
- **TNL-SPH** – implementation of the **smoothed-particle hydrodynamics** method
  - validated on common free-surface problems (e.g. dam break)

All aforementioned modules have **full multi-GPU support** based on the common core library.

Other utility modules provide support for the development of the project, e.g. **benchmarks**, **Python bindings**, etc.

TNL Data Structures

TNL provides many data structures:

- **General**
  - one-dimensional `Array` and `Vector`, also `StaticArray`, `StaticVector`, `DistributedArray`, `DistributedVector`
  - all vector types support **expression templates** for efficient vector expressions
  - configurable multidimensional array type (`NDArray`)
- **Matrices**
  - `DenseMatrix`, `SparseMatrix`, and special types like `TridiagonalMatrix`
  - common implementation based on the **segments** abstraction layer
  - sparse layout optimized for GPUs
- **Meshes**
  - `Grid` – structured orthogonal grid
  - `Mesh` – unstructured polyhedral mesh
  - `DistributedGrid`, `DistributedMesh`

Unstructured meshes in TNL

History:

- Initial design and implementation – Vítězslav Žabka
- Generalization, improvements and implementation for GPUs – Jakub Klinkovský
- Publication in ACM Transactions on Mathematical Software
- Extension for polygonal and polyhedral meshes – Ján Bobot (supervised by Jakub Klinkovský)

📃 J. Klinkovský, T. Oberhuber, R. Fučík, V. Žabka, *Configurable open-source data structure for distributed conforming unstructured homogeneous meshes with GPU support*. ACM Transactions on Mathematical Software. 2022, 48(3), 1-33. ISSN 0098-3500. DOI: 10.1145/3536164.

📘 J. Bobot, *Implementation of data structure for polyhedral numerical meshes in TNL*, Master's thesis, Faculty of Information Technology Czech Technical University in Prague, Feb. 2022, `https://dspace.cvut.cz/handle/10467/99479`.

Unstructured meshes in TNL: terminology

Definition (conforming mesh)

Let $\mathcal{M}$ be a mesh. If for all mesh entities $E_1, E_2 \in \mathcal{M}$ the intersection of their closures $\overline{E}_1 \cap \overline{E}_2$ is either an empty set or a mesh entity, then $\mathcal{M}$ is called a conforming mesh.

Definition (unstructured mesh)

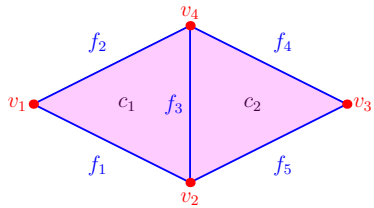
A mesh is called an unstructured mesh, if each vertex of the mesh can be a vertex of non-constant number of cells.

Definition (homogeneous mesh)

If all cells of a mesh have the same shape, then the mesh is said to be **weakly homogeneous**. If all mesh entities of the same dimension have the same shape, then the mesh is said to be **(strongly) homogeneous**.

Unstructured meshes in TNL: internal representation

- Sparse matrix formats for incidence matrices (*Sliced Ellpack*)
- Generation and storage of the dual graph (adjacency matrix)
- Static configuration – data types, cell topology, storage of incidence matrices, dual graph and other internal data structures
- Polygonal and polyhedral entities – dynamic topologies



$$I_{0,1} = \left(\begin{array}{c|ccccc} & f_1 & f_2 & f_3 & f_4 & f_5 \\ \hline v_1 & 1 & 1 & & & \\ v_2 & 1 & & 1 & & 1 \\ v_3 & & & & 1 & 1 \\ v_4 & & 1 & 1 & 1 & \end{array}\right) \quad I_{0,2} = \left(\begin{array}{c|cc} & c_1 & c_2 \\ \hline v_1 & 1 & \\ v_2 & 1 & 1 \\ v_3 & & 1 \\ v_4 & 1 & 1 \end{array}\right) \quad I_{1,2} = \left(\begin{array}{c|cc} & c_1 & c_2 \\ \hline f_1 & 1 & \\ f_2 & 1 & \\ f_3 & 1 & 1 \\ f_4 & & 1 \\ f_5 & & 1 \end{array}\right)$$

$$I_{1,0} = I_{0,1}^T \qquad I_{2,0} = I_{0,2}^T \qquad I_{2,1} = I_{1,2}^T$$

Unstructured meshes in TNL: benchmarking methodology

Cases for comparison:

- 2 algorithms:
  1. calculation of cell measures (areas in 2D, volumes in 3D)
  2. calculation of cell boundary measures (face areas in 3D, edge lengths in 2D)
- 4 mesh configurations – different combinations of data types
- 4 mesh types – triangular (cca 60k cells), tetrahedral (cca 2M cells), polygonal (cca 30k cells), polyhedral (cca 340k cells)
- 1 CPU core, 12 CPU cores (Intel Xeon Gold 6146); GPU (NVIDIA Tesla V100)
- comparison between TNL and MOAB

Quantities used for evaluation:

- computational time CT [ms]
- effective bandwidth EBW [GB/s] – calculated based on *useful data size* for each mesh
- CPU speed-up Sp_ℓ and GPU speed-up GSp_ℓ

Benchmark 1 – polygonal and polyhedral meshes

Table: **Calculation of cell measures** on the finest polygonal mesh $2D_5^*$ and finest polyhedral mesh $3D_5^*$. Each row corresponds to different data types for `Real`, `GlobalIndex` and `LocalIndex` in the mesh configuration, which are recorded in the triplet in the "types" column: `f` (`float`), `d` (`double`), `s` (`short int`), `i` (`int`), and `l` (`long int`).

			1 th.		12 threads			GPU			
		types	CT	EBW	CT	EBW	Sp_{12}	CT	EBW	GSp_1	GSp_{12}
TNL	$2D_5^*$	f, i, s	0.240	5.8	0.035	40	6.8	0.012	115	19.9	2.9
		f, l, i	0.236	9.2	0.031	69	7.5	0.013	171	18.7	2.5
		d, i, s	0.295	6.7	0.040	50	7.5	0.013	148	22.0	2.9
		d, l, i	0.292	9.5	0.038	73	7.7	0.014	196	20.7	2.7
	$3D_5^*$	f, i, s	87.438	1.1	9.868	10	8.9	2.861	34	30.6	3.4
		f, l, i	119.207	1.4	14.333	12	8.3	4.004	42	29.8	3.6
		d, i, s	106.037	1.2	11.817	10	9.0	3.410	36	31.1	3.5
		d, l, i	133.311	1.5	16.329	12	8.2	4.437	44	30.0	3.7
MOAB	$2D_5^*$	d, l, i	2.664	1.0	2.416	1	1.1				
	$3D_5^*$	d, l, i	686.646	0.3	439.212	0	1.6				

Benchmark 2 – polygonal and polyhedral meshes

Table: **Calculation of cell boundary measures** on the finest polygonal mesh $2D_5^*$ and finest polyhedral mesh $3D_5^*$. Each row corresponds to different data types for `Real`, `GlobalIndex` and `LocalIndex` in the mesh configuration, which are recorded in the triplet in the "types" column: `f` (`float`), `d` (`double`), `s` (`short int`), `i` (`int`), and `l` (`long int`).

