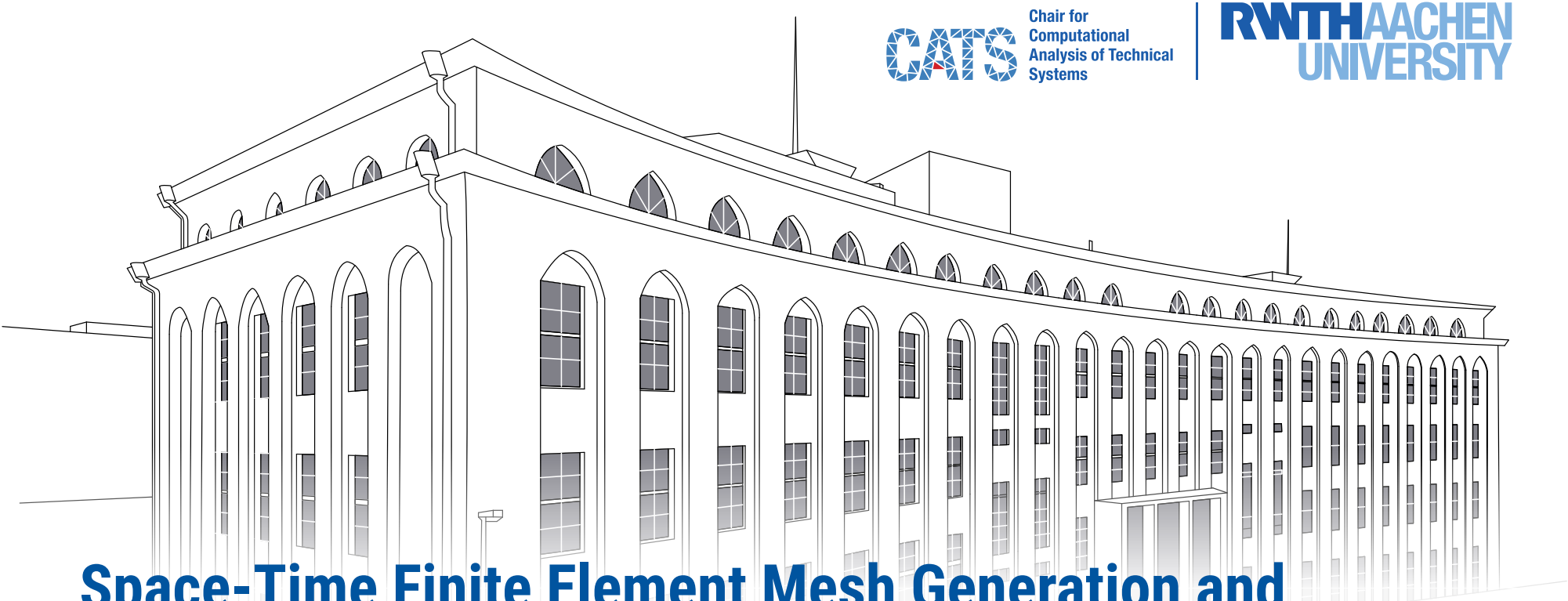


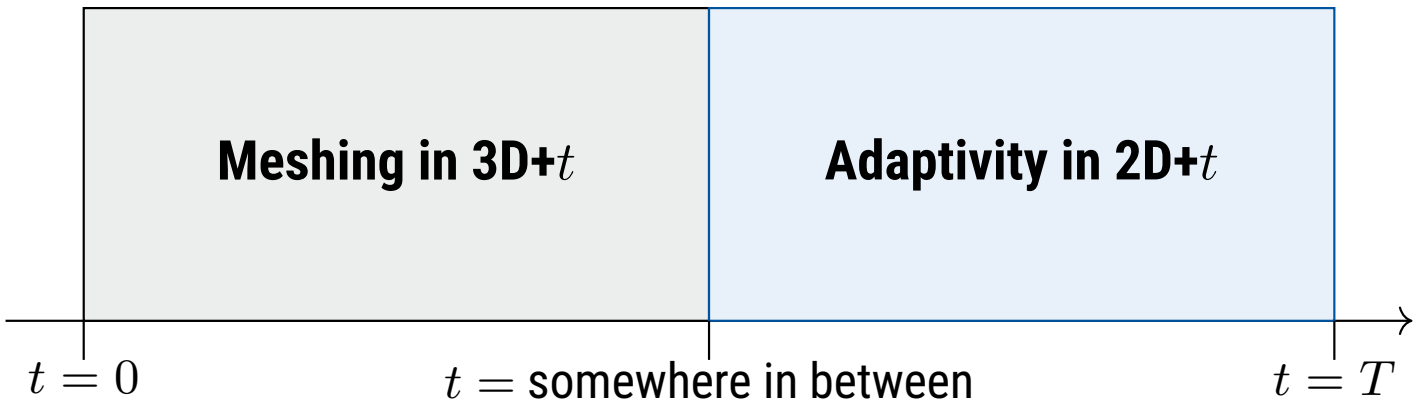
Space-Time Finite Element Mesh Generation and Adaption for Fluid Dynamics

Oliver Wege, Norbert Hosters, Marek Behr

Chair for Computational Analysis of Technical Systems (CATS)
RWTH Aachen University, Aachen, Germany

Workshop on Triangulations and More General Meshes for Space-Time Applications — SoCG 2026, New Brunswick, NJ, USA — June 5, 2026

Agenda


Meshing in 3D+t
Adaptivity in 2D+t
$t = 0$
$t =$ somewhere in between
$t = T$

Space-time finite elements

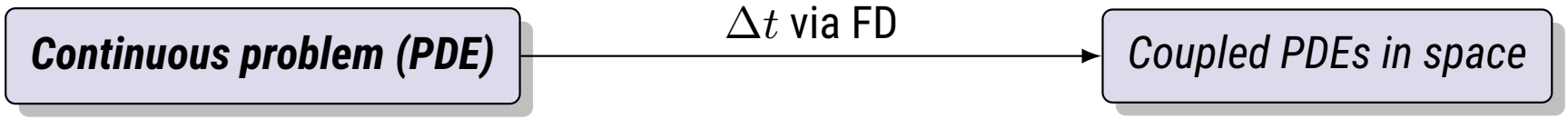
Discretization of transient continuum problems

Continuous problem (PDE)

