RVE-based Micromechanical Analysis of Fiber-Matrix Debonding in Thin Ply FRPC Laminates

Finite Element Model: Geometry and Discretization

L. Di Stasio[1,2], Z. Ayadi[1], J. Varna[2]

[1] EEIGM, Université de Lorraine, Nancy, France

[2] Division of Materials Science, Luleå University of Technology, Luleå, Sweden

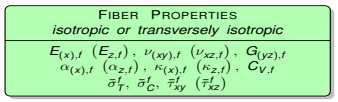
May 26, 2016

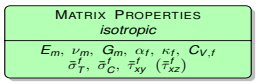
Outline

- The kinks in thin ply mechanics
- Geometries
- Body-fitted grids for the simulation of fracture propagation in curved domains
- Appendices

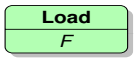
THIN PLY MECHANICS

Work-flow of composite structural design





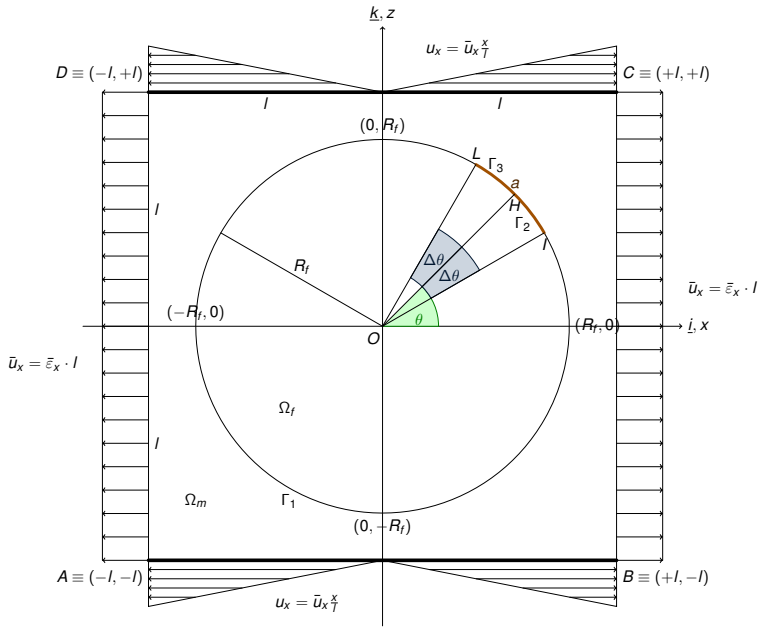
Failure load analysis



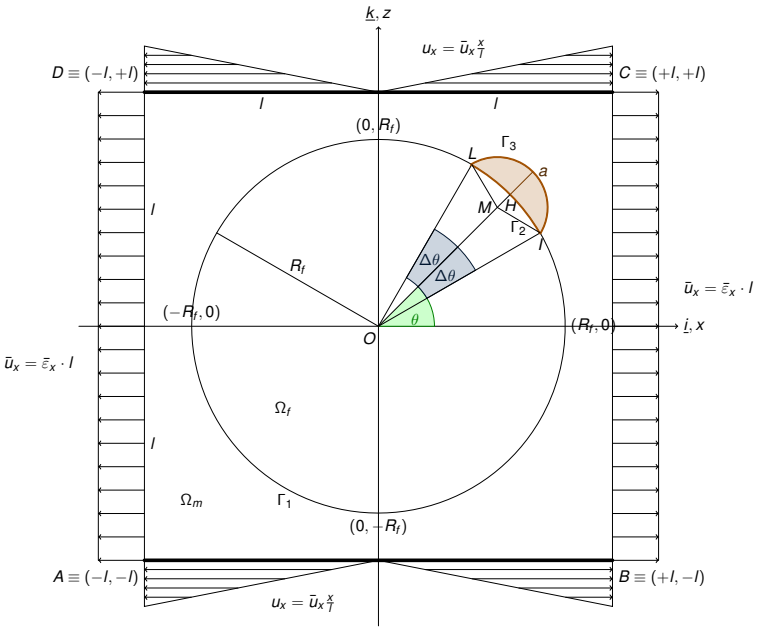
Progressive load analysis

GEOMETRIES

Single RVE model

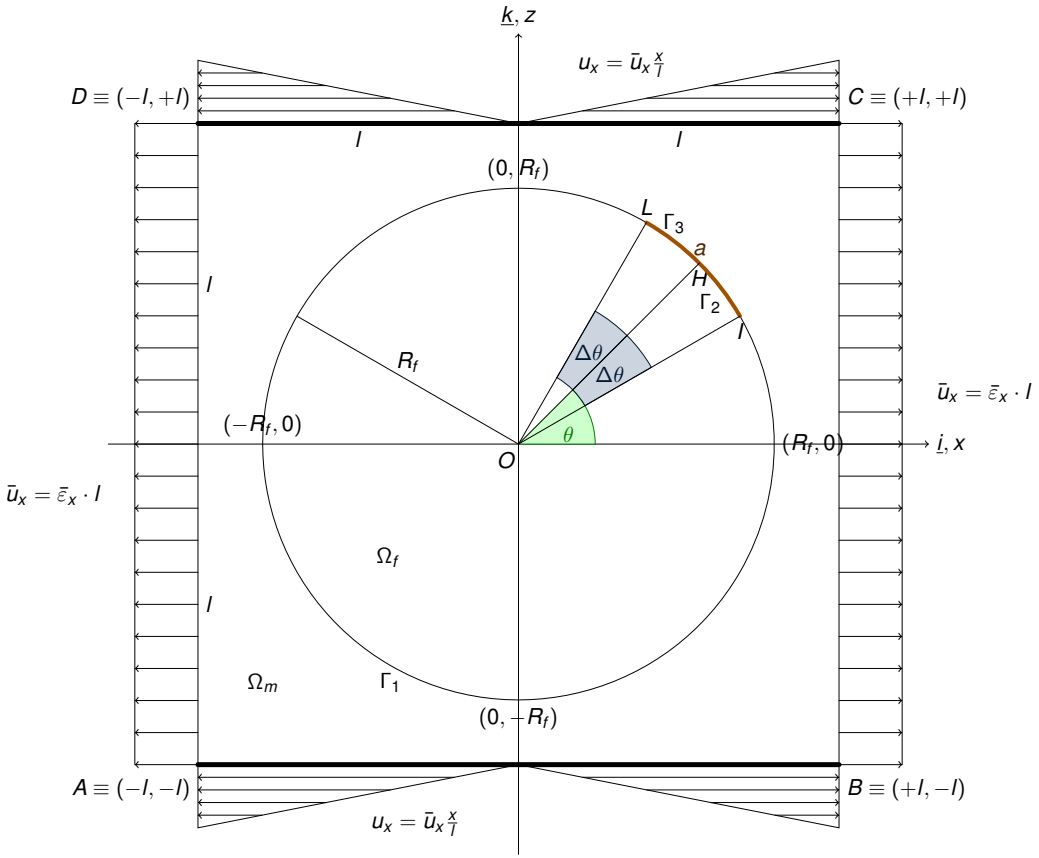


(a) Crack closed in the radial direction.



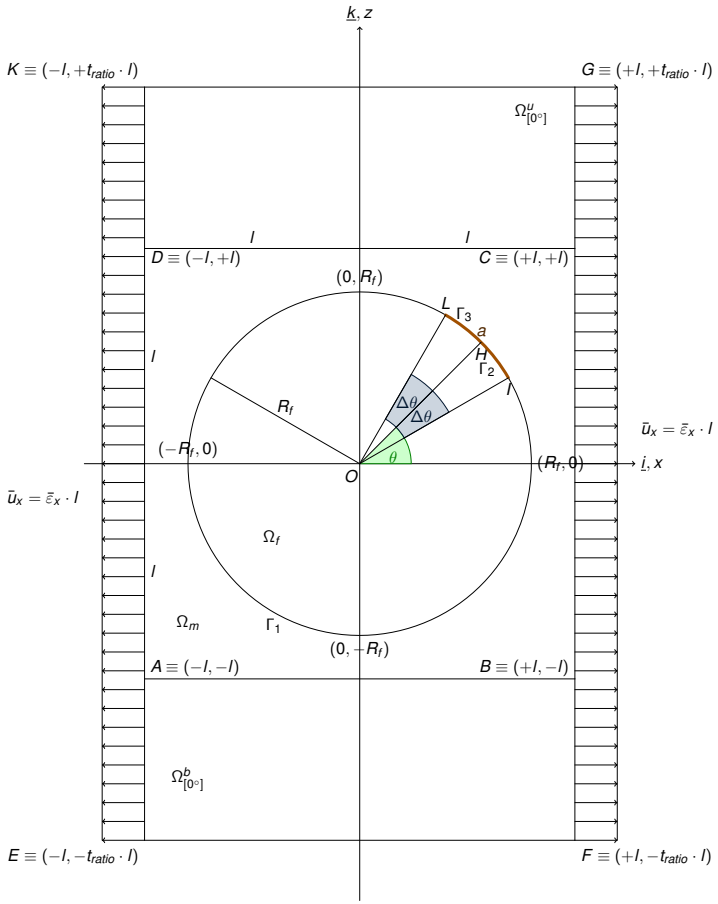
(b) Crack open in the radial direction.