			1 th.		12 threads			GPU			
		types	CT	EBW	CT	EBW	Sp_{12}	CT	EBW	GSp_1	GSp_{12}
TNL	$2D_5^*$	f, i, s	1.344	2.1	0.150	19	9.0	0.013	223	105.4	11.7
		f, l, i	1.337	3.3	0.148	30	9.0	0.013	332	99.2	11.0
		d, i, s	1.379	2.9	0.152	27	9.1	0.014	289	98.4	10.8
		d, l, i	1.353	4.2	0.153	37	8.9	0.014	398	94.8	10.7
	$3D_5^*$	f, i, s	87.859	1.3	7.772	15	11.3	0.369	320	238.2	21.1
		f, l, i	93.225	2.1	7.973	24	11.7	0.511	379	182.3	15.6
		d, i, s	109.202	1.5	9.151	18	11.9	0.489	329	223.4	18.7
		d, l, i	114.673	2.1	9.307	25	12.3	0.653	362	175.5	14.2
MOAB	$2D_5^*$	d, l, i	46.439	0.1	19.438	0	2.4				
	$3D_5^*$	d, l, i	4768.330	0.0	1459.910	0	3.3				

Algorithms in TNL

Linear algebra:

- Common matrix operations, many are in development
- Optimized sparse matrix–vector multiplication (SpMV) on all architectures
- Solvers for sparse linear systems
  - 7 iterative methods
  - 3 preconditioners (only 1 works on GPUs)
  - wrappers for Hypre, Ginkgo and UMFPACK (direct solver)

ODE solvers:

- 1st order (Euler, Midpoint), 2nd order (Heun, Ralston, Fehlberg), 3rd order (Kutta, Heun, Van der Houwen/Wray, Ralston, SSPRK3, Bogacki-Shampin), 4th order (Runge-Kutta, Ralston, Runge-Kutta-Merson), 5th order (Cash-Karp, Dormand-Prince, Fehlberg), Matlab (ode1, ode2, ode23, ode45), including methods with adaptive choice of time step.

Optimization methods:

- Gradient descent, Adagrad, RMSprop, momentum methods.

TNL-MHFEM: Mixed-Hybrid Finite Element Method

The numerical scheme is implemented for a PDE system written in the **NumDwarf** form

$$\sum_{j=1}^{n} N_{i,j} \frac{\partial Z_j}{\partial t} + \sum_{j=1}^{n} \boldsymbol{u}_{i,j} \cdot \nabla Z_j + \nabla \cdot \left[m_i \left(- \sum_{j=1}^{n} \mathbf{D}_{i,j} \nabla Z_j + \boldsymbol{w}_i \right) + \sum_{j=1}^{n} Z_j \boldsymbol{a}_{i,j} \right] + \sum_{j=1}^{n} r_{i,j} Z_j = f_i$$

for $i \in \{1, \dots, n\}$, where:

- $\boldsymbol{Z} = [Z_1, \dots, Z_n]^T$ are the unknown functions of $\boldsymbol{x}$ and t,
- $\boldsymbol{x} \in \Omega \subset \mathbb{R}^D$, where $D \in \{1, 2, 3\}$ is the spatial dimension, and $t \in [0, t_{\max}]$,
- $\mathbf{N} = [N_{i,j}]_{i,j=1}^n$, $\boldsymbol{u} = [\boldsymbol{u}_{i,j}]_{i,j=1}^n$, $\boldsymbol{m} = [m_i]_{i=1}^n$, $\mathbf{D} = [\mathbf{D}_{i,j}]_{i,j=1}^n$, $\boldsymbol{w} = [\boldsymbol{w}_i]_{i=1}^n$, $\boldsymbol{a} = [\boldsymbol{a}_{i,j}]_{i,j=1}^n$, $\boldsymbol{r} = [r_{i,j}]_{i,j=1}^n$, $\boldsymbol{f} = [f_i]_{i=1}^n$ are general problem-specific coefficients.

MHFEM-based numerical scheme for NumDwarf

- Spatial discretization: **mixed-hybrid finite element method** and **discontinuous Galerkin** method
  - unstructured simplex mesh $\mathcal{K}_h$
  - only the lowest-order Raviart–Thomas–Nédelec space $\mathbf{RTN}_0(\mathcal{K}_h)$
- Temporal discretization: **Euler method**
  - **method of frozen coefficients** for linearization in time (semi-implicit)
  - **explicit advection** terms $\boldsymbol{u}_{i,j}$ and $\boldsymbol{a}_{i,j}$, **implicit diffusion** term $\boldsymbol{v}_i = -\sum_{j=1}^{n} \mathbf{D}_{i,j} \nabla Z_j + \boldsymbol{w}_i$
- Stabilization:
  - **mass-lumping**
  - upwind schemes for advective terms: either **explicit upwind** or **implicit upwind**
- Properties:
  - locally conservative scheme, solution of one linear system per time step
  - allows to solve problems with vanishing diffusion (explicit upwind variant)

R. Fučík, J. Klinkovský, J. Solovský, T. Oberhuber, J. Mikyška, *Multidimensional mixed-hybrid finite element method for compositional two-phase flow in heterogeneous porous media and its parallel implementation on GPU*. Computer Physics Communications. 2019, 238 165-180. DOI: 10.1016/j.cpc.2018.12.004.

Two-phase incompressible flow in porous media

$$\Phi \rho_w \frac{\partial S_w}{\partial t} - \rho_w \nabla \cdot (\lambda_w K(\nabla p_w - \rho_w \boldsymbol{g})) = 0,$$
$$\Phi \rho_n \frac{\partial S_n}{\partial t} - \rho_n \nabla \cdot (\lambda_n K(\nabla p_n - \rho_n \boldsymbol{g})) = 0,$$

where:

- Φ [-] is the material porosity, K [m^2] is the material permeability, $\boldsymbol{g}$ [$\mathrm{m\,s}^{-2}$] is the gravitational acceleration, and ρ_w, ρ_n [$\mathrm{kg\,m}^{-3}$] are constant phase densities,
- subscripts $\alpha \in \{w, n\}$ denote symbols related to the **wetting** and **non-wetting** phase, respectively,
- **algebraic constraints** (with Brooks–Corey and Burdine models):

$$p_n - p_w = p_c,$$
$$S_w + S_n = 1,$$
$$p_c(S_w) = (S_{w,e})^{-\frac{1}{\lambda_*}} p_*,$$
$$S_{\alpha,e} = \frac{S_\alpha - S_{\alpha,r}}{1 - S_{w,r} - S_{n,r}},$$

$$k_{r,w}(S_w) = (S_{w,e})^{3+\frac{2}{\lambda_*}},$$
$$k_{r,n}(S_n) = (S_{n,e})^2 \left(1 - (1 - S_{n,e})^{1+\frac{2}{\lambda_*}}\right),$$
$$\lambda_\alpha = k_{r,\alpha}(S_\alpha)/\mu_\alpha.$$

NumDwarf coefficients for two-phase flow in porous media

$n = 2$, $\{p_w, p_n\}$ are selected as the primary unknowns, i.e., $\boldsymbol{Z} = [p_w, p_n]^T$, and the coefficients are set as follows:

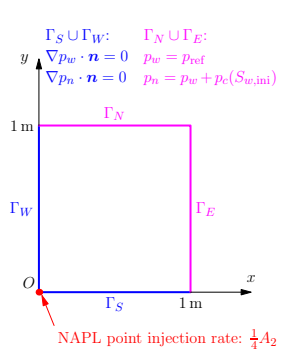
$$\mathbf{N} = \begin{pmatrix} -\Phi \frac{\mathrm{d}S_w}{\mathrm{d}p_c} & \Phi \frac{\mathrm{d}S_w}{\mathrm{d}p_c} \\ \Phi \frac{\mathrm{d}S_w}{\mathrm{d}p_c} & -\Phi \frac{\mathrm{d}S_w}{\mathrm{d}p_c} \end{pmatrix}, \quad \boldsymbol{u} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \qquad \boldsymbol{m} = \begin{pmatrix} \frac{\lambda_w}{\lambda_t} \\ \frac{\lambda_n}{\lambda_t} \end{pmatrix},$$

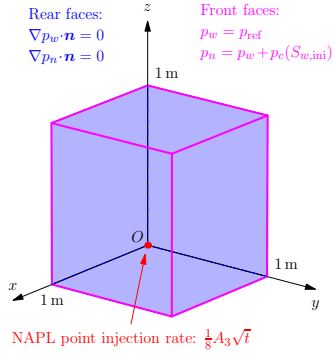
$$\mathbf{D} = \begin{pmatrix} \lambda_t K \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \lambda_t K \mathbf{I} \end{pmatrix}, \qquad \boldsymbol{w} = \begin{pmatrix} \lambda_t \rho_w K \boldsymbol{g} \\ \lambda_t \rho_n K \boldsymbol{g} \end{pmatrix}, \quad \boldsymbol{a} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \boldsymbol{r} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

where $\lambda_t = \lambda_w + \lambda_n$.