Discrete problem

Space-time finite elements

Discretization of transient continuum problems

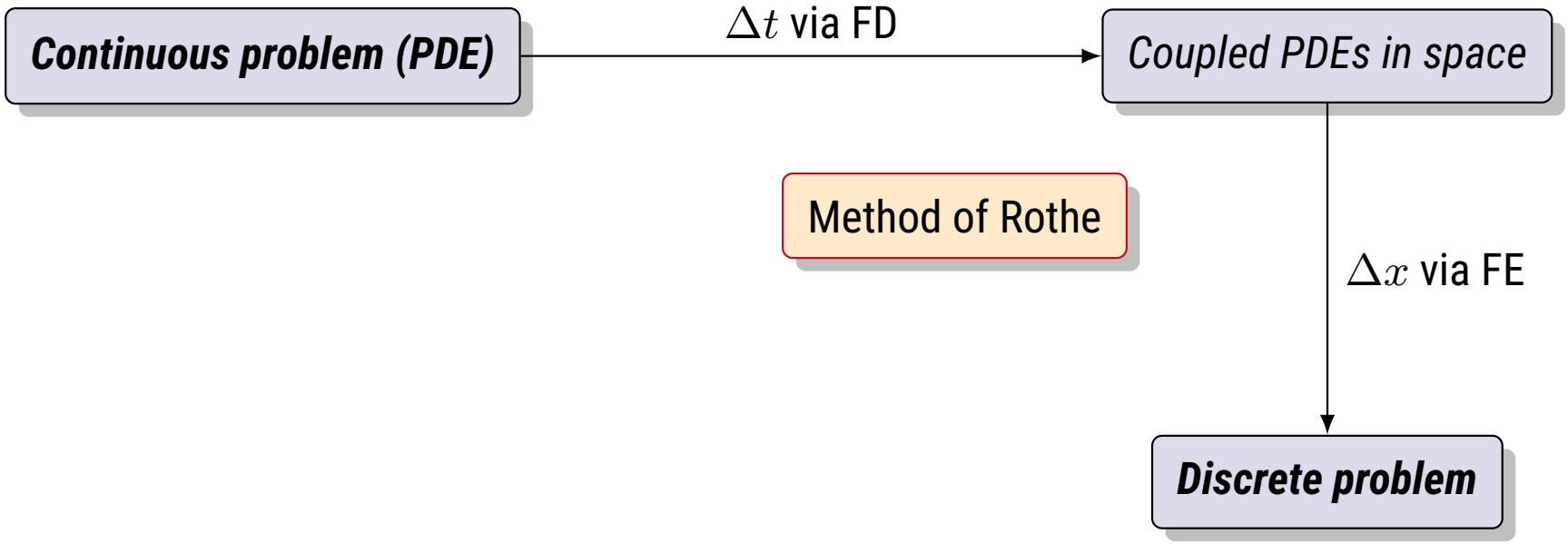


Discrete problem

FD = Finite Differences

Space-time finite elements

Discretization of transient continuum problems

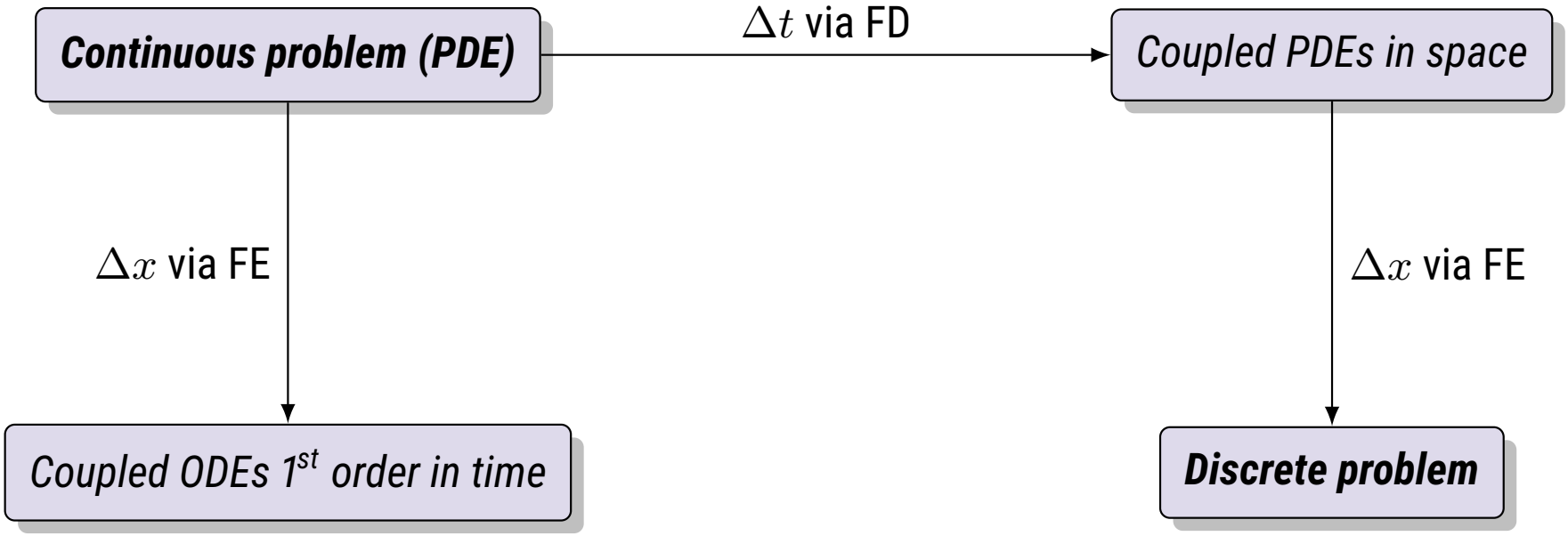


FD = Finite Differences

FE = Finite Elements

Space-time finite elements

Discretization of transient continuum problems

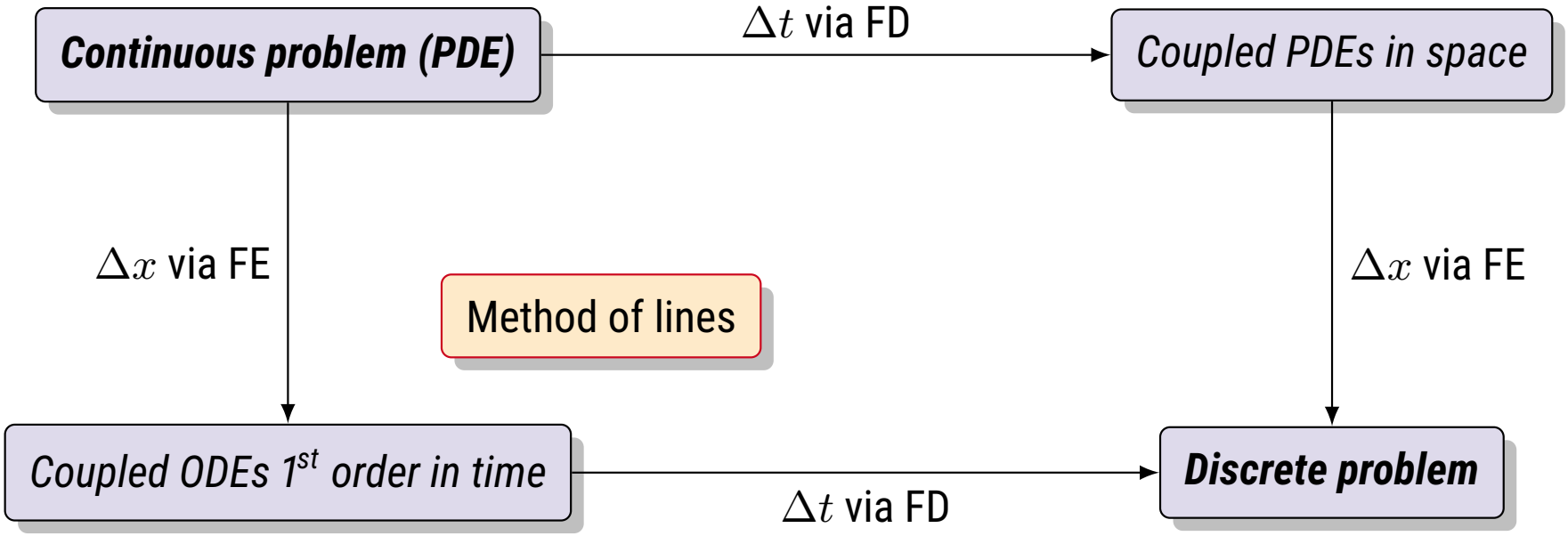


FD = Finite Differences

FE = Finite Elements

Space-time finite elements

Discretization of transient continuum problems

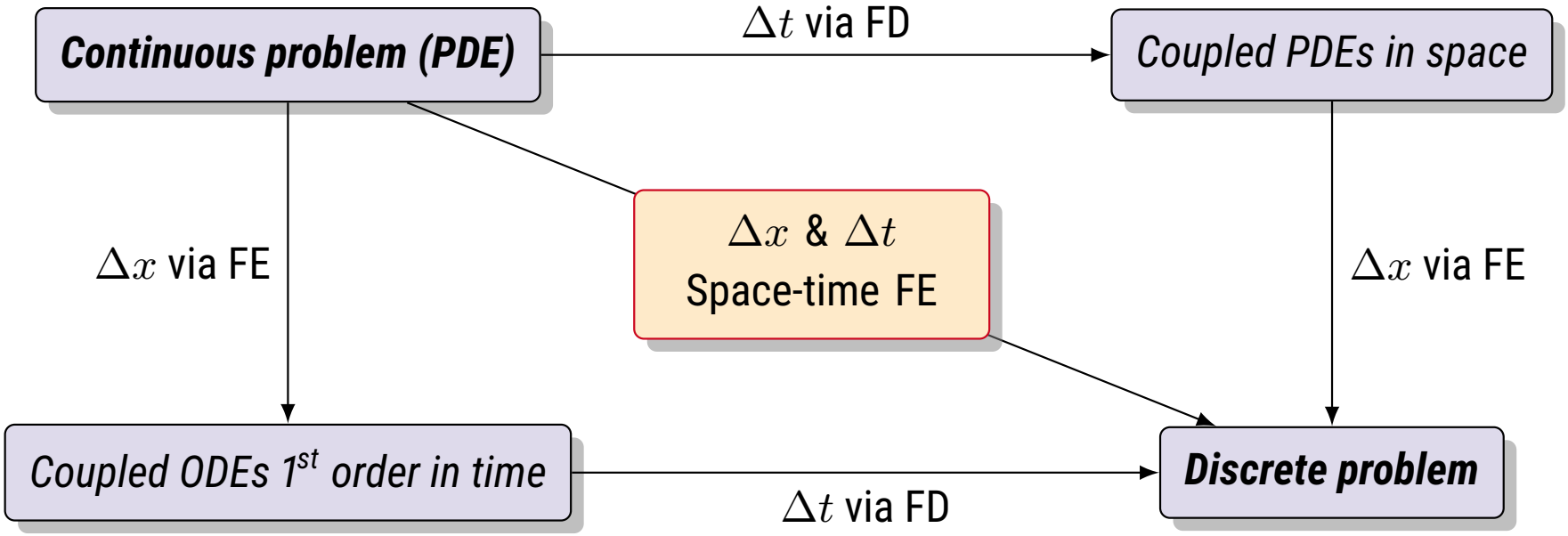


FD = Finite Differences

FE = Finite Elements

Space-time finite elements

Discretization of transient continuum problems



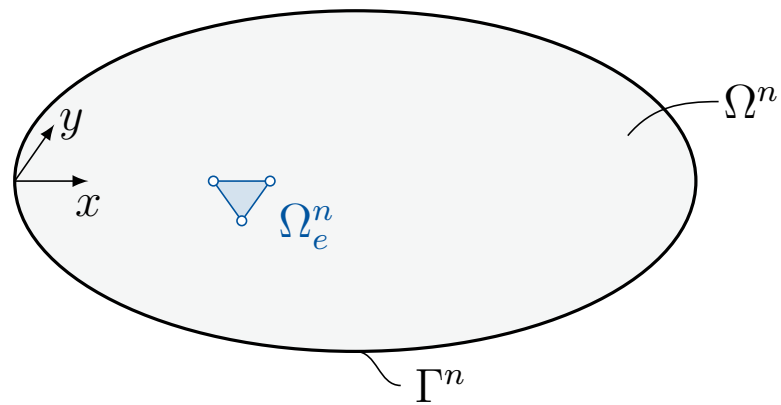
FD = Finite Differences

FE = Finite Elements

Space-time finite elements

Extending the finite element function space to time

- Spatial domain Ω, $\Gamma = \partial\Omega$



Space-time finite elements

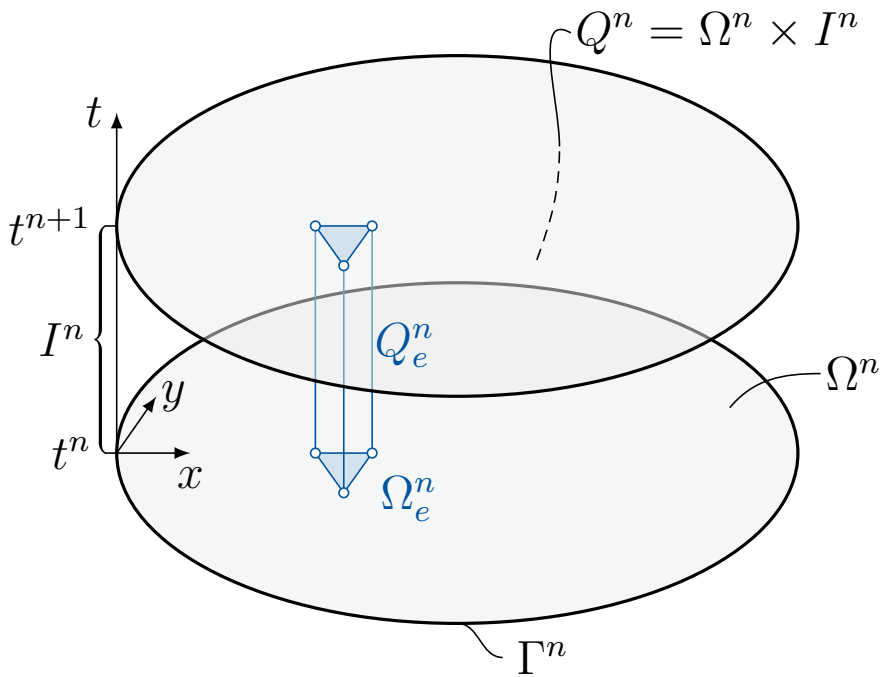
Extending the finite element function space to time

- Spatial domain Ω, $\Gamma = \partial\Omega$
- Slab thickness $\equiv$ time interval

$$I^n \in \left[t^n, t^{n+1}\right]$$

- Integration over space and time

$$\int_{Q^n} \bullet \, dQ = \int_I \int_\Omega \bullet \, d\Omega dt$$



Space-time finite elements

Extending the finite element function space to time

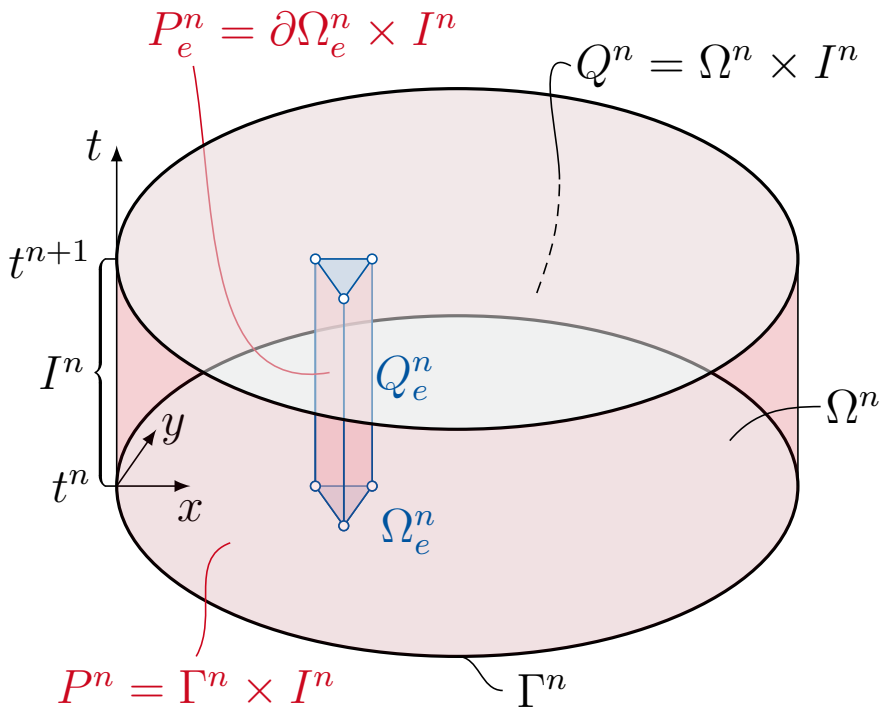
- Spatial domain Ω, $\Gamma = \partial\Omega$
- Slab thickness $\equiv$ time interval

$$I^n \in \left[t^n, t^{n+1}\right]$$

- Integration over space and time

$$\int_{Q^n} \bullet \, dQ = \int_I \int_\Omega \bullet \, d\Omega dt$$

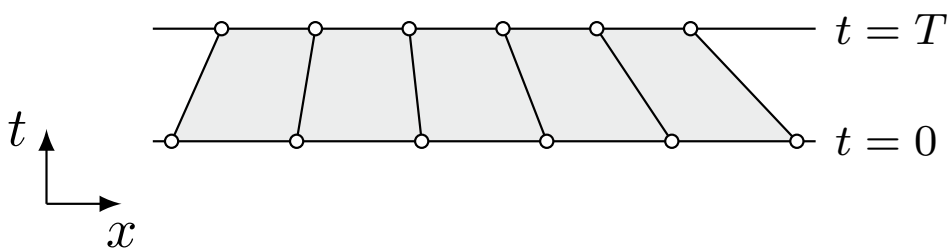
$$\int_{P^n} \bullet \, dP = \int_I \int_\Gamma \bullet \, d\Gamma dt$$



Space-time finite elements

Fundamental flavors of the time meshing

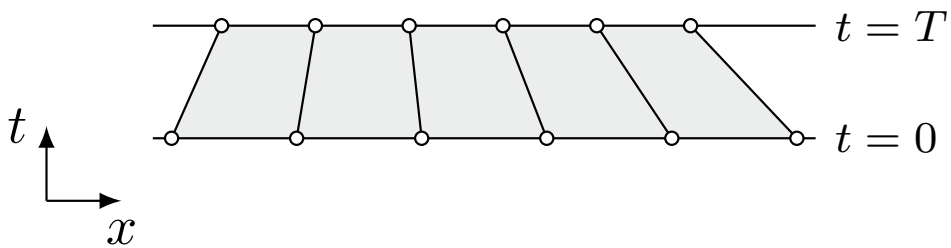
Prismatic Space-Time



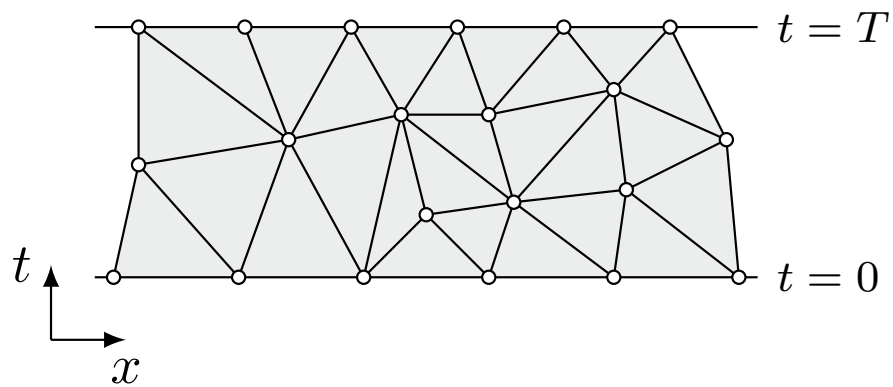
Space-time finite elements

Fundamental flavors of the time meshing

Prismatic Space-Time



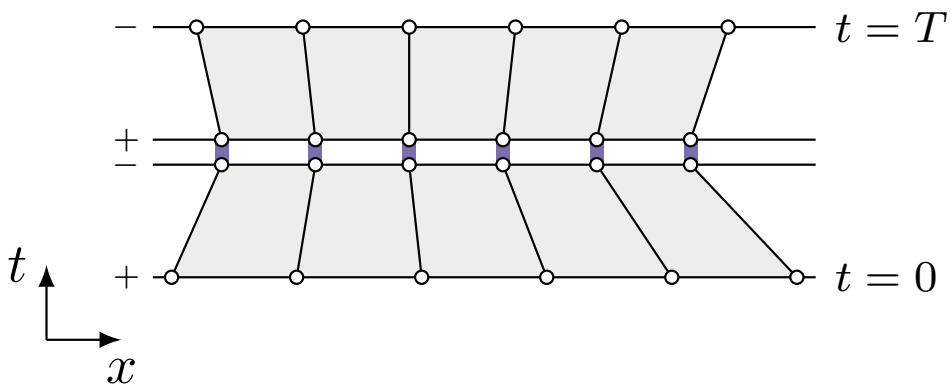
Simplex Space-Time



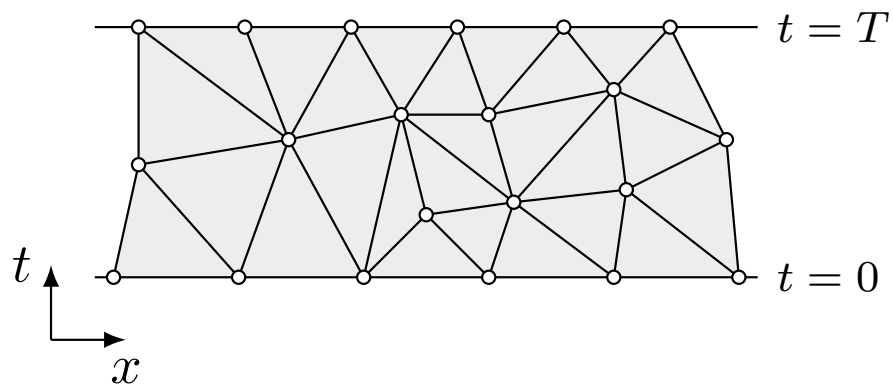
Space-time finite elements

Fundamental flavors of the time meshing

Prismatic Space-Time



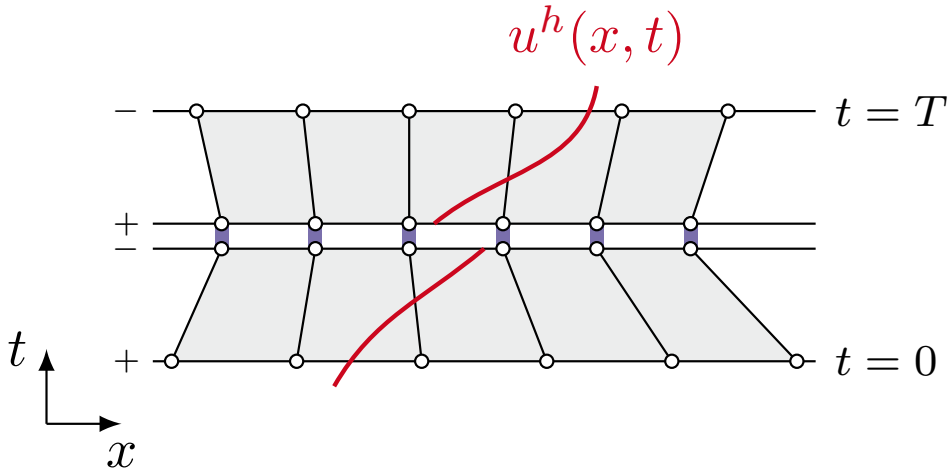
Simplex Space-Time



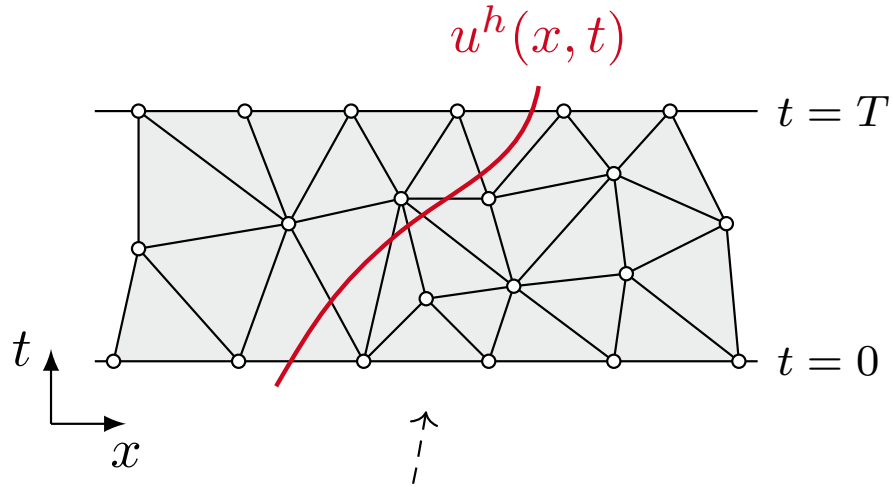
Space-time finite elements

Fundamental flavors of the time meshing

(Discontinuous)
Prismatic Space-Time



(Continuous)
Simplex Space-Time



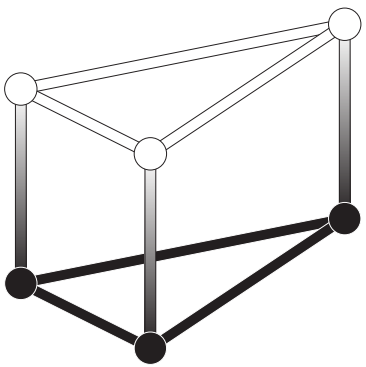
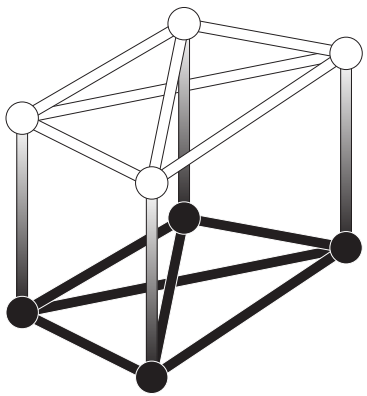
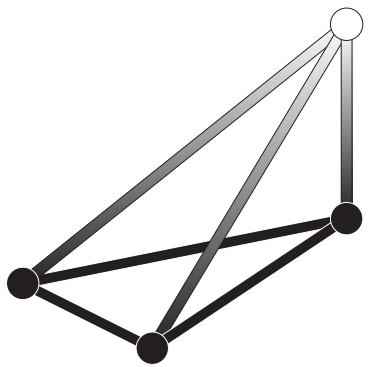
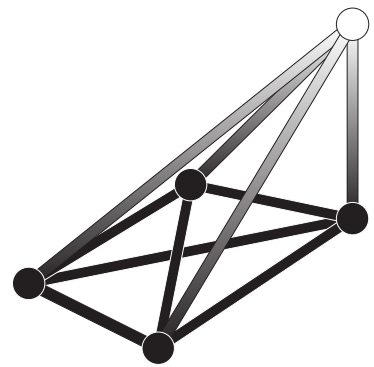
discontinuous slabs possible here too

Space-time finite elements

Prismatic vs. simplex space-time finite elements

○ white node at t^{n+1}

● black node at t^n

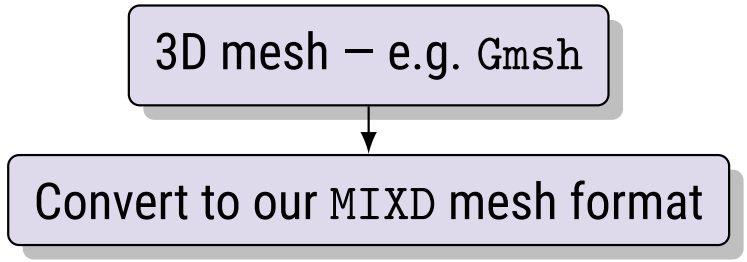
Prismatic		Simplex	
3D	4D	3D	4D
			
▪ *Structured* mesh extrusion in time ▪ Global slab thickness Δt		▪ *Unstructured* mesh extrusion in time ▪ Enables **local time refinement**	

We assume linear equal-order Lagrange finite elements.