Generalized McWhorter–Sunada problem

- Special case of incompressible two-phase flow in porous media without gravity with known semi-analytical solution
- Original solutions with $D \in \{1, 2\}$ generalized for $D \in \mathbb{N}$ by R. Fučík et al.
- Volumetric injection rate $Q_0 = Q_0(t) = A_D t^{\frac{D-2}{2}}$, numerical and semi-analytical solutions compared for $D \in \{2, 3\}$





Verification results – experimental order of convergence

Table: Results of the numerical analysis using the L_1 and L_2 norms of E_{h,S_n} for the 2D problem on a series of triangular discretizations.

Id.	h [m]	τ [s]	$\|E_{h,S_n}\|_1$	$EOC_{S_n,1}$	$\|E_{h,S_n}\|_2$	$EOC_{S_n,2}$
$2D_1^{\triangle}$	6.71×10^{-2}	454.55	1.54×10^{-2}		3.25×10^{-2}	
$2D_2^{\triangle}$	3.49×10^{-2}	145.99	8.14×10^{-3}	0.97	1.89×10^{-2}	0.84
$2D_3^{\triangle}$	1.64×10^{-2}	44.64	4.44×10^{-3}	0.80	1.19×10^{-2}	0.61
$2D_4^{\triangle}$	8.73×10^{-3}	13.44	2.41×10^{-3}	0.97	7.79×10^{-3}	0.67
$2D_5^{\triangle}$	4.23×10^{-3}	5.00	1.29×10^{-3}	0.86	4.90×10^{-3}	0.64

Note: τ is a pre-selected constant time step for which the lowest error with given h was obtained.

Verification results – experimental order of convergence

Table: Results of the numerical analysis using the L_1 and L_2 norms of E_{h,S_n} for the 3D problem on a series of tetrahedral discretizations.

Id.	h [m]	τ [s]	$\|E_{h,S_n}\|_1$	$EOC_{S_n,1}$	$\|E_{h,S_n}\|_2$	$EOC_{S_n,2}$
$3D_1^{\triangle}$	2.13×10^{-1}	833.33	1.15×10^{-2}		3.48×10^{-2}	
$3D_2^{\triangle}$	1.27×10^{-1}	571.43	8.02×10^{-3}	0.69	2.52×10^{-2}	0.62
$3D_3^{\triangle}$	6.29×10^{-2}	232.56	4.41×10^{-3}	0.86	1.49×10^{-2}	0.75
$3D_4^{\triangle}$	3.48×10^{-2}	101.01	2.40×10^{-3}	1.03	8.62×10^{-3}	0.93
$3D_5^{\triangle}$	1.84×10^{-2}	25.00	1.26×10^{-3}	1.01	5.48×10^{-3}	0.71

Note: τ is a pre-selected constant time step for which the lowest error with given h was obtained.

Computational benchmark on CPU (Intel Xeon Gold 6146)

			TNL						Hypre		
			OpenMP			MPI			MPI		
Cores	CPUs	Nodes	CT	Sp	Eff	CT	Sp	Eff	CT	Sp	Eff
1			188243.0	1.0	1.00	188706.0	1.0	1.00	37991.2	1.0	1.00
2			102074.0	1.8	0.92	93659.1	2.0	1.01	21170.2	1.8	0.90
4			55937.6	3.4	0.84	49553.0	3.8	0.95	11252.2	3.4	0.84
6			40796.4	4.6	0.77	35594.3	5.3	0.88	7798.1	4.9	0.81
8			32026.3	5.9	0.73	28958.6	6.5	0.81	6085.4	6.2	0.78
12	1	1/2	26369.7	7.1	0.59	23839.0	7.9	0.66	4708.8	8.1	0.67
24	2	1	15695.0	12.0	0.50	12184.2	15.5	0.65	2485.0	15.3	0.64
48	4	2				6029.0	31.3	0.65	1249.1	30.4	0.63
72	6	3				4054.7	46.5	0.65	880.2	43.2	0.60
96	8	4				2974.5	63.4	0.66	592.3	64.1	0.67
120	10	5				2483.0	76.0	0.63	471.2	80.6	0.67
144	12	6				2000.0	94.4	0.66	415.8	91.4	0.63
168	14	7				1607.7	117.4	0.70	372.2	102.1	0.61
192	16	8				1380.4	136.7	0.71	310.7	122.3	0.64
216	18	9				1209.6	156.0	0.72	277.5	136.9	0.63
240	20	10				1082.0	174.4	0.73	240.3	158.1	0.66
264	22	11				974.9	193.6	0.73	251.5	151.0	0.57
288	24	12				892.5	211.4	0.73	223.9	169.7	0.59
312	26	13				901.8	209.3	0.67	202.9	187.2	0.60
336	28	14				851.9	221.5	0.66	201.9	188.2	0.56

Computational benchmark on GPU (NVIDIA Tesla V100)

Note: TNL solver is Jacobi-preconditioned BiCGstab,
Hypre solver is BoomerAMG-preconditioned BiCGstab

GPU computations were performed on 1 node only (1-4 GPUs)

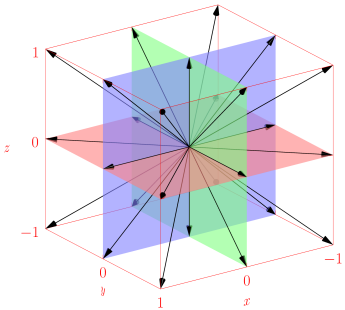
	TNL						Hypre					
	$2D_5^{\triangle}$			$3D_5^{\triangle}$			$2D_5^{\triangle}$			$3D_5^{\triangle}$		
GPUs	CT	Sp	Eff	CT	Sp	Eff	CT	Sp	Eff	CT	Sp	Eff
1	528.6	1.0	1.00	2654.8	1.0	1.00	389.8	1.0	1.00	2014.5	1.0	1.00
2	566.1	0.9	0.47	1415.4	1.9	0.94	500.6	0.8	0.39	1233.1	1.6	0.82
3	642.5	0.8	0.27	996.7	2.7	0.89	634.1	0.6	0.20	868.9	2.3	0.77
4	709.7	0.7	0.19	793.3	3.3	0.84	726.8	0.5	0.13	704.2	2.9	0.72

TNL-LBM: Lattice Boltzmann method in a nutshell

Lattice Boltzmann transport equation:

$$f_q(\boldsymbol{x} + \Delta t \boldsymbol{\xi}_q, t + \Delta t) - f_q(\boldsymbol{x}, t) = \Delta t \left(\mathcal{C}_q(\boldsymbol{x}, t) + \mathcal{S}_q(\boldsymbol{x}, t) \right)$$

- f_q ...q-th discrete distribution function
- $\boldsymbol{\xi}_q$ ...q-th discrete velocity
- $\mathcal{C}_q$ ...**collision operator** (here cumulant op.)
- $\mathcal{S}_q$ ...forcing term
- $q \in \{1, 2, 3, \ldots, Q\}$



Collision step:

- $f_q^*(\boldsymbol{x}, t) = f_q(\boldsymbol{x}, t) + \Delta t \left(\mathcal{C}_q(\boldsymbol{x}, t) + \mathcal{S}_q(\boldsymbol{x}, t) \right)$

Streaming step:

- $f_q(\boldsymbol{x} + \Delta t \boldsymbol{\xi}_q, t + \Delta t) = f_q^*(\boldsymbol{x}, t)$

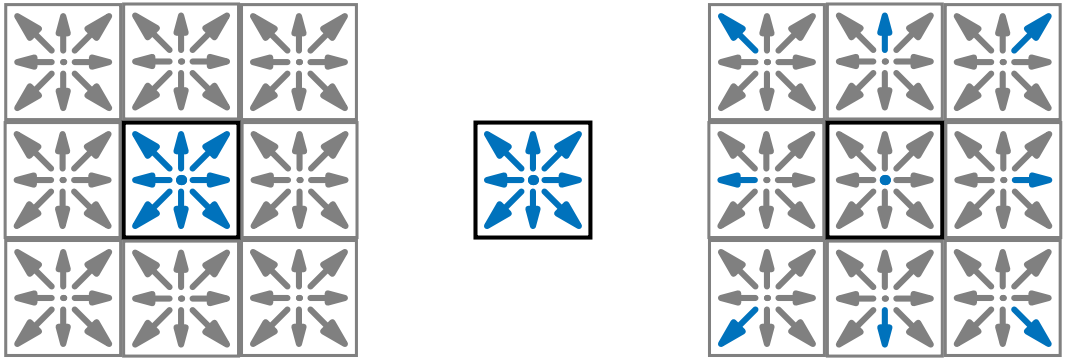
Macroscopic quantities (raw moments):

- $\rho(\boldsymbol{x}, t) = \sum_{q=1}^{Q} f_q(\boldsymbol{x}, t)$
- $\rho(\boldsymbol{x}, t) \boldsymbol{v}(\boldsymbol{x}, t) = \sum_{q=1}^{Q} f_q(\boldsymbol{x}, t) \boldsymbol{\xi}_q + \frac{1}{2} \Delta t \hat{\boldsymbol{F}}(\boldsymbol{x}, t)$
- $+$ energy, stress, ...