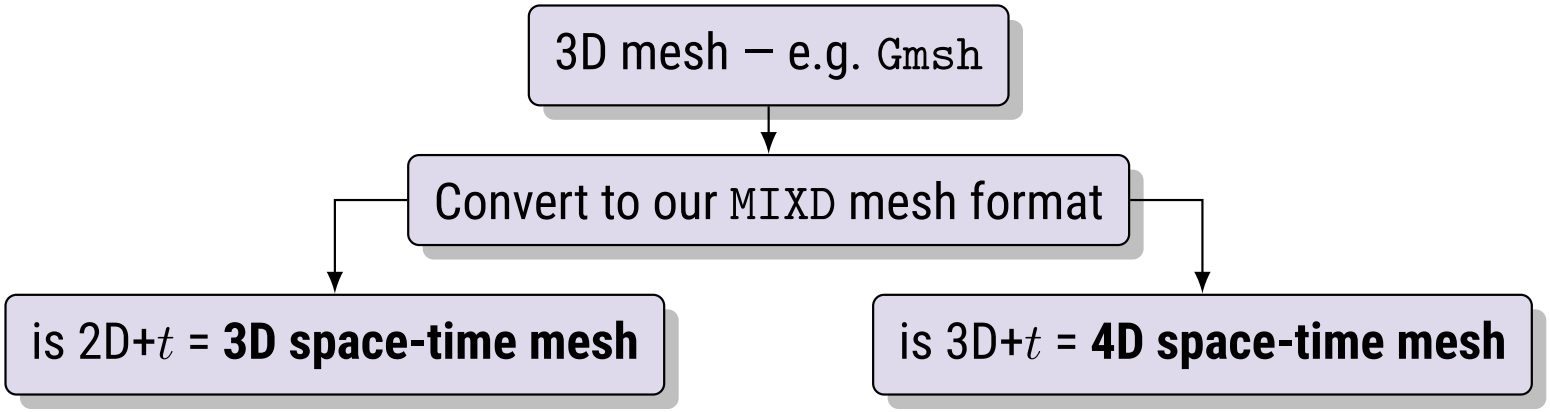
Overview of our mesh generation pipeline

3D mesh — e.g. `Gmsh`

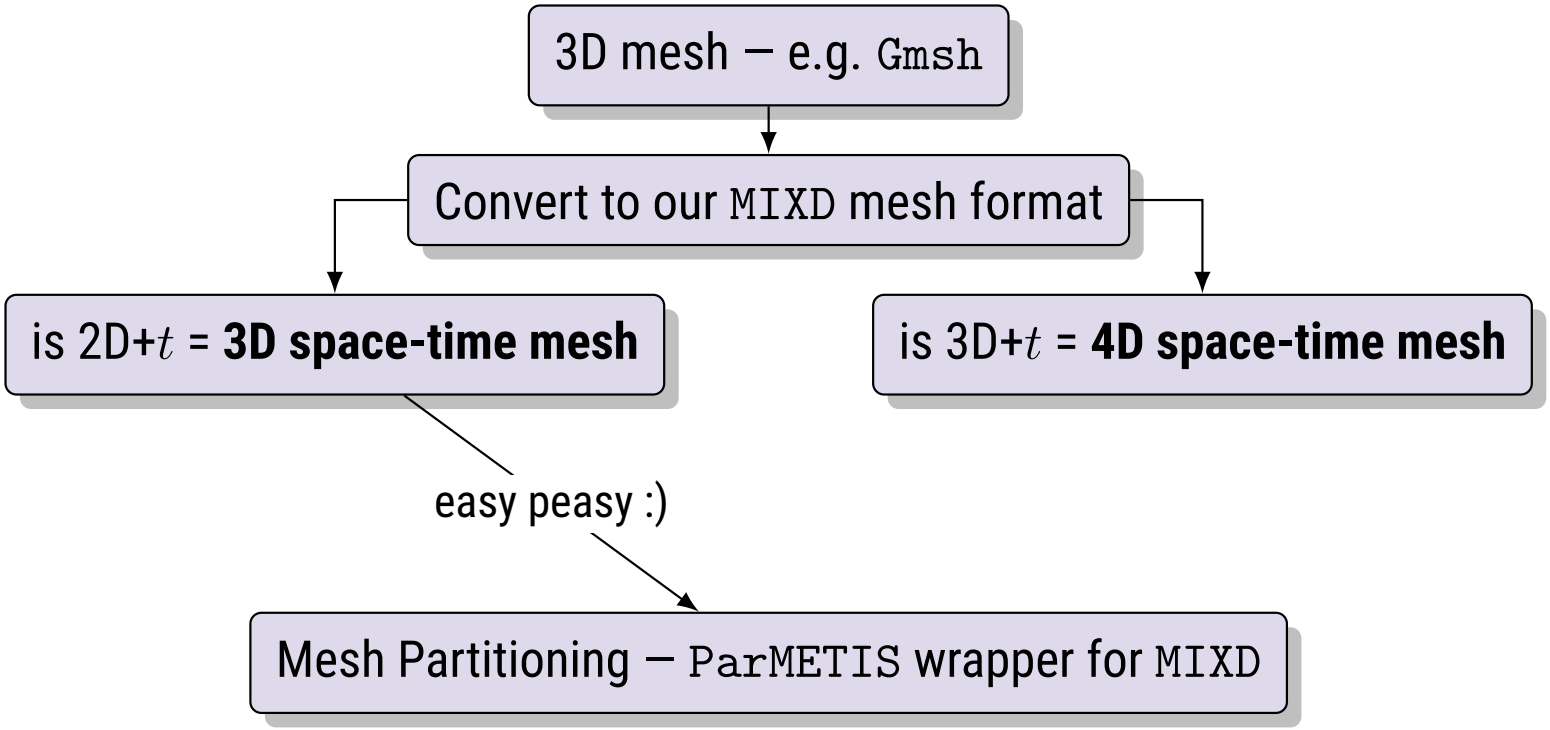
Overview of our mesh generation pipeline



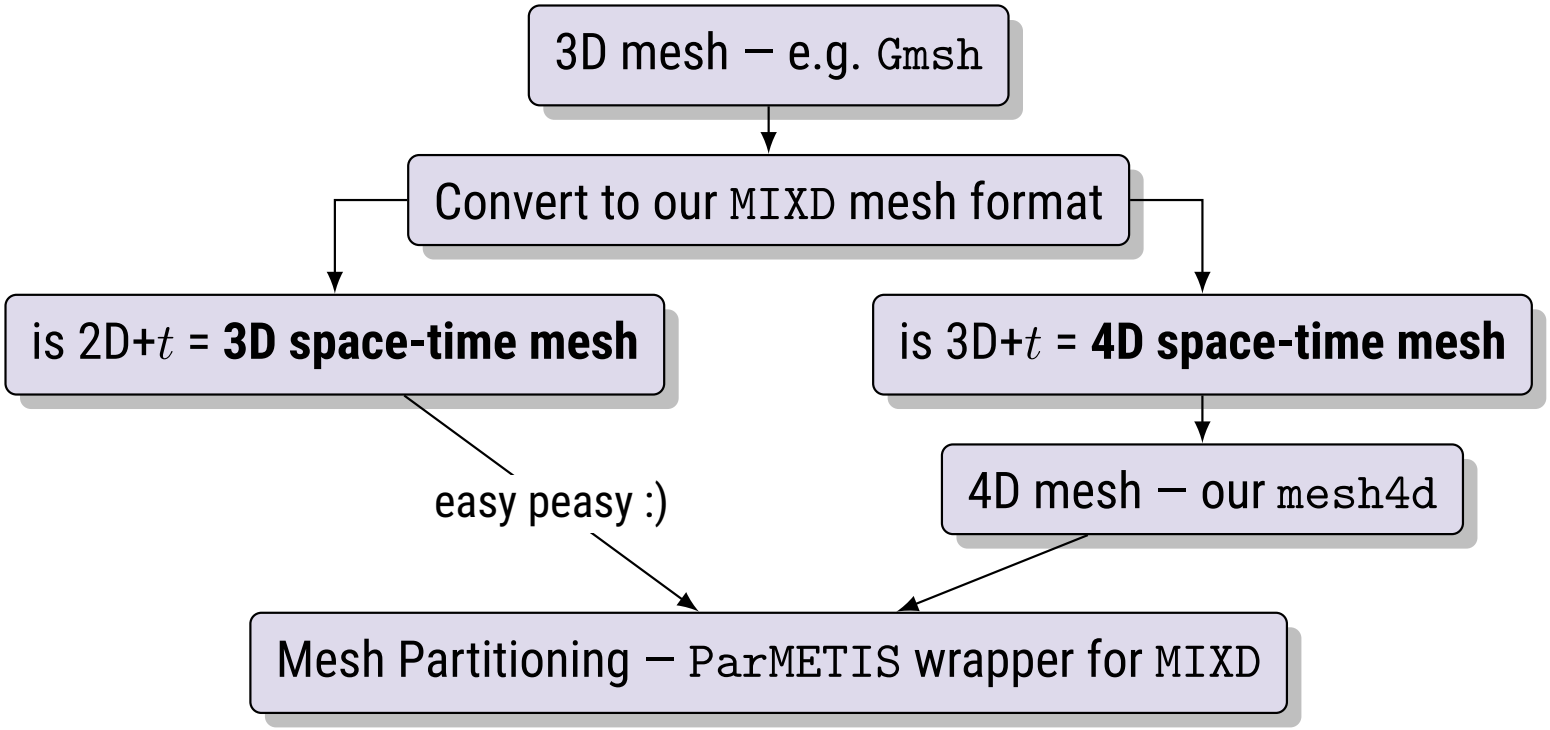
Overview of our mesh generation pipeline



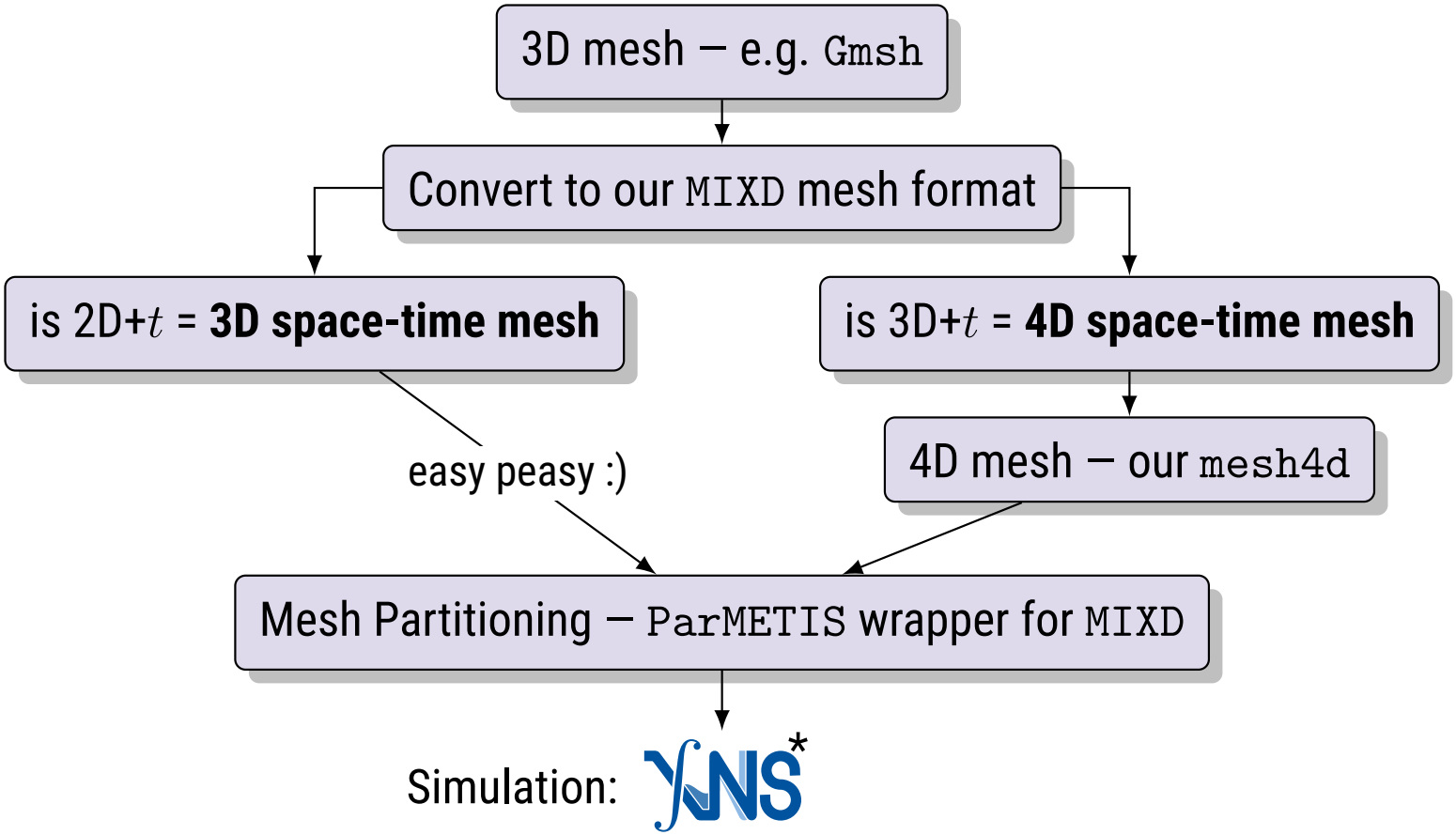
Overview of our mesh generation pipeline



Overview of our mesh generation pipeline

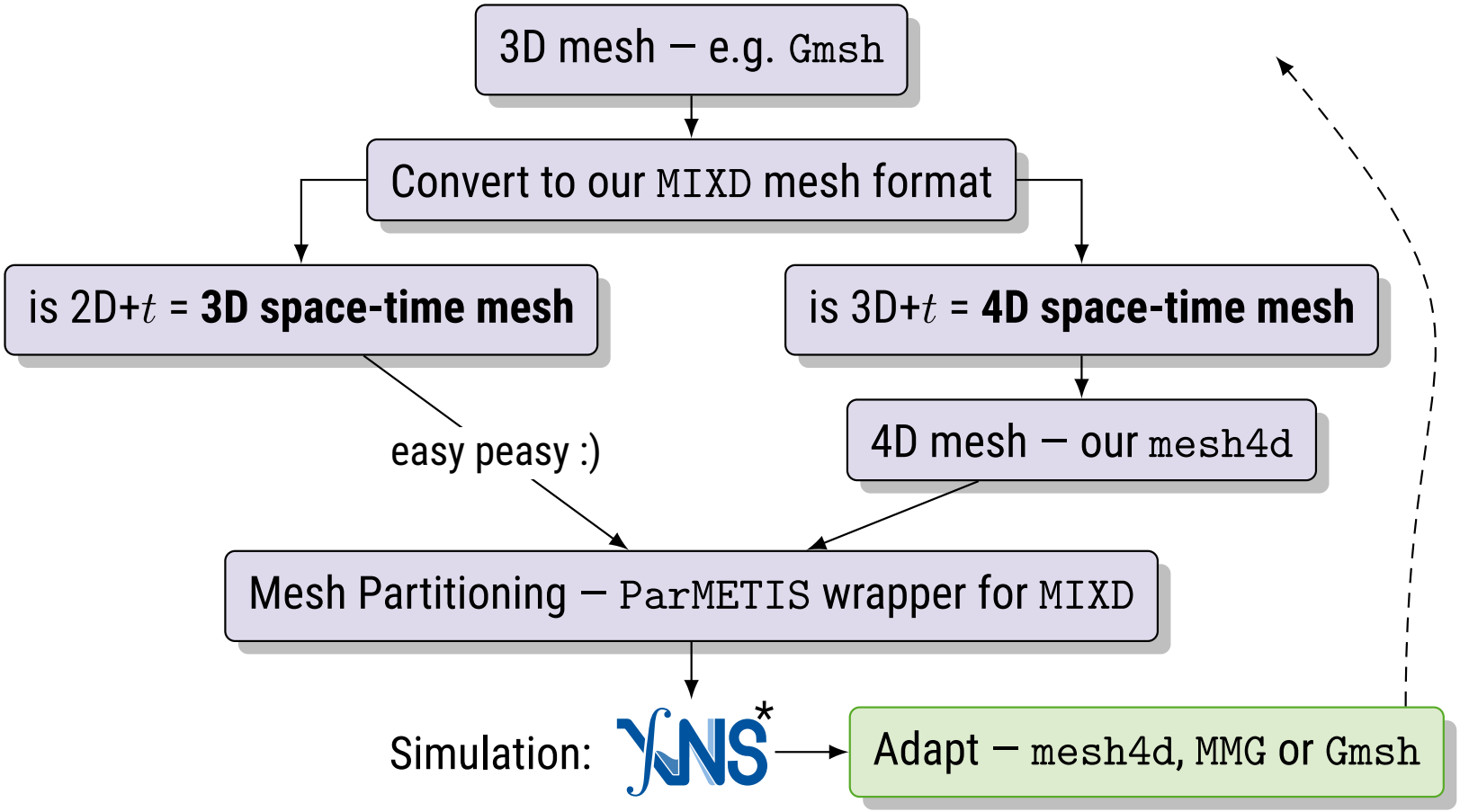


Overview of our mesh generation pipeline



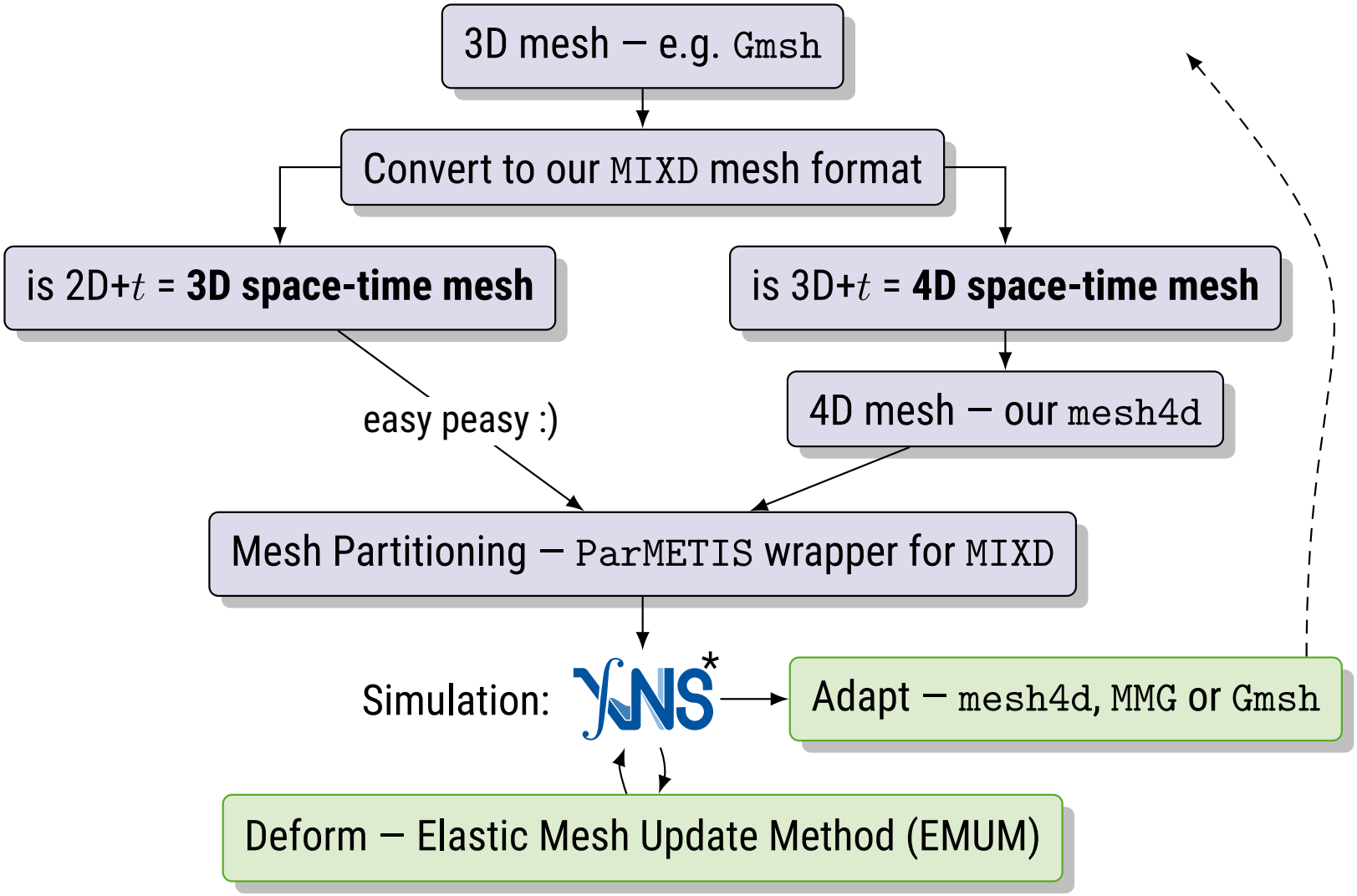
*our in-house finite element solver (open-source soon)

Overview of our mesh generation pipeline



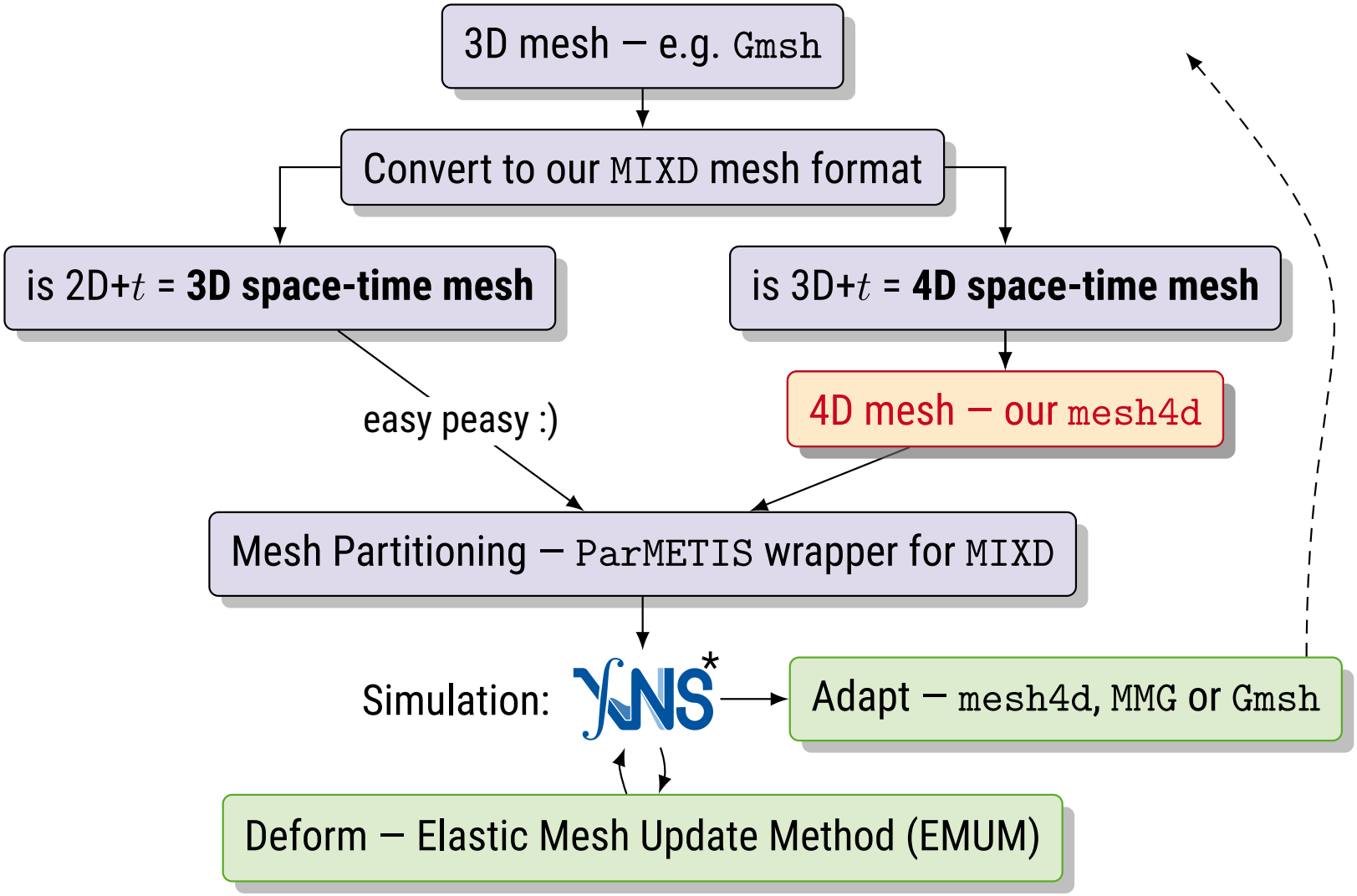
*our in-house finite element solver (open-source soon)

Overview of our mesh generation pipeline



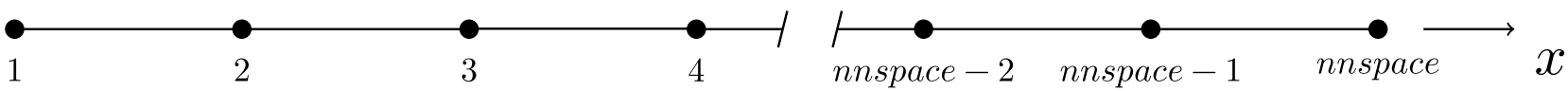
*our in-house finite element solver (open-source soon)

Overview of our mesh generation pipeline

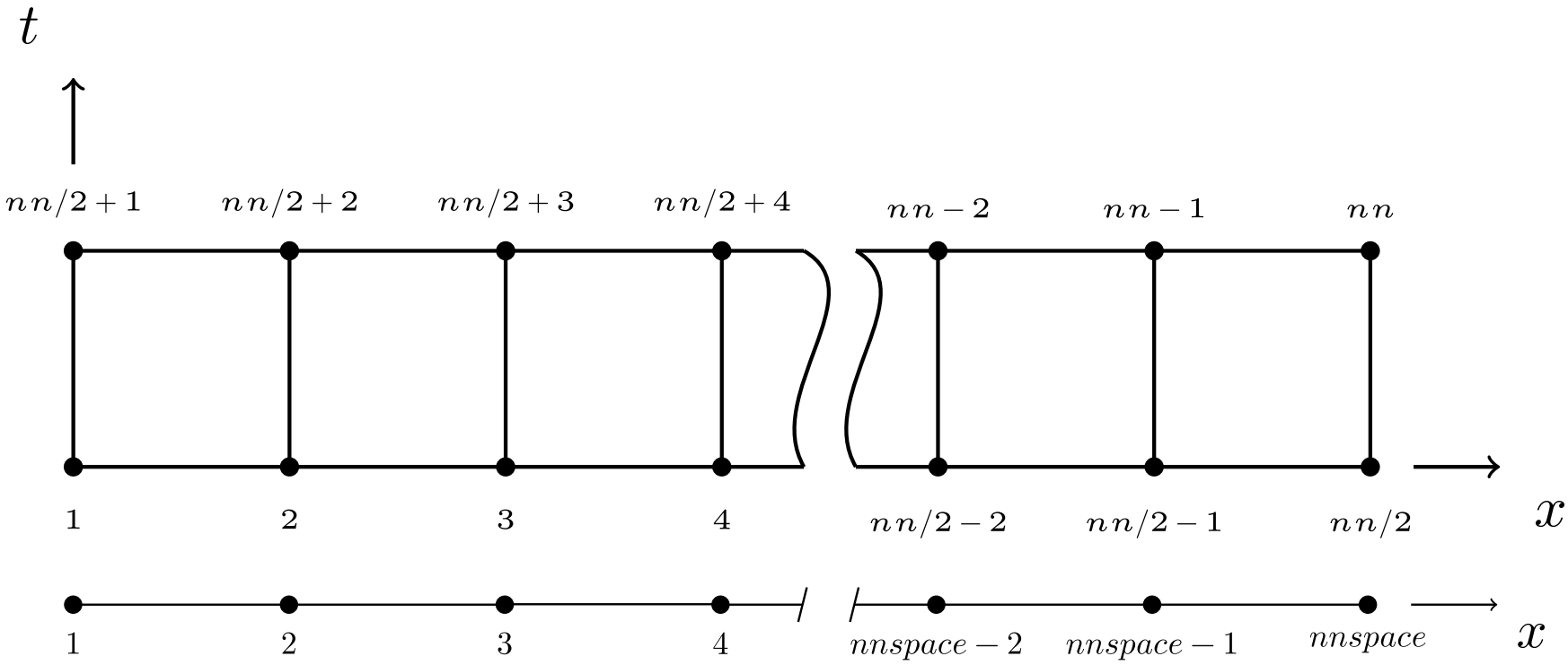


*our in-house finite element solver (open-source soon)

Prismatic space-time mesh generation



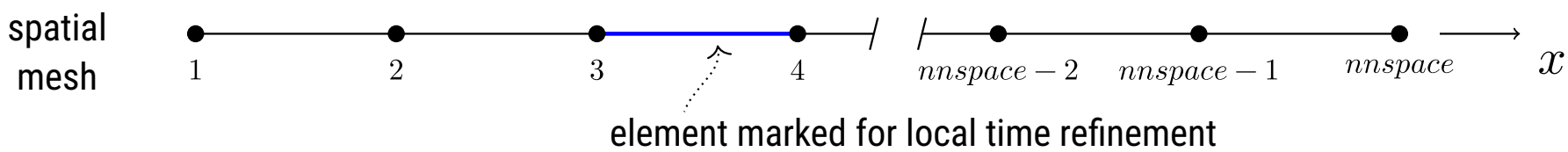
Prismatic space-time mesh generation



- Uniform extrusion of the spatial mesh into an additional time dimension
- Easy to implement and simple to handle
- No local time refinement possible

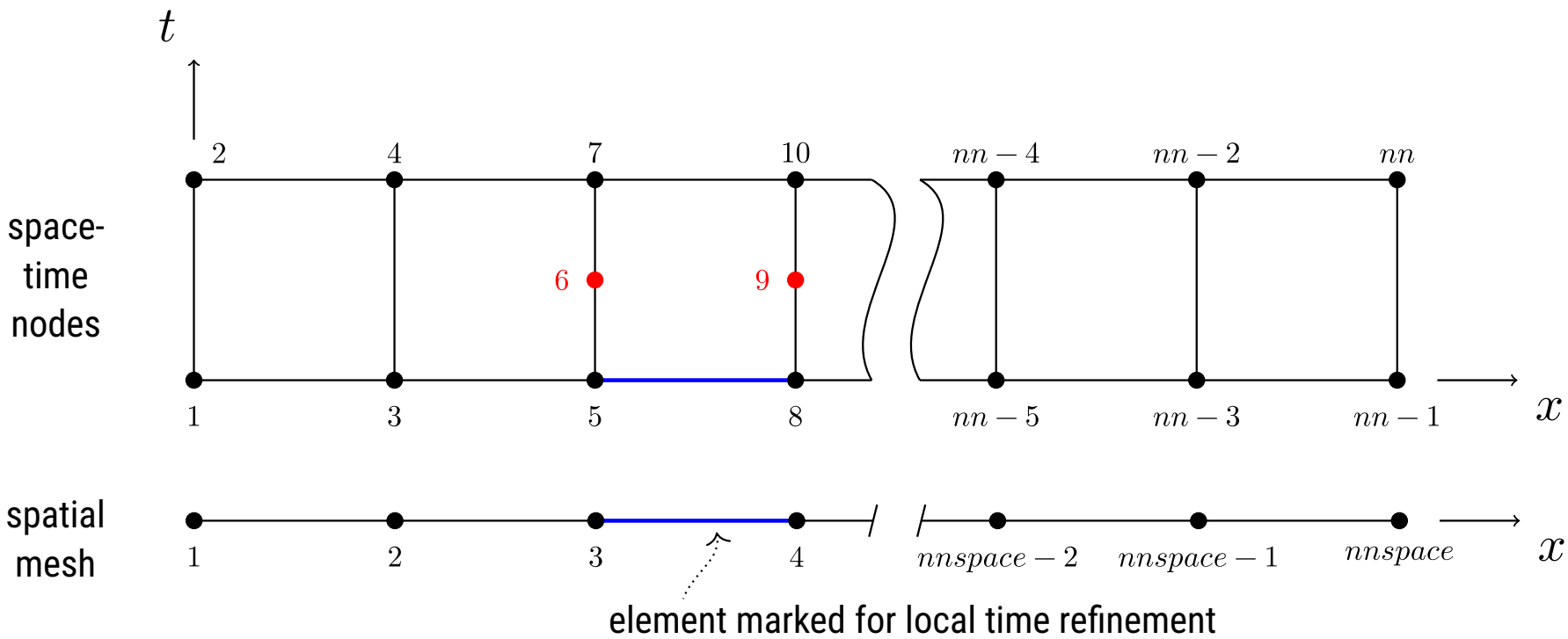
`mesh4d` – Semi-unstructured 4D simplex meshes