Overview of streaming schemes

- A-B pattern – uses two arrays A and B **(illustration on following slides)**
  - push scheme
  - pull scheme
- A-A pattern – uses only one array A **(illustration on following slides)**
- Esoteric twist, Esoteric push, Esoteric pull, compact LRnLA, etc. **(not implemented in TNL-LBM)**

Streaming with A-B pattern – push scheme



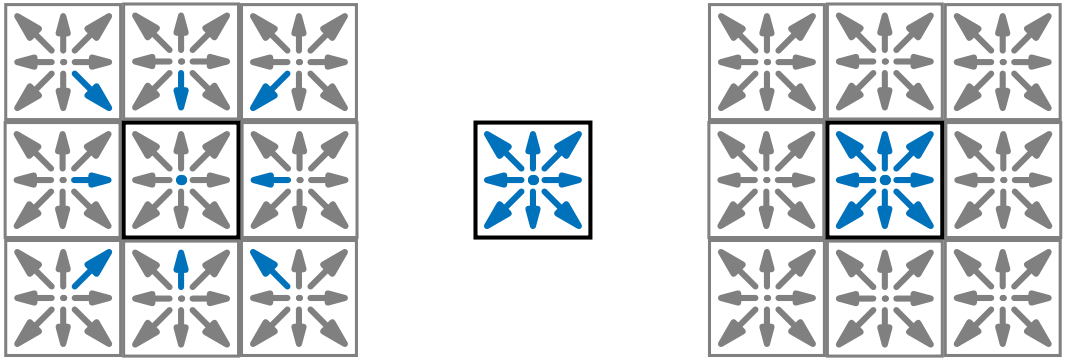
(a) Values read from the global memory (array A)

(b) Values stored in the registers

(c) Values stored into the global memory (array B)

Figure: Illustration of the **push** streaming scheme with the **A-B pattern**.

Streaming with A-B pattern – pull scheme



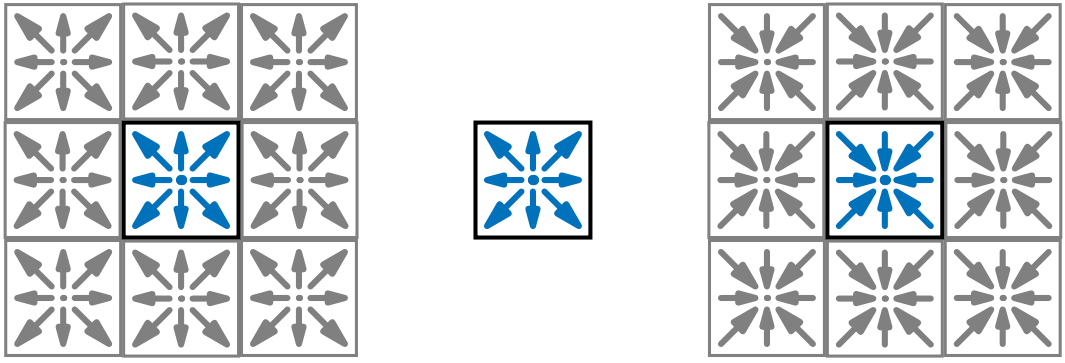
(a) Values read from the global memory (array A)

(b) Values stored in the registers

(c) Values stored into the global memory (array B)

Figure: Illustration of the **pull** streaming scheme with the **A-B pattern**.

Streaming with A-A pattern – even iteration



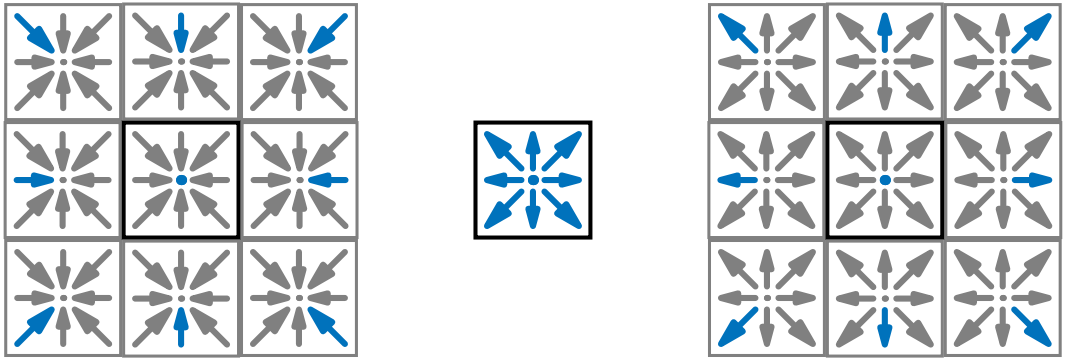
(a) Values read from the global memory (array A)

(b) Values stored in the registers

(c) Values stored into the global memory (array A)

Figure: Illustration of the **A-A pattern** streaming scheme in the **even iteration**.

Streaming with A-A pattern – odd iteration



(a) Values read from the global memory (array A)

(b) Values stored in the registers

(c) Values stored into the global memory (array A)

Figure: Illustration of the **A-A pattern** streaming scheme in the **odd iteration**.

Computational algorithm of LBM

1. Initialization (read input data, set initial condition, etc.)
2. While the final time is not reached, do **for all lattice sites in parallel**:
   1. Streaming step before collision (pull distribution functions from global memory)
   2. Compute macroscopic quantities (ρ, $\boldsymbol{v}$, etc.)
   3. Handle boundary conditions (boundary sites only)
   4. Collision step – evaluate the collision operator and forcing term
   5. Streaming step after collision (push distribution functions to global memory)
   6. Output macroscopic quantities to global memory

Notes:

- steps 1 to 6 inside the loop are called **LBM iteration**
- streaming steps before/after collision depend on the streaming pattern

Optimized LBM algorithm with domain decomposition

1. Initialization (read input data, set initial condition, etc.)
2. Copy distribution functions on the interfaces between neighboring subdomains
3. While the final time is not reached:
   1. For **all subdomains**, start processing **lattice sites next to the interfaces with other subdomains**
   2. For **all subdomains**, start processing **remaining lattice sites**
   3. For **all subdomains**, wait until lattice sites next to the interfaces are processed
   4. For **all subdomains**, copy distribution functions across the interfaces between subdomains
   5. For **all subdomains**, wait until the remaining lattice sites are processed

Implemented optimizations:

- Pipelining for asynchronous communication of relevant distribution functions (9 in each direction)
- Avoiding buffers in communication (needs specific ordering of data in multidimensional arrays)
- Direct GPU-GPU copies via "CUDA-aware" MPI

Karolina supercomputer – hardware specifications

Accelerated compute nodes in the Karolina supercomputer:

Number of nodes	72
Processors per node	2
CPU model	AMD EPYC 7763 (64 cores, 2.45-3.5 GHz)
Memory per node	1024 GB DDR4 3200 MT/s
Accelerators per node	8
GPU model	NVIDIA A100 (40 GB HBM2 memory)
Intra-node connection	NVLink 3.0 (12 sub-links, 25 GB/s per sub-link per direction)
Inter-node connection	$4\times$ 200 Gb/s InfiniBand ports

The supercomputer is operated by IT4Innovations (`https://www.it4i.cz/`).