Enabling local time refinement



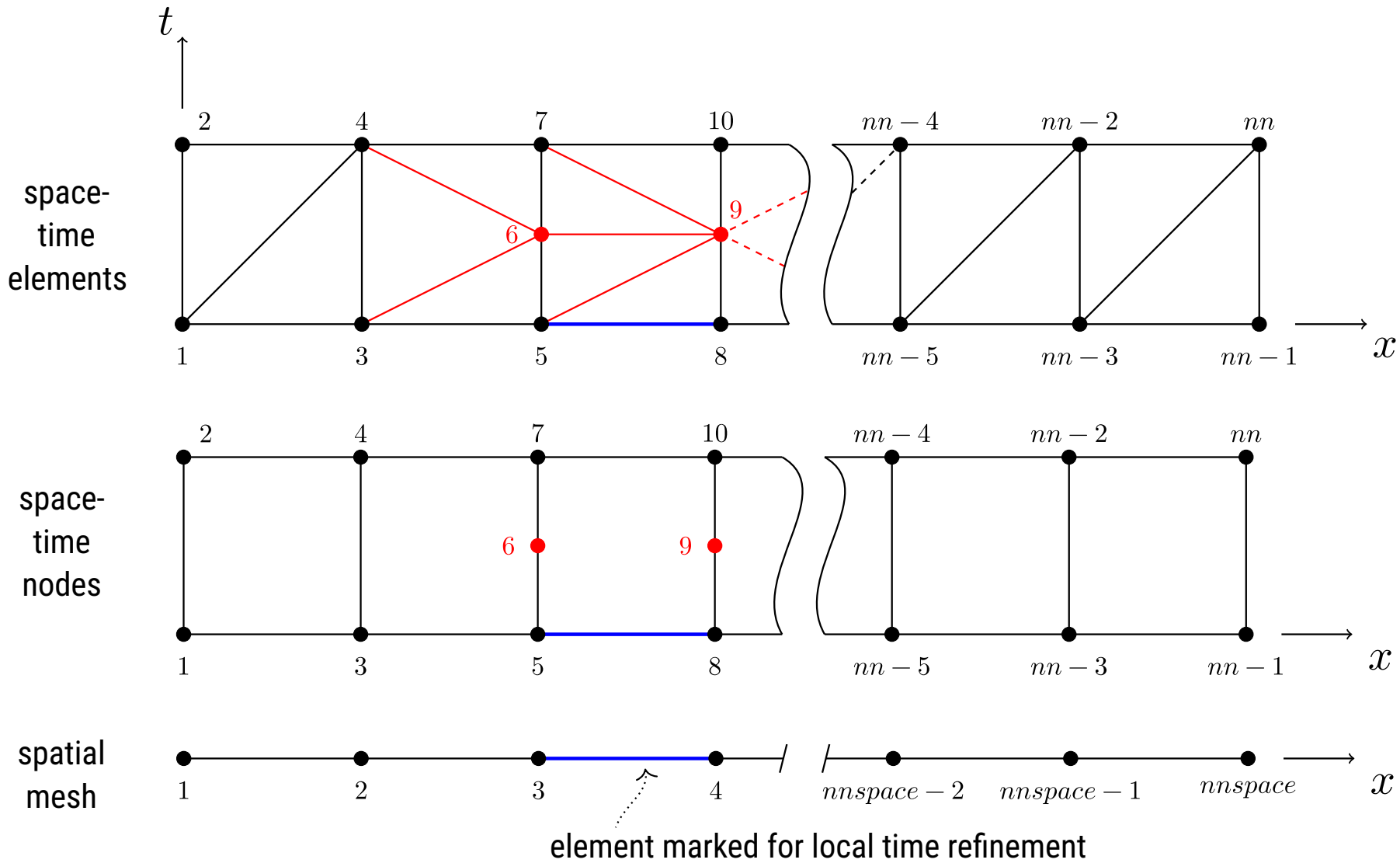
`mesh4d` – Semi-unstructured 4D simplex meshes

Enabling local time refinement



`mesh4d` – Semi-unstructured 4D simplex meshes

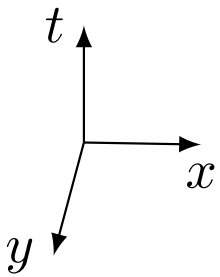
Enabling local time refinement

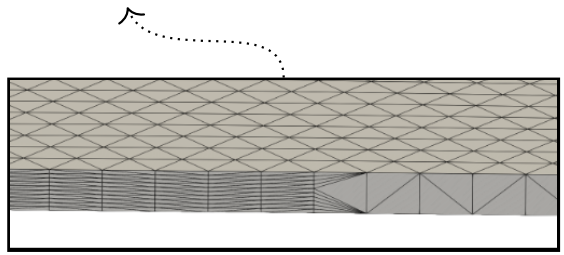


`mesh4d` – Semi-unstructured 4D simplex meshes

Examples from application: Two-phase flow

Injection of a molten polymer in an air-filled cavity





- **Discontinuous** simplex space-time mesh
- Level-set-based time refinement around the air-polymer interface
- User-defined width before and after the interface

`stmesh` – Towards fully unstructured 4D simplex meshes

Based on [Foteinos and Chrisochoides, 2015]

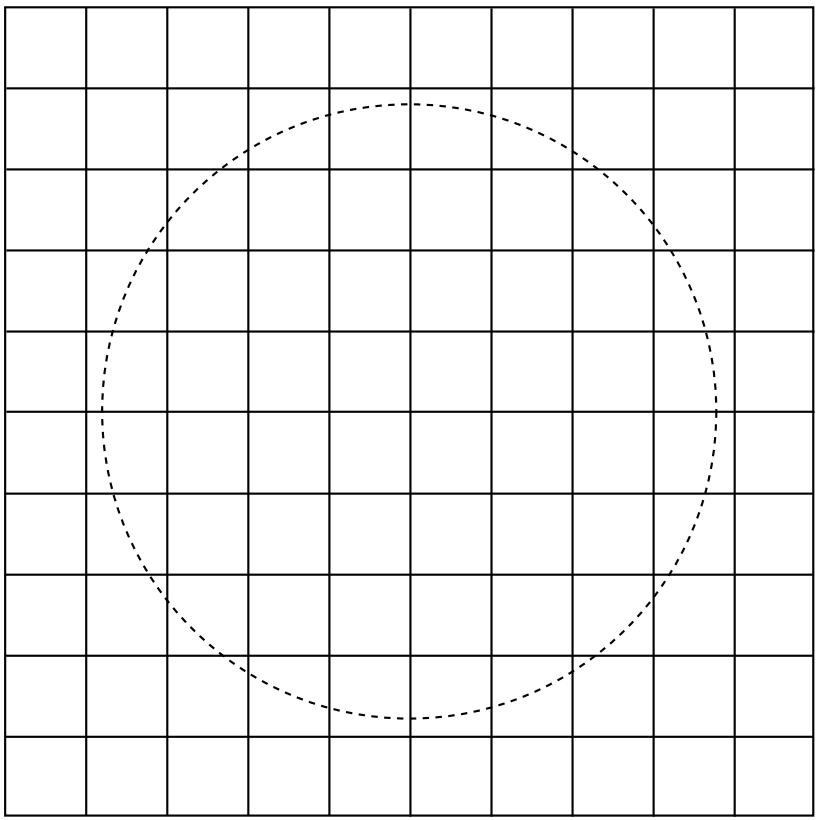
Input is a **segmented 4D image** $\mathcal{I} \subset \mathbb{R}^4$

Implicit surface representation

Level-set $f : \mathbb{R}^4 \to \mathbb{R}$ defines the object:

$$\partial Q = \big\{(\boldsymbol{x}, t) \,|\, f(\boldsymbol{x}, t) = 0\big\},$$
$$Q = \big\{(\boldsymbol{x}, t) \,|\, f(\boldsymbol{x}, t) < 0\big\}.$$

- $f = 0$ from voxel sign interpolation



`stmesh` – Towards fully unstructured 4D simplex meshes

Based on [Foteinos and Chrisochoides, 2015]

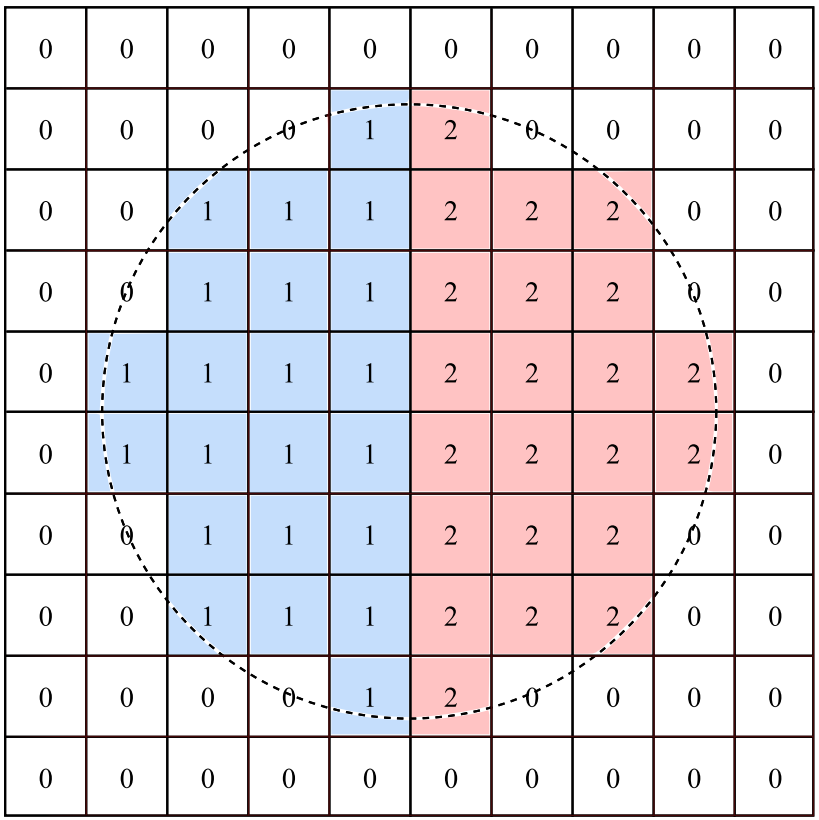
Input is a **segmented 4D image** $\mathcal{I} \subset \mathbb{R}^4$

Implicit surface representation

Level-set $f : \mathbb{R}^4 \to \mathbb{R}$ defines the object:

$$\partial Q = \big\{(\boldsymbol{x}, t) \,|\, f(\boldsymbol{x}, t) = 0\big\},$$
$$Q = \big\{(\boldsymbol{x}, t) \,|\, f(\boldsymbol{x}, t) < 0\big\}.$$

- $f = 0$ from voxel sign interpolation
- Boundary conditions in image (1 & 2)



`stmesh` – Towards fully unstructured 4D simplex meshes

Based on [Foteinos and Chrisochoides, 2015]

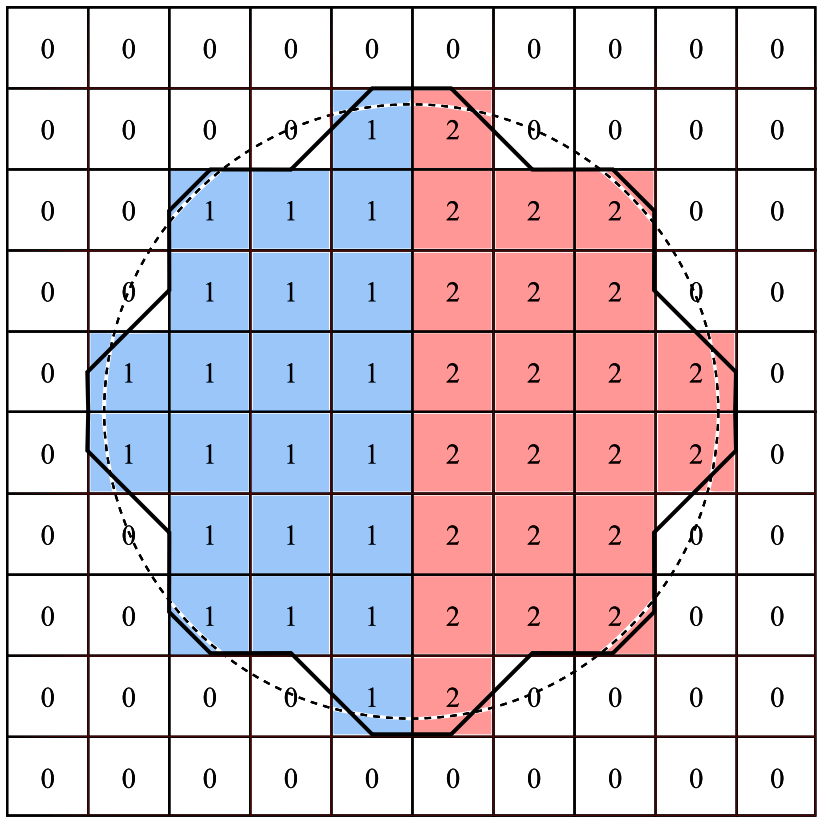
Input is a **segmented 4D image** $\mathcal{I} \subset \mathbb{R}^4$

Implicit surface representation

Level-set $f : \mathbb{R}^4 \to \mathbb{R}$ defines the object:

$$\partial Q = \{(\boldsymbol{x}, t) \mid f(\boldsymbol{x}, t) = 0\},$$
$$Q = \{(\boldsymbol{x}, t) \mid f(\boldsymbol{x}, t) < 0\}.$$

- $f = 0$ from voxel sign interpolation
- Boundary conditions in image (1 & 2)
- Surface extracted via marching cubes
- Triangulation via Delaunay (`CGAL`)



`stmesh` – Towards fully unstructured 4D simplex meshes

The algorithm

1. **Initialise**
   - Insert 16 corners of a hyper-box containing Q
   - Compute the initial Delaunay triangulation

`stmesh` – Towards fully unstructured 4D simplex meshes

The algorithm

1. **Initialise**
   - Insert 16 corners of a hyper-box containing Q
   - Compute the initial Delaunay triangulation
2. **Inspect**
   - Scan every pentatope for one of five *rule violations*:
     R1 surface fidelity, **R2** element size, **R3** aspect ratio and **R4/5** slivers

`stmesh` – Towards fully unstructured 4D simplex meshes

The algorithm

1. **Initialise**
   - Insert 16 corners of a hyper-box containing Q
   - Compute the initial Delaunay triangulation
2. **Inspect**
   - Scan every pentatope for one of five *rule violations*:
     R1 surface fidelity, **R2** element size, **R3** aspect ratio and **R4/5** slivers
3. **Refine**
   - Insert a vertex to fix the highest-priority violation
   - The Delaunay property is maintained at all times

`stmesh` – Towards fully unstructured 4D simplex meshes

The algorithm

1. **Initialise**
   - Insert 16 corners of a hyper-box containing Q
   - Compute the initial Delaunay triangulation
2. **Inspect**
   - Scan every pentatope for one of five *rule violations*:
     R1 surface fidelity, **R2** element size, **R3** aspect ratio and **R4/5** slivers
3. **Refine**
   - Insert a vertex to fix the highest-priority violation
   - The Delaunay property is maintained at all times
4. **Terminate**
   - Stop when no violation remains
   - Output the subset of pentatopes whose circumcenters lie *inside* Q

`stmesh` – Towards fully unstructured 4D simplex meshes

The Five Refinement Rules

Rule	Trigger	Action
R1	Circumball intersects ∂Q and no nearby surface vertex exists yet	Insert new vertex exactly on ∂Q

Priority: R1 $\succ$ R2 $\succ$ R3 $\succ$ R4 $\succ$ R5

`stmesh` – Towards fully unstructured 4D simplex meshes

The Five Refinement Rules

Rule	Trigger	Action
R1	Circumball intersects ∂Q and no nearby surface vertex exists yet	Insert new vertex exactly on ∂Q
R2	Circumball intersects ∂Q but is too large relative to local feature size	Insert circumcenter as new vertex

Priority: R1 $\succ$ R2 $\succ$ R3 $\succ$ R4 $\succ$ R5

`stmesh` – Towards fully unstructured 4D simplex meshes

The Five Refinement Rules

Rule	Trigger	Action
R1	Circumball intersects ∂Q and no nearby surface vertex exists yet	Insert new vertex exactly on ∂Q
R2	Circumball intersects ∂Q but is too large relative to local feature size	Insert circumcenter as new vertex
R3	Circumcenter inside Q and radius-to-shortest-edge ratio too large	Insert circumcenter as new vertex

Priority: R1 $\succ$ R2 $\succ$ R3 $\succ$ R4 $\succ$ R5

`stmesh` – Towards fully unstructured 4D simplex meshes

The Five Refinement Rules

Rule	Trigger	Action
R1	Circumball intersects ∂Q and no nearby surface vertex exists yet	Insert new vertex exactly on ∂Q
R2	Circumball intersects ∂Q but is too large relative to local feature size	Insert circumcenter as new vertex
R3	Circumcenter inside Q and radius-to-shortest-edge ratio too large	Insert circumcenter as new vertex
R4	Circumcenter inside Q and pentatope is a sliver	Insert a random good point inside a *picking region* around it

Priority: R1 $\succ$ R2 $\succ$ R3 $\succ$ R4 $\succ$ R5

`stmesh` – Towards fully unstructured 4D simplex meshes

The Five Refinement Rules

Rule	Trigger	Action
R1	Circumball intersects ∂Q and no nearby surface vertex exists yet	Insert new vertex exactly on ∂Q
R2	Circumball intersects ∂Q but is too large relative to local feature size	Insert circumcenter as new vertex
R3	Circumcenter inside Q and radius-to-shortest-edge ratio too large	Insert circumcenter as new vertex
R4	Circumcenter inside Q and pentatope is a sliver	Insert a random good point inside a *picking region* around it
R5	Surface tetrahedron constrained on ∂Q is a sliver	Insert a random good point inside a *picking region* around it

Priority: R1 $\succ$ R2 $\succ$ R3 $\succ$ R4 $\succ$ R5

`stmesh` – Towards fully unstructured 4D simplex meshes

Isosurface meshing

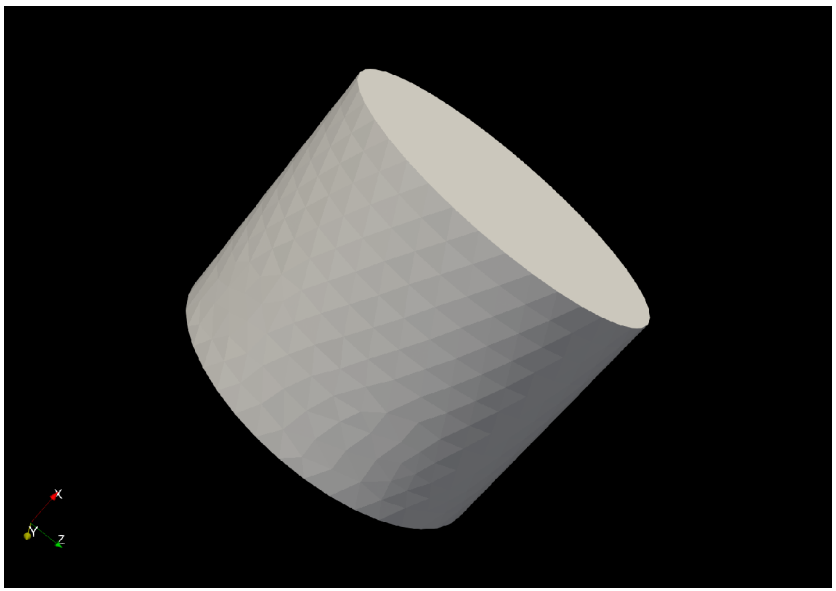
- The boundary ∂Q is **never triangulated separately**
- Surface vertices are discovered and inserted *on-the-fly* during refinement

Why not start from a surface mesh of the boundary?