Note: NVIDIA GPUDirect technology was not fully functional when computing the benchmark.

LBM: strong scaling on the Karolina supercomputer

		single precision			double precision		
N_{nodes}	N_{ranks}	GLUPS	Sp	Eff	GLUPS	Sp	Eff
1	1	5.2	1.0	1.00	2.8	1.0	1.00
1	2	10.2	2.0	0.98	5.5	2.0	1.00
1	4	20.4	3.9	0.98	11.1	4.0	1.01
1	8	41.1	7.9	0.99	22.3	8.1	1.01
2	16	80.4	15.5	0.97	44.1	16.0	1.00
4	32	145.2	28.0	0.87	85.5	31.0	0.97
8	64	258.6	49.8	0.78	153.7	55.7	0.87
16	128	301.1	58.0	0.45	225.1	81.6	0.64

- Lattice size: $512 \times 512 \times 512$
- Each MPI rank uses its own GPU
- GLUPS – billions of lattice updates per second
- Sp – speed-up relative to 1 GPU
- $Eff = Sp/N_{\text{ranks}}$ – parallel efficiency

Note: efficiency limited by 1D domain decomposition (not a problem for weak scaling)

LBM: weak scaling with 1D domain expansion

		single precision			double precision		
N_{nodes}	N_{ranks}	GLUPS	Sp	Eff	GLUPS	Sp	Eff
1	1	5.2	1.0	1.00	2.8	1.0	1.00
1	2	10.2	2.0	0.99	5.5	2.0	0.99
1	4	20.4	4.0	0.99	11.0	4.0	0.99
1	8	40.9	7.9	0.99	22.0	8.0	0.99
2	16	81.8	15.8	0.99	44.1	15.9	0.99
4	32	163.4	31.7	0.99	88.2	31.8	0.99
8	64	326.8	63.4	0.99	176.4	63.6	0.99
16	128	653.9	126.7	0.99	352.8	127.3	0.99

- Lattice size: $256N_{\text{ranks}} \times 256 \times 256$
- Each MPI rank uses its own GPU
- GLUPS – billions of lattice updates per second
- Sp – speed-up relative to 1 GPU
- $Eff = Sp/N_{\text{ranks}}$ – parallel efficiency

LBM: weak scaling with 3D domain expansion

		single precision				double precision			
N_{nodes}	N_{ranks}	N_x	GLUPS	Sp	Eff	N_x	GLUPS	Sp	Eff
1	1	512	5.2	1.0	1.00	256	2.8	1.0	1.00
1	2	645	9.5	1.8	0.91	323	5.3	1.9	0.95
1	4	813	19.0	3.7	0.92	406	10.6	3.8	0.96
1	8	1024	40.9	7.9	0.98	512	22.4	8.1	1.01
2	16	1290	74.3	14.3	0.89	645	40.4	14.6	0.91
4	32	1625	153.8	29.6	0.93	813	82.9	30.0	0.94
8	64	2048	324.0	62.4	0.98	1024	176.2	63.7	1.00
16	128	2580	604.9	116.5	0.91	1290	301.3	108.9	0.85

Lattice sizes:
$N_x = N_y = N_z = \lfloor 512 \sqrt[3]{N_{\text{ranks}}} \rceil$ in single precision,
$N_x = N_y = N_z = \lfloor 256 \sqrt[3]{N_{\text{ranks}}} \rceil$ in double precision

Coupled LBM-MHFEM Computational Approach

Incompressible Navier–Stokes equations (in $\Omega_1 \times (0, t_{\max})$):

$$\nabla \cdot \boldsymbol{v} = 0, \tag{1a}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} = -\frac{1}{\rho} \nabla p + \nu \Delta \boldsymbol{v}, \tag{1b}$$

$\boldsymbol{v}$ **fluid velocity,**
ρ fluid density,
p fluid pressure,
ν kinematic viscosity of the fluid,

Generic advection–diffusion equation (passive transport in $\Omega_2 \subset \Omega_1$):

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \boldsymbol{v} - D_0 \nabla \phi) = 0, \tag{2a}$$

Or in non-conservative form:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi - \nabla \cdot (D_0 \nabla \phi) = 0. \tag{2b}$$

$\boldsymbol{v}$ **fluid velocity,**
ϕ generic physical quantity (e.g., temperature or mass concentration),
D_0 diffusion coefficient.

Coupled LBM-MHFEM approach

- NSE – lattice Boltzmann method (LBM)
- ADE – mixed-hybrid finite element method (MHFEM)
- One-way coupling via the velocity field $\boldsymbol{v}$

Which formulation of the ADE should be used?

- $\nabla \cdot \boldsymbol{v} = 0$ is not satisfied exactly by the LBM solver (weak compressibility)
- The interpolated velocity field is not locally conservative
- Numerical schemes for the **conservative and non-conservative variants are not equivalent**
- MHFEM discretization of the **non-conservative transport equation** includes a term that compensates for non-zero discrete velocity divergence

LBM-MHFEM: coupling details

Interpolation of the velocity $\boldsymbol{v}$:

- Trilinear or tricubic interpolation
- Evaluation at **cell side centers** (not cell centers) – to satisfy balancing requirements imposed by the MHFEM discretization

Time stepping:

- MHFEM allows to use larger time steps than LBM
- **Adaptive time-stepping** strategy for MHFEM based on a CFL-like condition

Test problem configuration

- Turbulent air flow in Ω_1 – inflow boundary conditions with synthetic fluctuations
- Inflow value $\phi_{\text{in}} = 1$, initial value $\phi(\boldsymbol{x}, 0) = 1$ for $\boldsymbol{x} \in \Omega_2$
  $\rightarrow$ trivial analytical solution $\phi(\boldsymbol{x}, t) = 1$ for all $\boldsymbol{x} \in \Omega_2$ and $t > 0$
- Three resolutions: RES-A1 ($\delta_x = 8.06$ mm), RES-A2 ($\delta_x = 3.97$ mm), RES-A3 ($\delta_x = 1.97$ mm)
- $t_{\max} = 10$ s

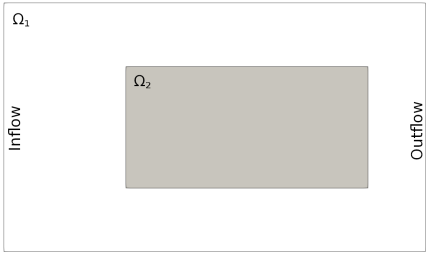


Figure: Schematic configuration of the computational domains Ω_1 and Ω_2 (2D cross-section of a 3D cuboidal channel along the plane $y = 0$).

Test problem – velocity field

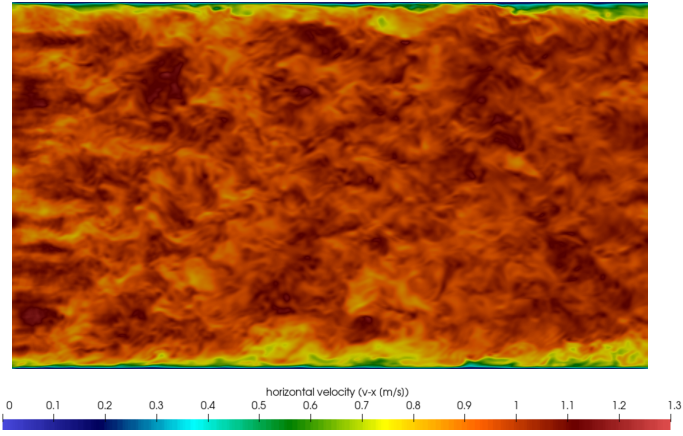


Figure: Horizontal velocity field (v_x) along the plane $y = 0$ in Ω_1, computed in resolution RES-A2.

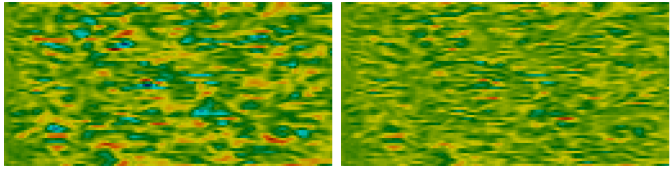
Experimental convergence analysis

Table: Results of the experimental convergence analysis for different formulations and variants of the MHFEM scheme.