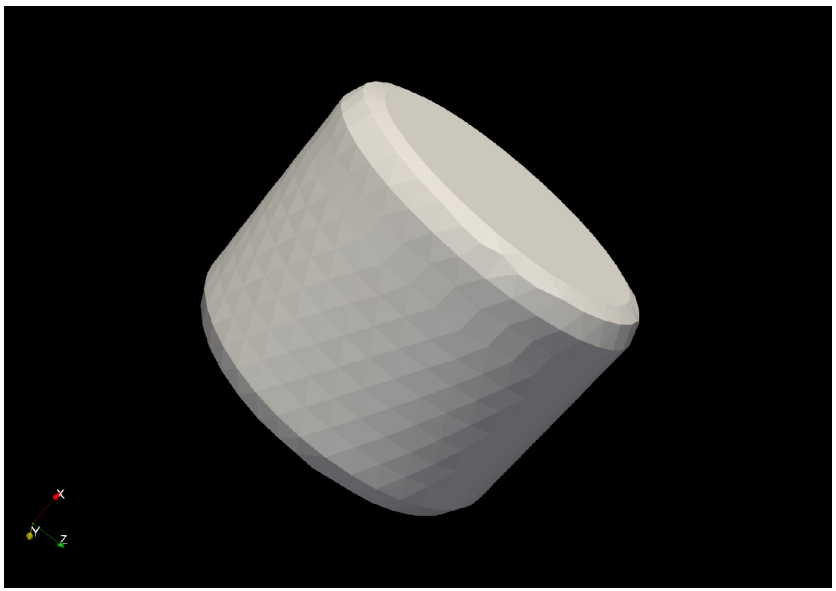
- They inherit small input angles from the segmentation/decimation step
- Small angles degrade quality and can prevent termination
- Isosurface-based methods avoid this entirely

`stmesh` – Towards fully unstructured 4D simplex meshes

Rounded edge issue



Desired geometry

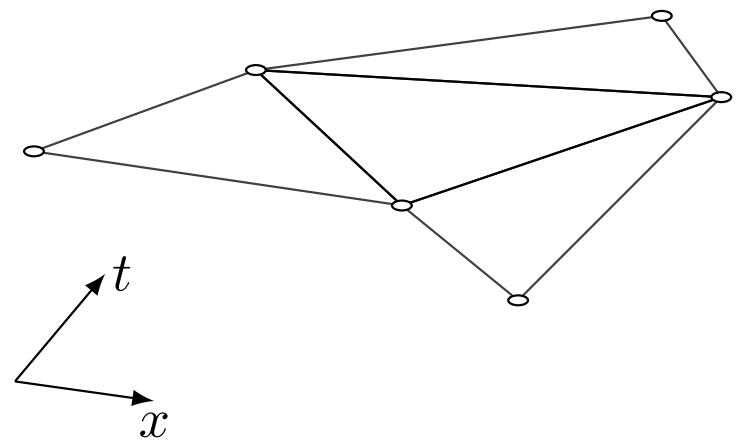


Actual 3D slice in yzt-hyperplane

2D+t adaptivity

A posteriori *Kelly* error estimator [Kelly et al., 1983, De SR Gago et al., 1983]

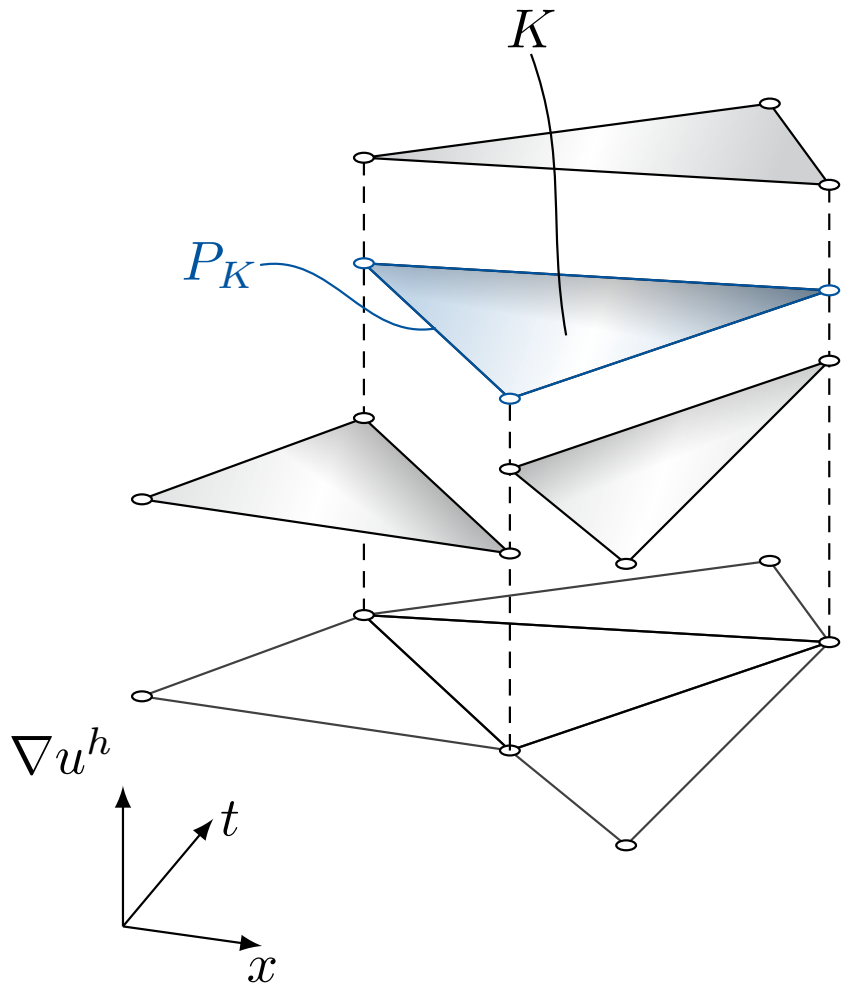
- $\mathcal{C}^0$-continuous finite elements



2D+t adaptivity

A posteriori *Kelly* error estimator [Kelly et al., 1983, De SR Gago et al., 1983]

- $\mathcal{C}^0$-continuous finite elements



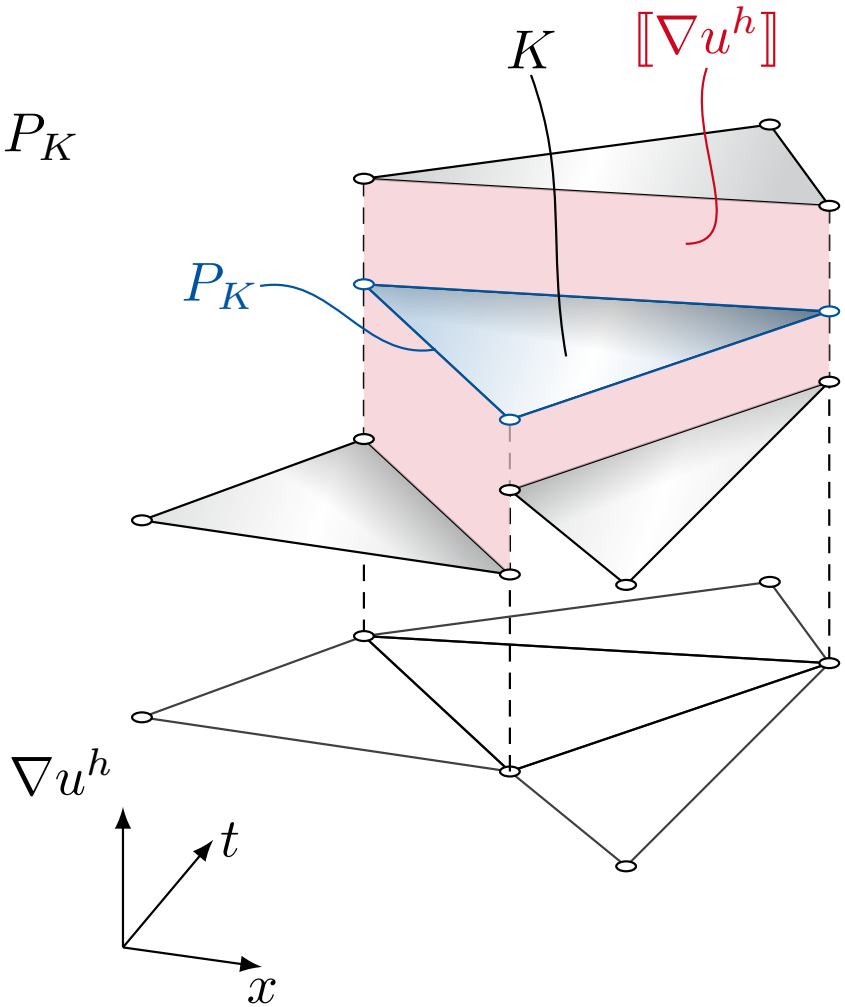
2D+t adaptivity

A posteriori *Kelly* error estimator [Kelly et al., 1983, De SR Gago et al., 1983]

- $\mathcal{C}^0$-continuous finite elements, hence

$$[\![\nabla u^h]\!] = \nabla u^h_- - \nabla u^h_+ \neq 0 \quad \forall (\boldsymbol{x}, t) \in P_K$$

- Space-time element boundary P_K
- Solution gradient jump $[\![\nabla u^h]\!]$



2D+t adaptivity

A posteriori *Kelly* error estimator [Kelly et al., 1983, De SR Gago et al., 1983]

- $\mathcal{C}^0$-continuous finite elements, hence

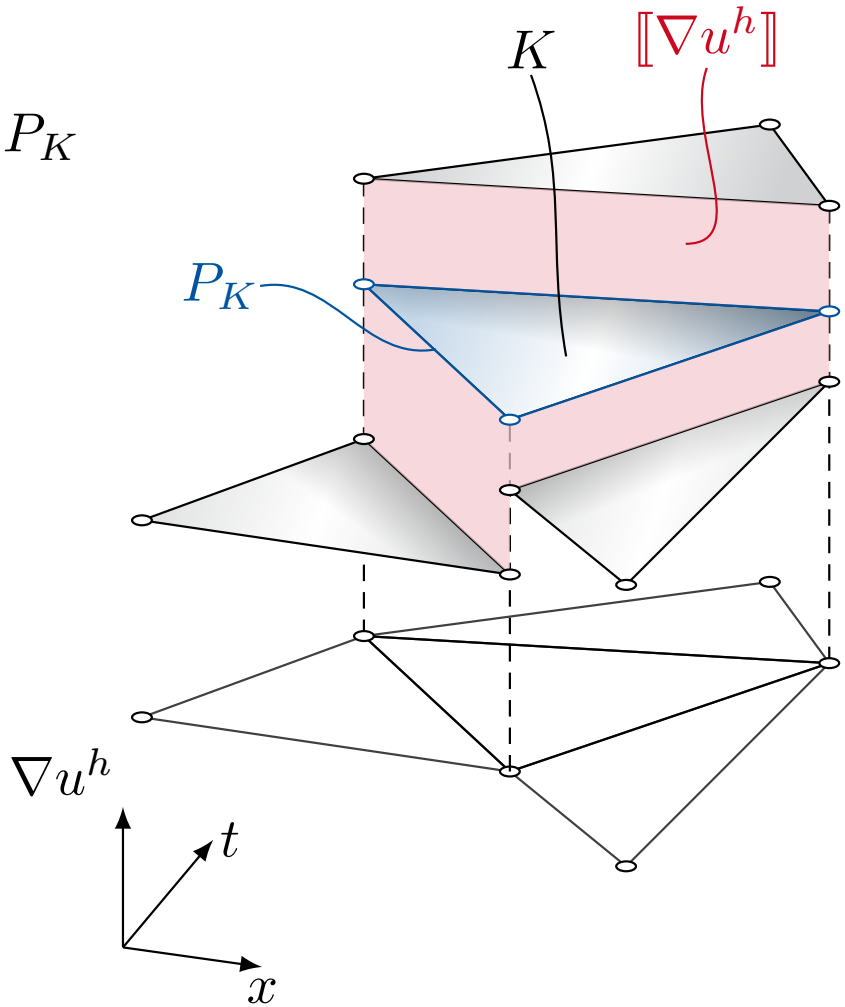
$$[\![\nabla u^h]\!] = \nabla u^h_- - \nabla u^h_+ \neq 0 \quad \forall (\boldsymbol{x}, t) \in P_K$$