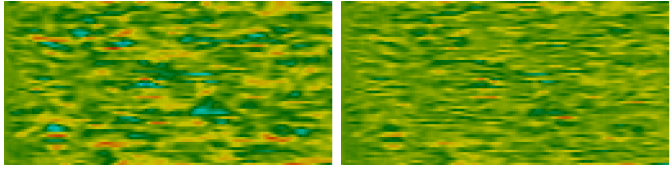
			conservative		non-conservative	
interp.	upwind	resolution	$\|\phi - \phi_h\|_1$	$\|\phi - \phi_h\|_2$	$\|\phi - \phi_h\|_1$	$\|\phi - \phi_h\|_2$
linear	explicit	RES-A1	4.01×10^{-3}	1.11×10^{-2}	1.21×10^{-14}	3.70×10^{-13}
		RES-A2	2.01×10^{-3}	5.71×10^{-3}	5.05×10^{-15}	2.85×10^{-13}
		RES-A3	7.82×10^{-4}	2.28×10^{-3}	8.00×10^{-15}	1.34×10^{-12}
	implicit	RES-A1	3.24×10^{-3}	8.95×10^{-3}	3.62×10^{-13}	2.40×10^{-12}
		RES-A2	1.62×10^{-3}	4.64×10^{-3}	5.80×10^{-14}	2.62×10^{-13}
		RES-A3	6.23×10^{-4}	1.82×10^{-3}	3.97×10^{-14}	2.19×10^{-13}
cubic	explicit	RES-A1	3.25×10^{-3}	8.75×10^{-3}	1.19×10^{-14}	3.56×10^{-13}
		RES-A2	1.31×10^{-3}	3.63×10^{-3}	7.44×10^{-15}	5.01×10^{-13}
		RES-A3	3.98×10^{-4}	1.09×10^{-3}	8.68×10^{-15}	1.45×10^{-12}
	implicit	RES-A1	2.63×10^{-3}	7.08×10^{-3}	5.28×10^{-13}	2.46×10^{-12}
		RES-A2	1.07×10^{-3}	2.96×10^{-3}	5.73×10^{-14}	2.50×10^{-13}
		RES-A3	3.24×10^{-4}	8.83×10^{-4}	3.37×10^{-14}	1.91×10^{-13}

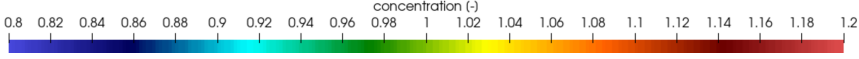
Conservative formulation – transported quantity ϕ (RES-A1)

(a) Linear interp., explicit upwind (b) Cubic interp., explicit upwind



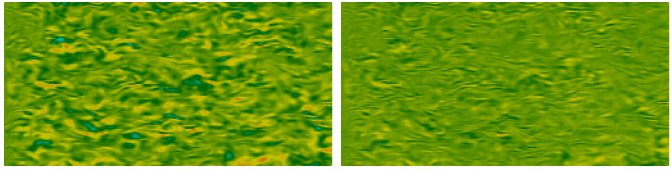
(c) Linear interp., implicit upwind (d) Cubic interp., implicit upwind



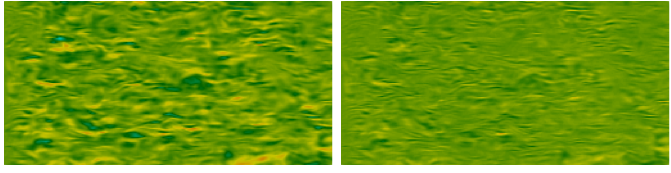


Conservative formulation – transported quantity ϕ (RES-A2)

(a) Linear interp., explicit upwind (b) Cubic interp., explicit upwind



(c) Linear interp., implicit upwind (d) Cubic interp., implicit upwind

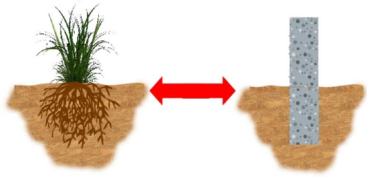


Application: Vapor Transport in Turbulent Air Flow

- Validation of the coupled LBM-MHFEM computational approach on experimental data
- Joint work with Radek Fučík, Andrew C. Trautz[a] and Tissa H. Illangasekare[b]

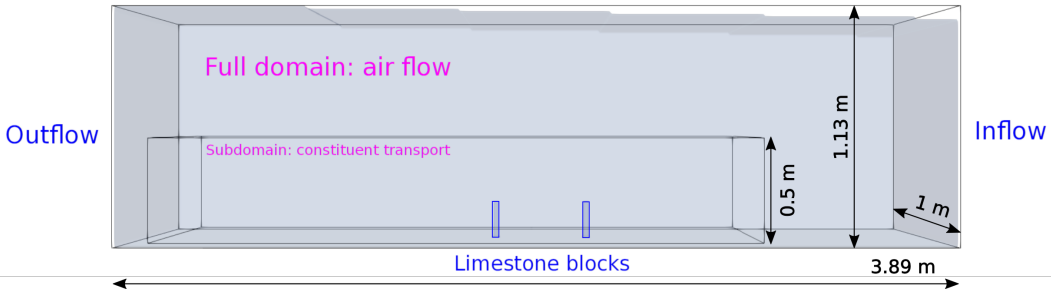
  [a] US Army Engineer Research and Development Center

  [b] Center for Experimental Study of Subsurface Environmental Processes, Colorado School of Mines
- Investigation of environmental effects of land-atmospheric interactions in a climate-controlled, low-speed wind tunnel interfaced with a soil tank (CESEP, Colorado School of Mines, USA)
- Live vegetation approximated with limestone blocks



Computational domain

Only part of the wind tunnel above soil surface; 2 identical blocks; different spacings.



Governing equations: air flow and vapor transport

NSE (air flow in $\Omega_1 \times (0, t_{\max})$):

$$\nabla \cdot \boldsymbol{v} = 0, \tag{3a}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v} = -\frac{1}{\rho} \nabla p + \nu \Delta \boldsymbol{v}, \tag{3b}$$

- $\boldsymbol{v}$ **fluid velocity,**
- ρ fluid density,
- p fluid pressure,
- ν kinematic viscosity of the fluid,

ADE (vapor transport in $\Omega_2 \subset \Omega_1$):

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \boldsymbol{v} - D_0 \nabla \phi) = 0, \tag{4a}$$

Or in non-conservative form:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi - \nabla \cdot (D_0 \nabla \phi) = 0. \tag{4b}$$

- $\boldsymbol{v}$ **fluid velocity,**
- ϕ relative humidity,
- D_0 diffusion coefficient.

Boundary conditions

Walls:

- No-slip condition for velocity
- Fixed value of relative humidity at the bottom

Outflow:

- Fixed pressure value, zero velocity gradient in the normal direction
- Zero gradient of relative humidity (also on top, front, and back sides of Ω_2)

Inflow:

- Mean velocity profile based on the power-law + velocity fluctuations based on synthetic turbulence model
- Mean relative humidity profile based on experimental data

Limestone blocks:

- No-slip condition on all sides except downstream
- Small inflow velocity (approx. 1 mm/s) on the downstream side
- High fixed value of relative humidity is prescribed to model transpiration from the fully-wetted surface of the blocks

Boundary conditions

Walls:

- No-slip condition for velocity
- Fixed value of relative humidity at the bottom

Outflow:

- Fixed pressure value, zero velocity gradient in the normal direction
- Zero gradient of relative humidity (also on top, front, and back sides of Ω_2)

Inflow:

- Mean velocity profile based on the power-law + velocity fluctuations based on synthetic turbulence model
- Mean relative humidity profile based on experimental data

Limestone blocks:

- No-slip condition on all sides except downstream
- Small inflow velocity (approx. 1 mm/s) on the downstream side
- High fixed value of relative humidity is prescribed to model transpiration from the fully-wetted surface of the blocks

Boundary conditions

Walls:

- No-slip condition for velocity
- Fixed value of relative humidity at the bottom

Outflow:

- Fixed pressure value, zero velocity gradient in the normal direction
- Zero gradient of relative humidity (also on top, front, and back sides of Ω_2)

Inflow:

- Mean velocity profile based on the power-law + velocity fluctuations based on synthetic turbulence model
- Mean relative humidity profile based on experimental data

Limestone blocks:

- No-slip condition on all sides except downstream
- Small inflow velocity (approx. 1 mm/s) on the downstream side
- High fixed value of relative humidity is prescribed to model transpiration from the fully-wetted surface of the blocks

Boundary conditions

Walls:

- No-slip condition for velocity
- Fixed value of relative humidity at the bottom

Outflow:

- Fixed pressure value, zero velocity gradient in the normal direction
- Zero gradient of relative humidity (also on top, front, and back sides of Ω_2)

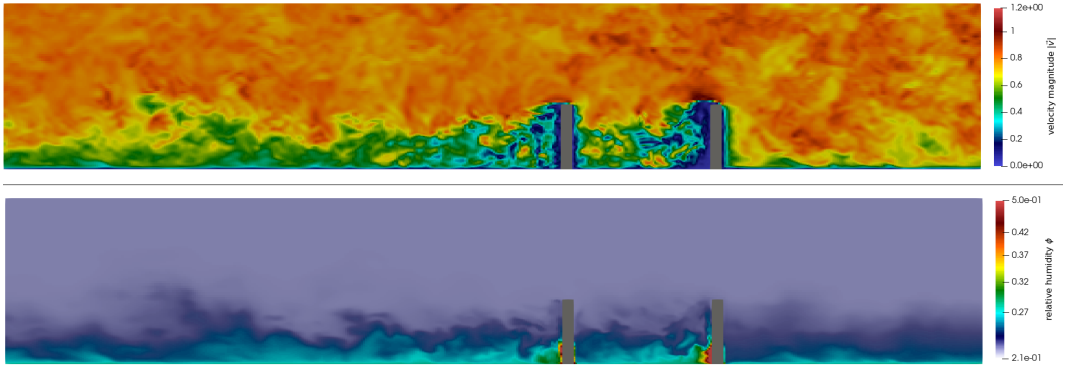
Inflow:

- Mean velocity profile based on the power-law + velocity fluctuations based on synthetic turbulence model
- Mean relative humidity profile based on experimental data

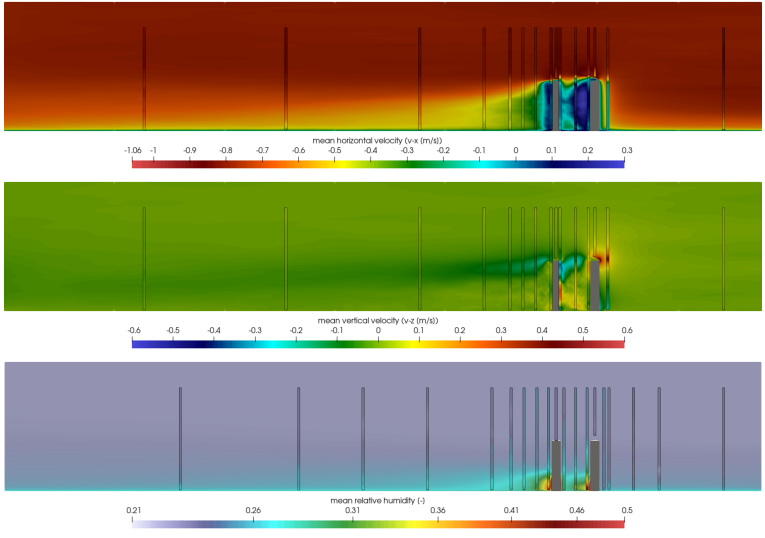
Limestone blocks:

- No-slip condition on all sides except downstream
- Small inflow velocity (approx. 1 mm/s) on the downstream side
- High fixed value of relative humidity is prescribed to model transpiration from the fully-wetted surface of the blocks

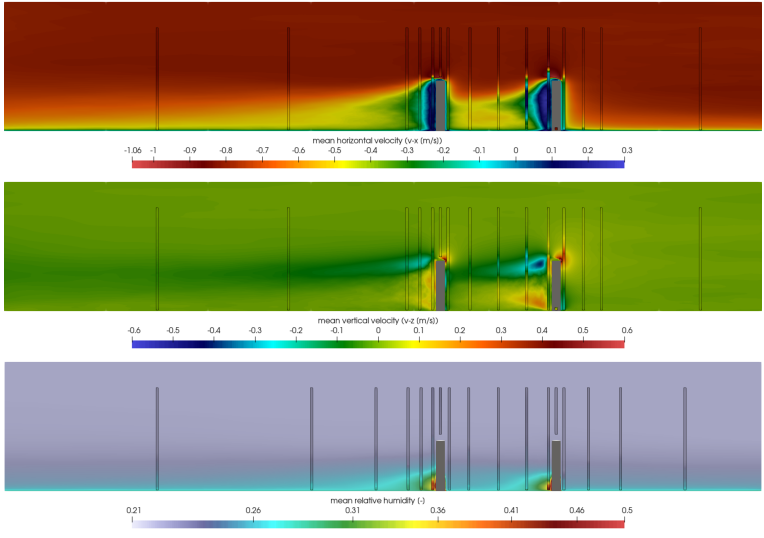
Simulations – velocity and relative humidity profiles



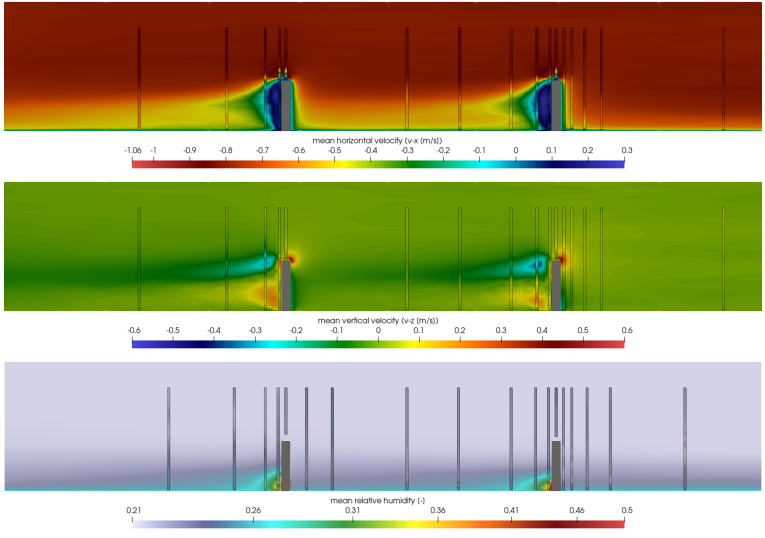
Qualitative comparison with experiment (EX-1: 15 cm)



Qualitative comparison with experiment (EX-2: 45 cm)

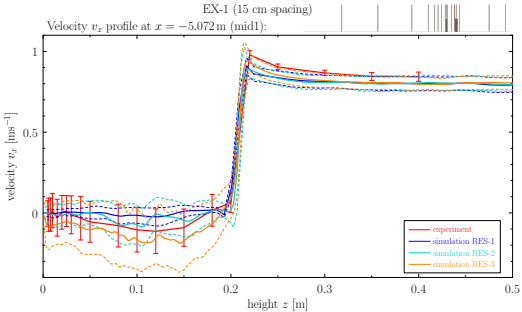


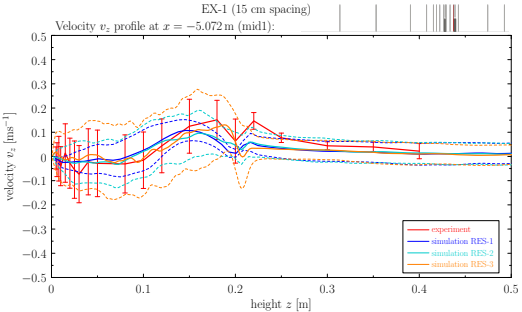
Qualitative comparison with experiment (EX-3: 105 cm)



Quantitative comparison with experiment (EX-1: 15 cm)