- Space-time element boundary P_K
- Solution gradient jump $[\![\nabla u^h]\!]$

Kelly error estimator

$$\eta(K) = c_K \int_{P_K} [\![\nabla u^h]\!] \, dP_K$$

- Constant factor $c_K \propto |P_K|$



2D+t adaptivity – isotropic

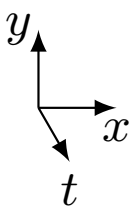
Flow around moving cylinder

Velocity in x-direction

Kelly error indicators η

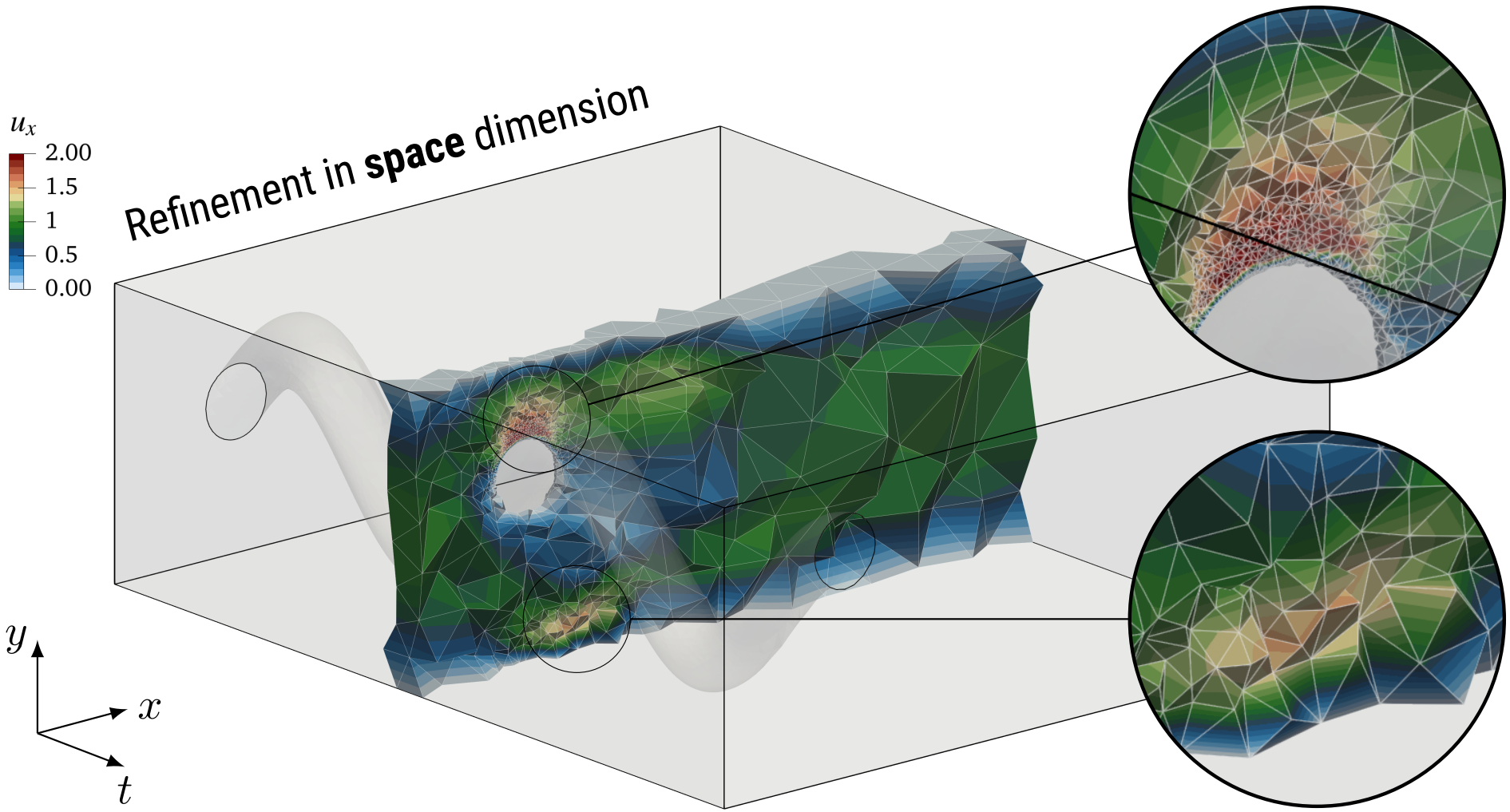
initial mesh

refined mesh



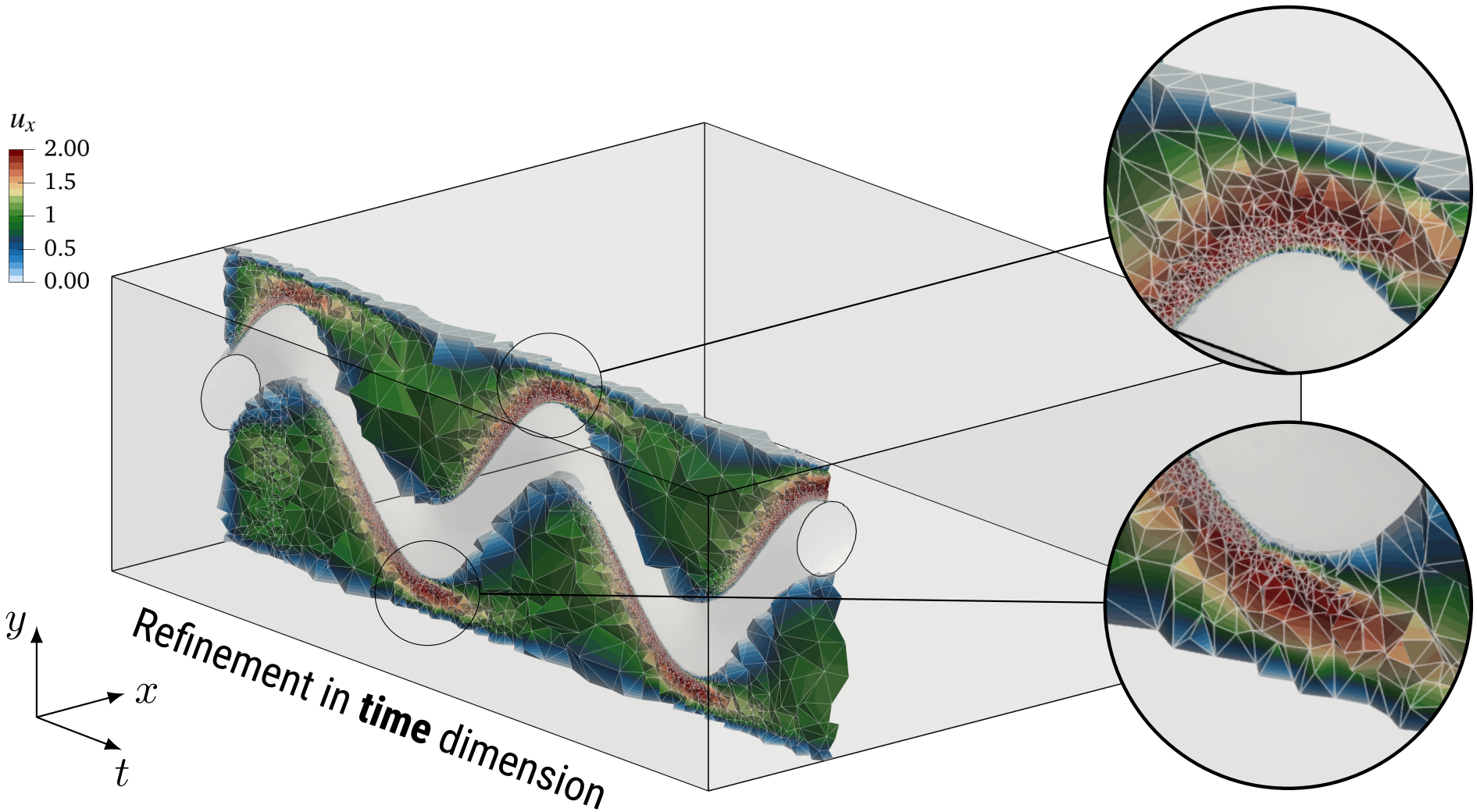
2D+t adaptivity – isotropic

Flow around moving cylinder



2D+t adaptivity – isotropic

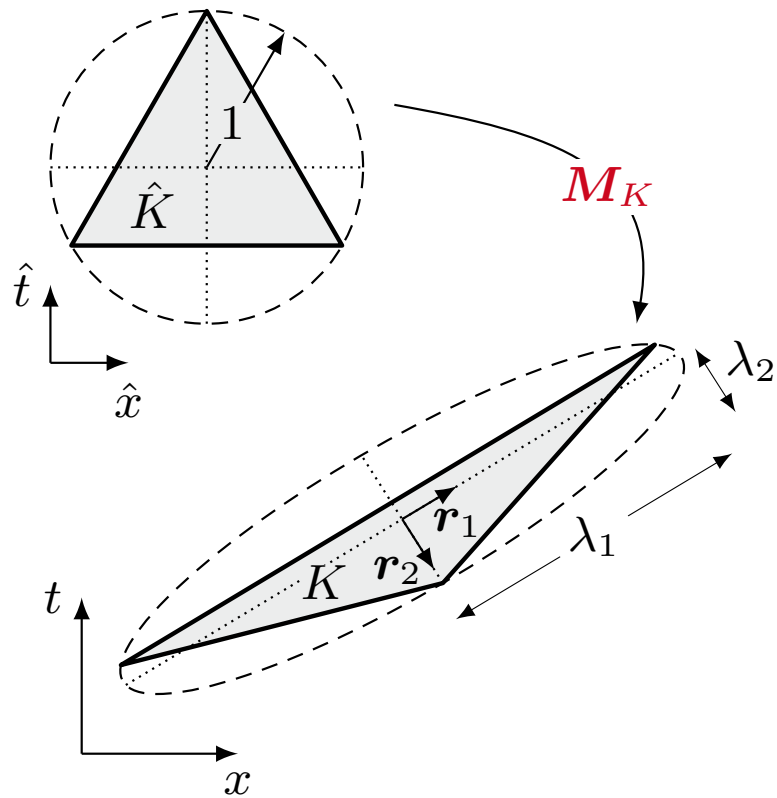
Flow around moving cylinder



2D+t adaptivity – anisotropic

Why anisotropic?

- Adapt mesh differently in x than in t
- Or even differently in all mesh dims.
- Find optimal mesh for a given u^h



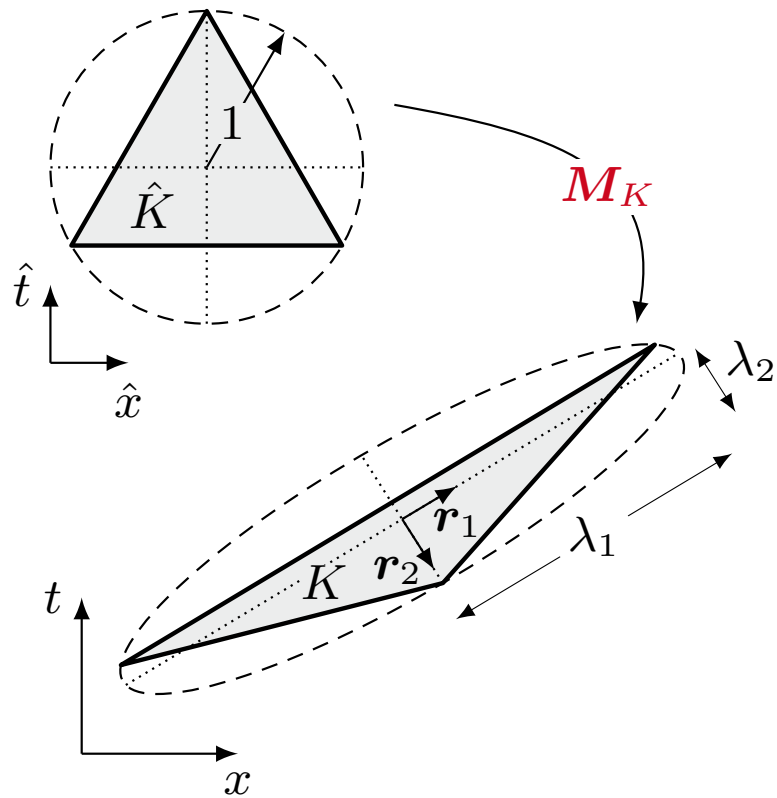
2D+t adaptivity – anisotropic

Why anisotropic?

- Adapt mesh differently in x than in t
- Or even differently in all mesh dims.
- Find optimal mesh for a given u^h

Element mapping

$$\begin{bmatrix} x \\ t \end{bmatrix} = M_K \begin{bmatrix} \hat{x} \\ \hat{t} \end{bmatrix} + b_K$$



2D+t adaptivity – anisotropic

Why anisotropic?

- Adapt mesh differently in x than in t
- Or even differently in all mesh dims.
- Find optimal mesh for a given u^h

Element mapping

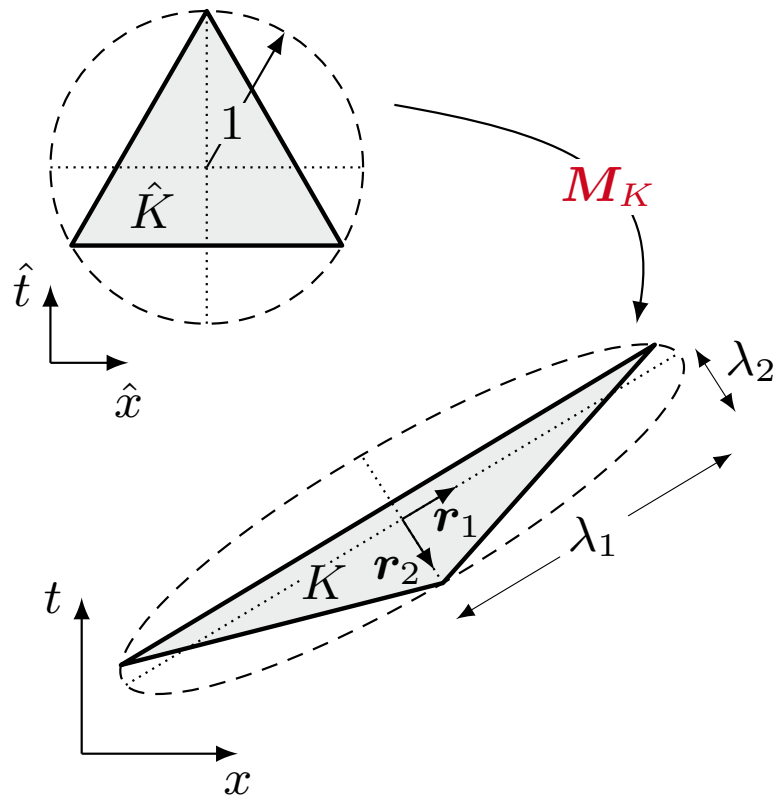
$$\begin{bmatrix} x \\ t \end{bmatrix} = M_K \begin{bmatrix} \hat{x} \\ \hat{t} \end{bmatrix} + b_K$$

$M_K = R_K^T \Lambda_K R_K Z_K$ - Jacobian metric

$\Lambda_K = \mathrm{diag}(\lambda_i)$ - size

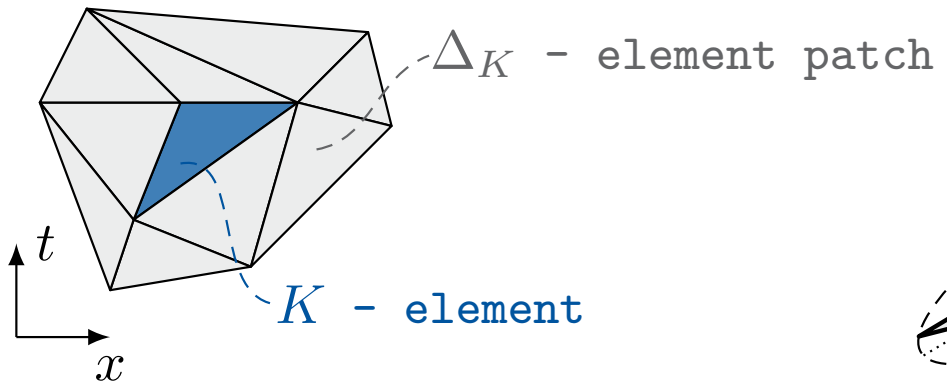
$R_K^T = [r_1, \ldots, r_d]$ - orientation

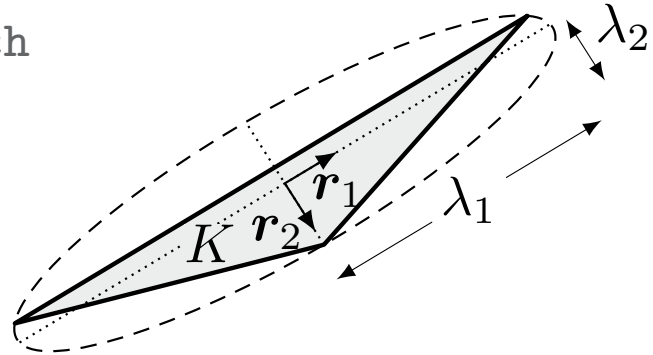
$i = 1, \ldots, d$



2D+t adaptivity – anisotropic

Anisotropic space-time error estimation





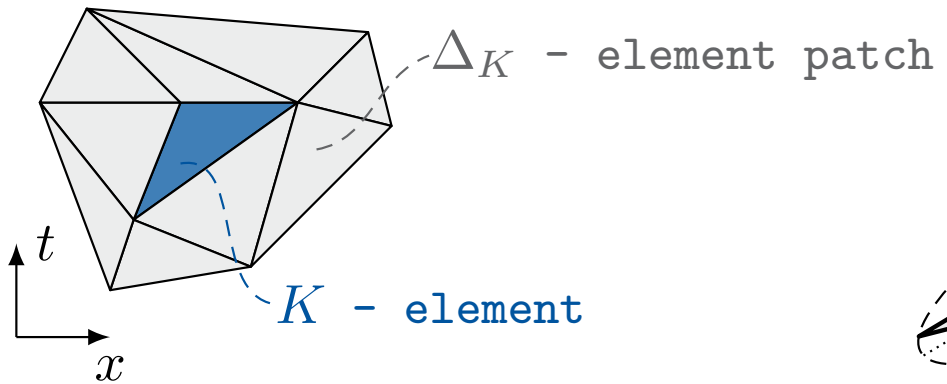
1. For each K, assemble a **second moment matrix** on the surrounding patch Δ_K

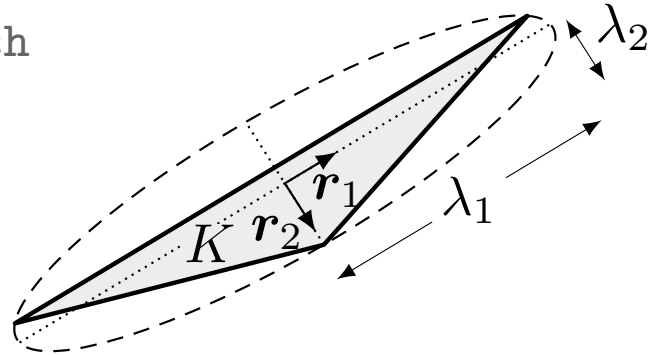
$$\boldsymbol{G}_{\Delta_K}(\boldsymbol{\eta}) = \sum_{T \in \Delta_K} \int_T \boldsymbol{\eta} \otimes \boldsymbol{\eta} \, dT \qquad \text{with} \qquad \boldsymbol{\eta} = \left[\eta_{\boldsymbol{x}}, \eta_t\right]$$

Original non-space-time version: [Formaggia and Perotto, 2001, Formaggia and Perotto, 2003]

2D+t adaptivity – anisotropic

Anisotropic space-time error estimation





1. For each K, assemble a **second moment matrix** on the surrounding patch Δ_K

$$\boldsymbol{G}_{\Delta_K}(\boldsymbol{\eta}) = \sum_{T \in \Delta_K} \int_T \boldsymbol{\eta} \otimes \boldsymbol{\eta} \, dT \qquad \text{with} \qquad \boldsymbol{\eta} = \left[\eta_{\boldsymbol{x}}, \eta_t\right]$$