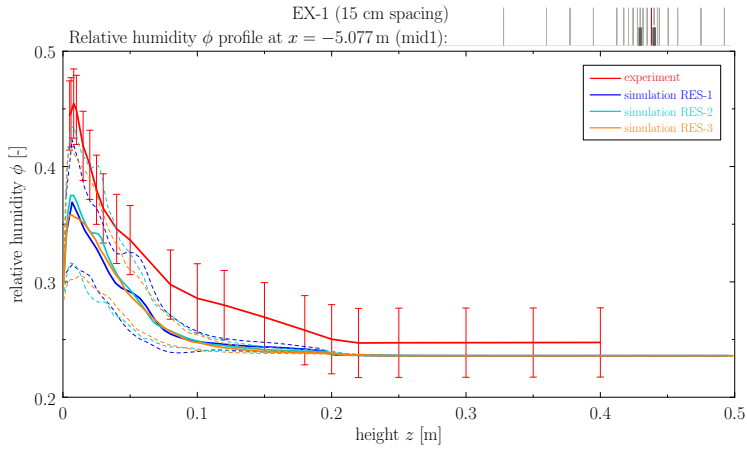
Horizontal and vertical velocity components:





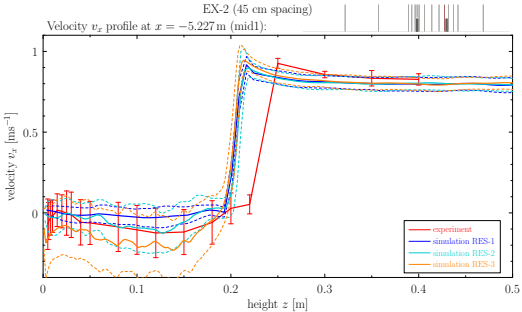
Quantitative comparison with experiment (EX-1: 15 cm)

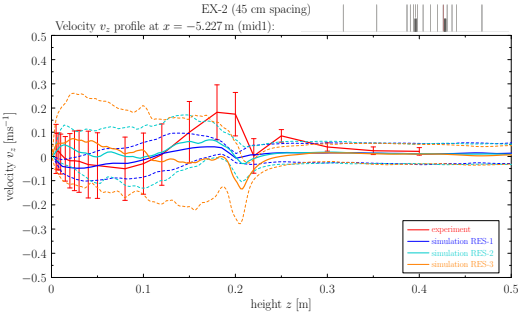
Relative humidity:



Quantitative comparison with experiment (EX-2: 45 cm)

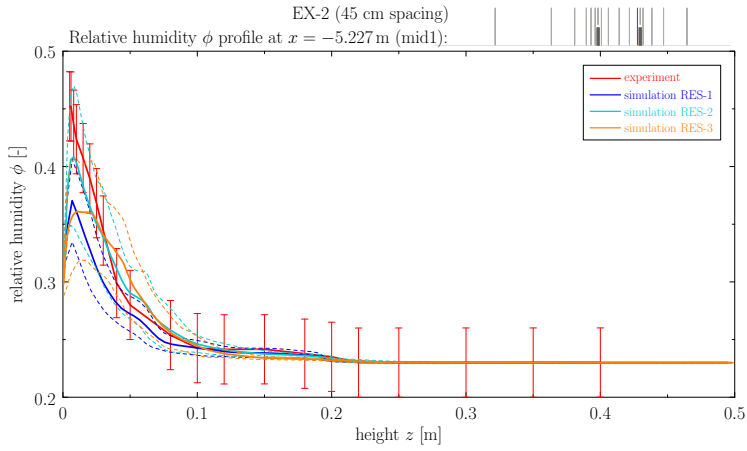
Horizontal and vertical velocity components:





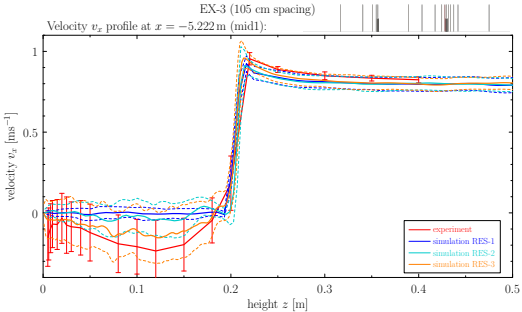
Quantitative comparison with experiment (EX-2: 45 cm)

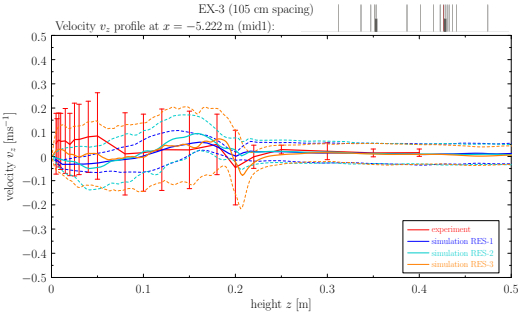
Relative humidity:



Quantitative comparison with experiment (EX-3: 105 cm)

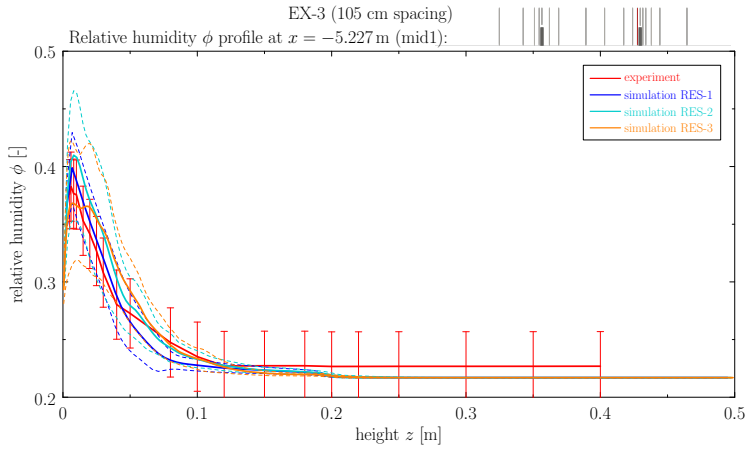
Horizontal and vertical velocity components:





Quantitative comparison with experiment (EX-3: 105 cm)

Relative humidity:



Conclusion

- Template Numerical Library (TNL) provides building blocks for numerical solvers
- TNL-MHFEM: implementation for distributed GPU-accelerated computing, interfaced with the Hypre library for the solution of linear systems
- TNL-LBM: efficient and scalable LBM implementation (NVIDIA CUDA + MPI)
- Coupled LBM-MHFEM computational approach for NSE coupled with a general system of advection–diffusion–reaction PDEs

Future plans and opportunities:

- Coupling different models and methods
- Distributed solver with immersed boundary method
- Adaptivity/non-uniform grid for LBM
- Other LBM models: multiphase, compressible, ...

References

📃 T. Oberhuber, J. Klinkovský, R. Fučík, *TNL: Numerical library for modern parallel architectures*. Acta Polytechnica. 2021, 61(SI), 122-134. ISSN 1805-2363. DOI: 10.14311/AP.2021.61.0122.

📃 J. Klinkovský, T. Oberhuber, R. Fučík, V. Žabka, *Configurable open-source data structure for distributed conforming unstructured homogeneous meshes with GPU support*. ACM Transactions on Mathematical Software. 2022, 48(3), 1-33. ISSN 0098-3500. DOI: 10.1145/3536164.

📃 R. Fučík, J. Klinkovský, J. Solovský, T. Oberhuber, J. Mikyška, *Multidimensional mixed-hybrid finite element method for compositional two-phase flow in heterogeneous porous media and its parallel implementation on GPU*. Computer Physics Communications. 2019, 238 165-180. DOI: 10.1016/j.cpc.2018.12.004.

📃 J. Klinkovský, A. C. Trautz, R. Fučík, T. H. Illangasekare, *Lattice Boltzmann method–based efficient GPU simulator for vapor transport in the boundary layer over a moist soil: Development and experimental validation*. Computers & Mathematics with Applications. 2023, 138, 65-87. DOI: 10.1016/j.camwa.2023.02.021