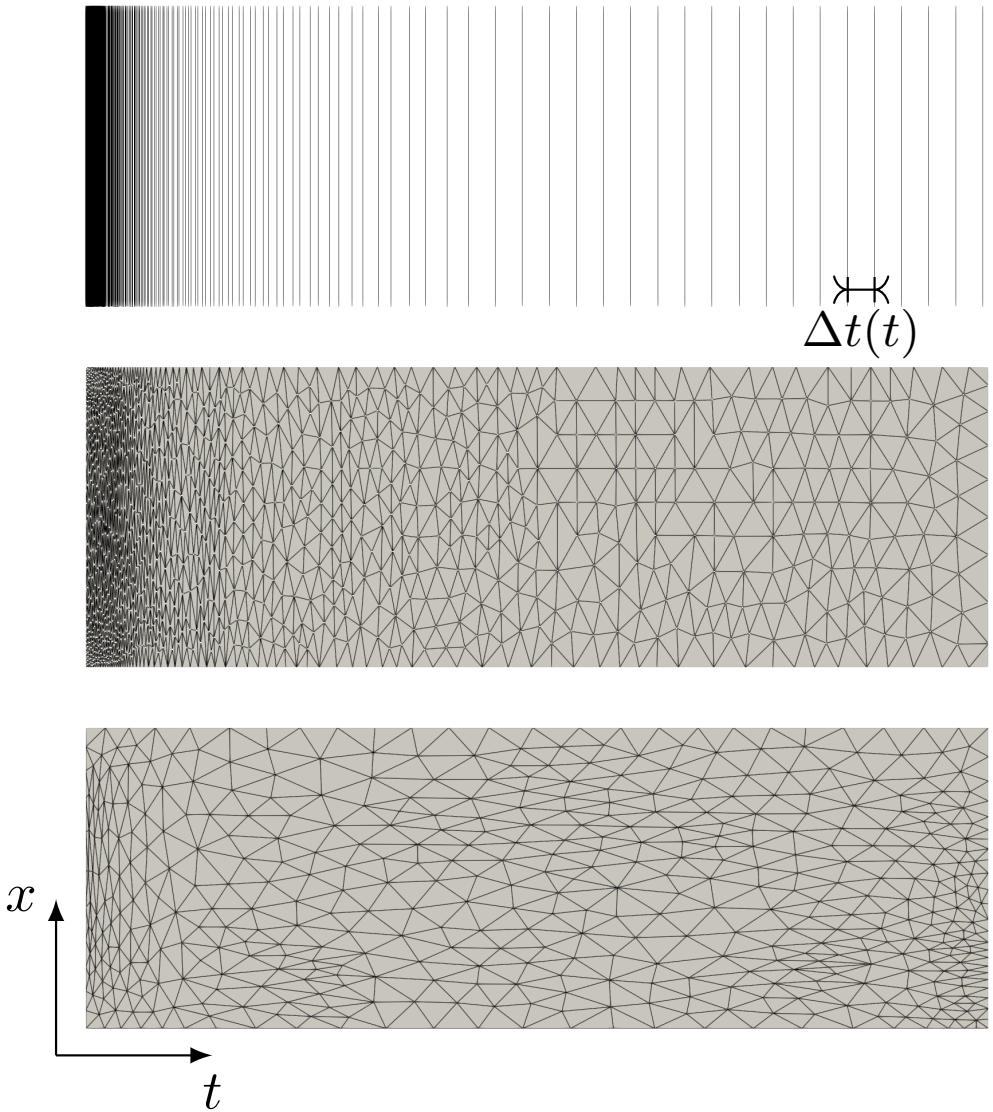
2. Use the eigenpairs $\{\boldsymbol{g}_i, \mu_i\}, i = 1, \ldots, d$ of G_{Δ_K} to construct new size field

$$\{\boldsymbol{g}_i, \mu_i\}_K \mapsto \{\boldsymbol{r}_i, \lambda_i\}_K$$

Original non-space-time version: [Formaggia and Perotto, 2001, Formaggia and Perotto, 2003]

2D+t adaptivity – anisotropic

From semi-discrete to unstructured space-time



Semi-discrete (Finite Differences)

- adaptive Δt

Semi-anisotropic space-time FE

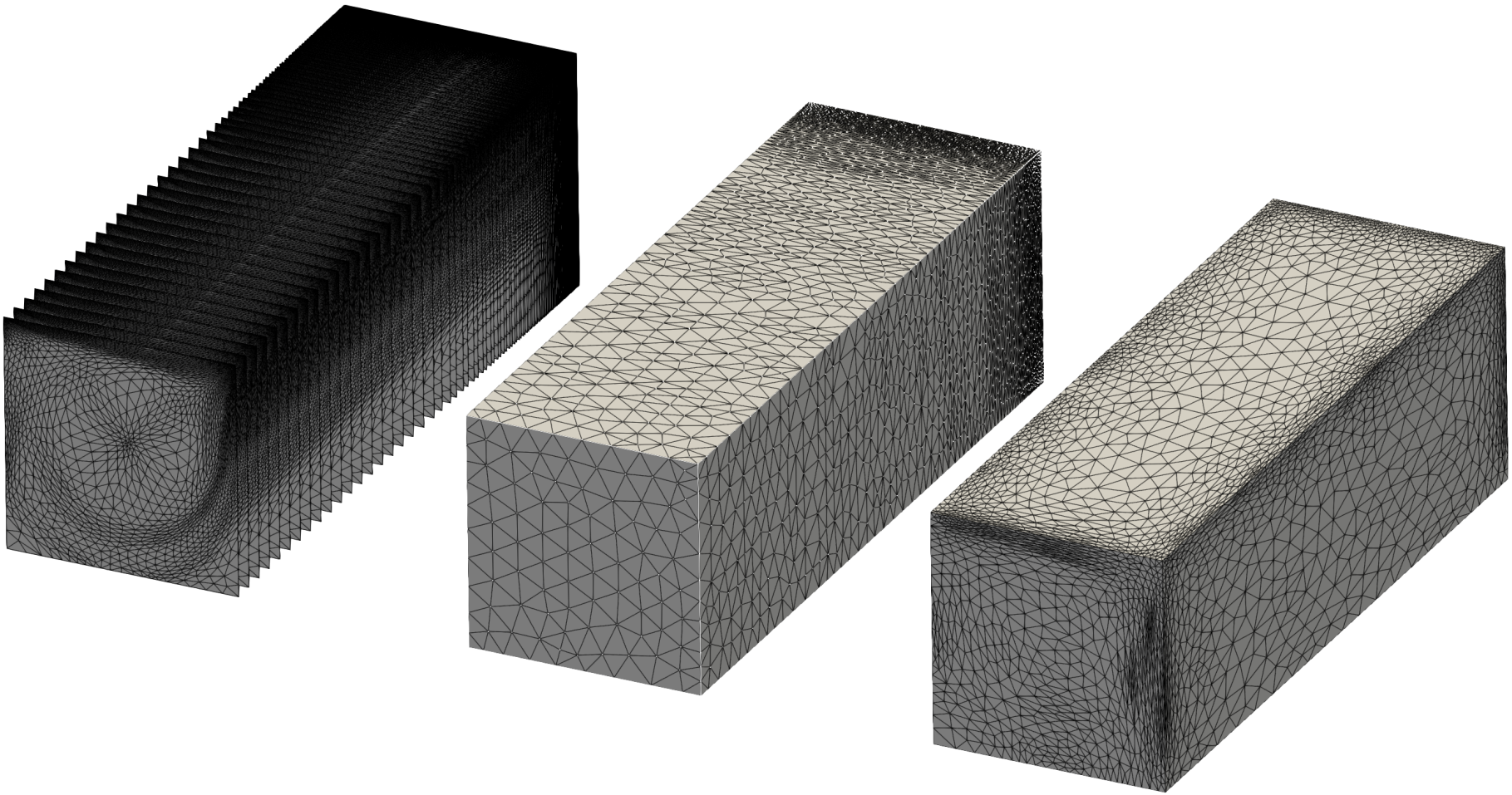
- spatially global Δt

Fully anisotropic space-time FE

- spatially local Δt

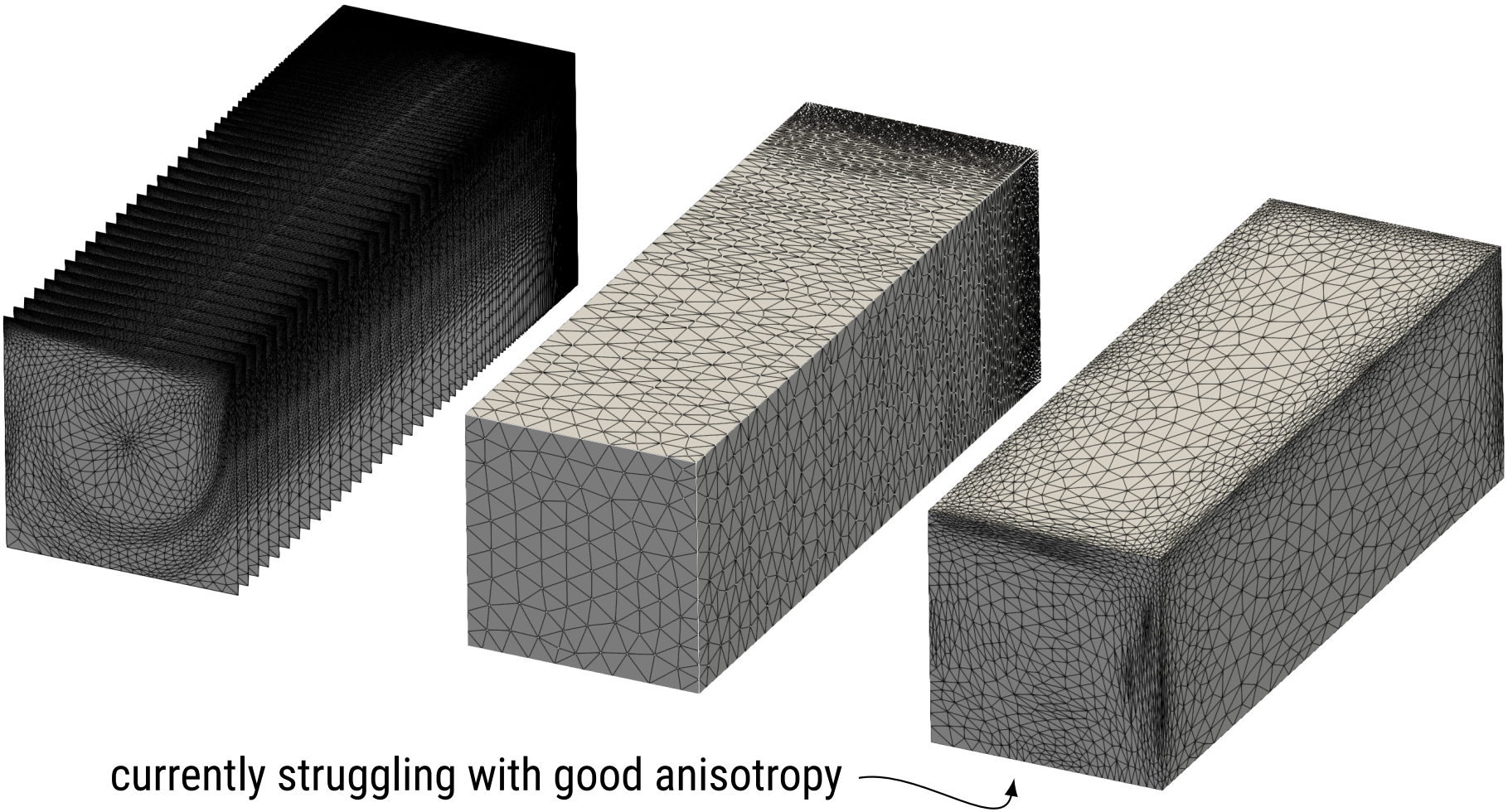
2D+t adaptivity – anisotropic

From semi-discrete to unstructured space-time



2D+t adaptivity – anisotropic

From semi-discrete to unstructured space-time



currently struggling with good anisotropy

Space-time side quests

Time-periodicity

The problem

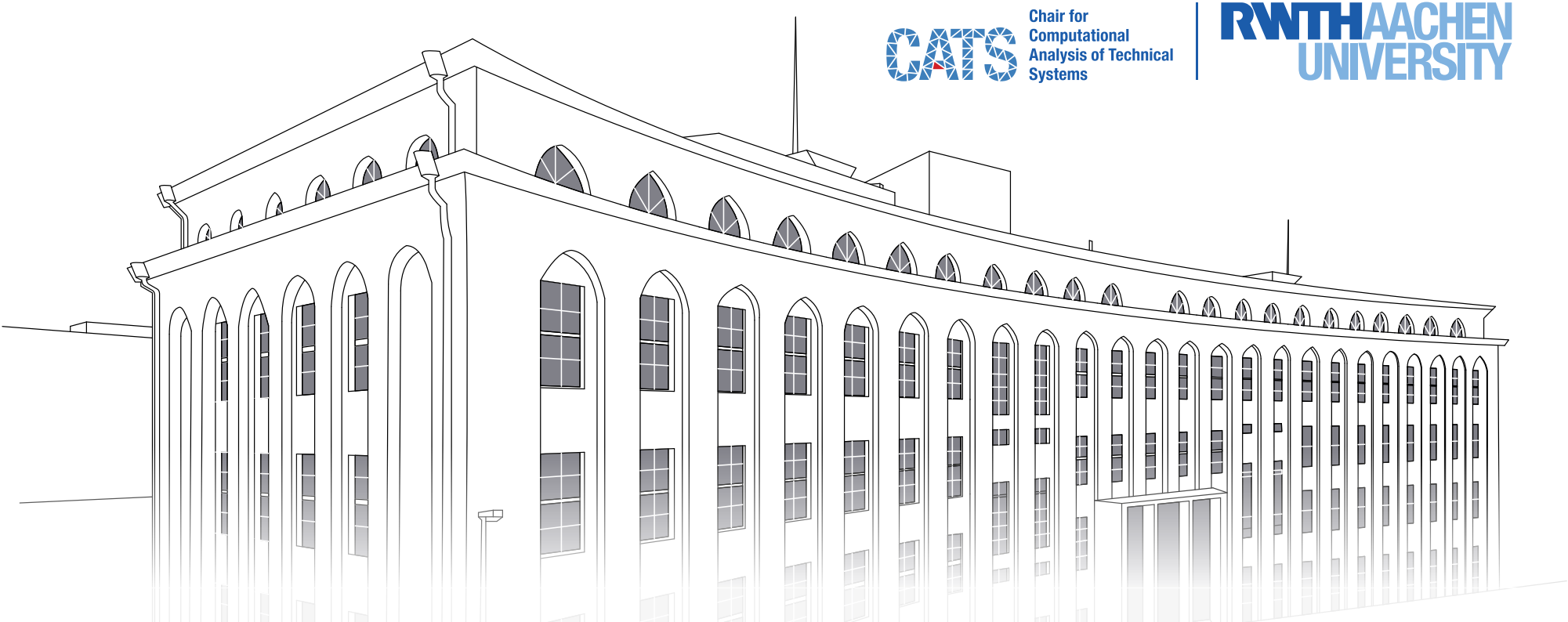
- The time period T has to be estimated a priori
- Finding T as part of solving the PDE problem `- in progress -`
- Phase condition has to be enforced `- in progress -`

Space-time side quests

Solid mechanics

Linear elasticity

Elasto-plasticity (local load substeps)



Looking forward to the discussion with you!

Oliver Wege, Norbert Hosters, Marek Behr

Chair for Computational Analysis of Technical Systems (CATS)
RWTH Aachen University, Aachen, Germany

Workshop on Triangulations and More General Meshes for Space-Time Applications — SoCG 2026, New Brunswick, NJ, USA — June 5, 2026

Selected references

Behr, M. (2008).
Simplex space–time meshes in finite element simulations.
International journal for numerical methods in fluids, 57(9):1421–1434.

De SR Gago, J., Kelly, D. W., Zienkiewicz, O. C., and Babuska, I. (1983).
A posteriori error analysis and adaptive processes in the finite element method: Part ii—adaptive mesh refinement.
International journal for numerical methods in engineering, 19(11):1621–1656.

Formaggia, L. and Perotto, S. (2001).
New anisotropic a priori error estimates.
Numerische Mathematik, 89(4):641–667.

Formaggia, L. and Perotto, S. (2003).
Anisotropic error estimates for elliptic problems.
Numerische Mathematik, 94(1):67–92.

Foteinos, P. and Chrisochoides, N. (2015).
4d space–time delaunay meshing for medical images.
Engineering with Computers, 31(3):499–511.

Kelly, D. W., De SR Gago, J., Zienkiewicz, O. C., and Babuska, I. (1983).
A posteriori error analysis and adaptive processes in the finite element method: Part i—error analysis.
International journal for numerical methods in engineering, 19(11):1593–1619.

Von Danwitz, M., Karyofylli, V., Hosters, N., and Behr, M. (2019).
Simplex space-time meshes in compressible flow simulations.
International Journal for Numerical Methods in Fluids, 91(1):29–48.

Von Danwitz, M., Voulis, I., Hosters, N., and Behr, M. (2023).
Time-continuous and time-discontinuous space-time finite elements for advection-diffusion problems.
International Journal for Numerical Methods in Engineering, 124(14):3117–3144.