Single RVE model

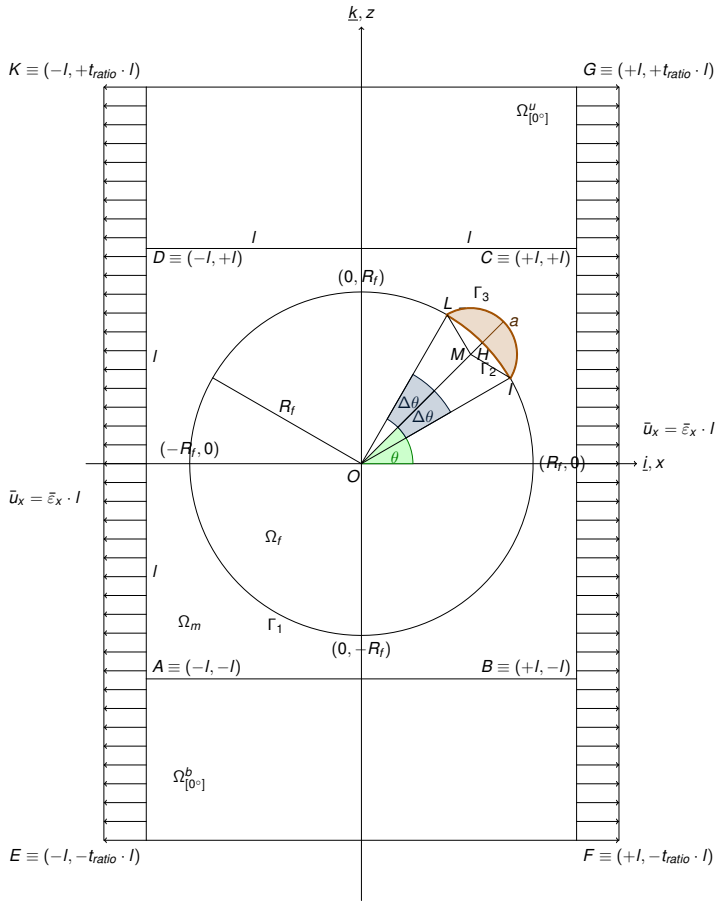


Initial state of single RVE model: crack closed in the radial direction.

Bounded RVE model

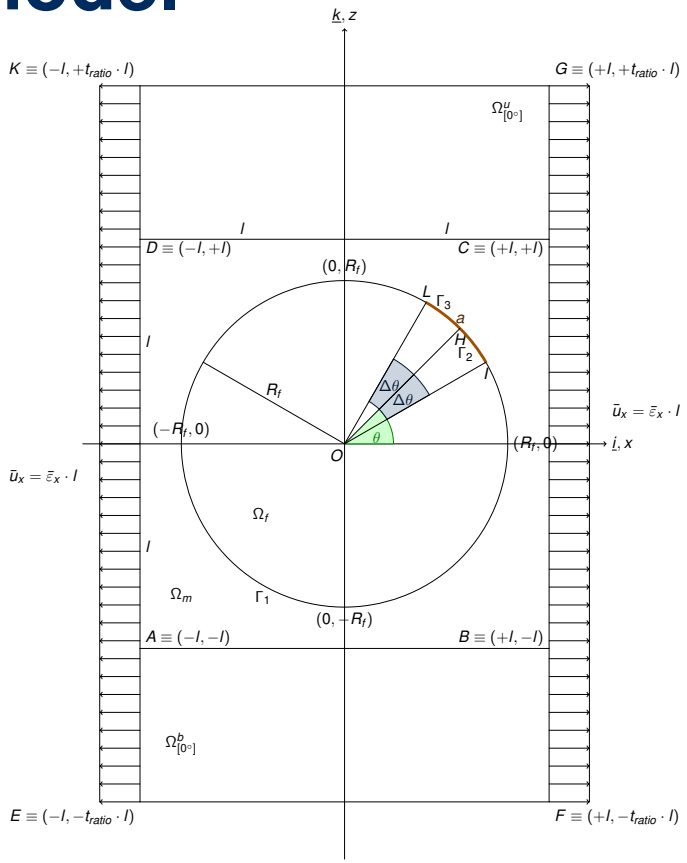


(a) Crack closed in the radial direction.



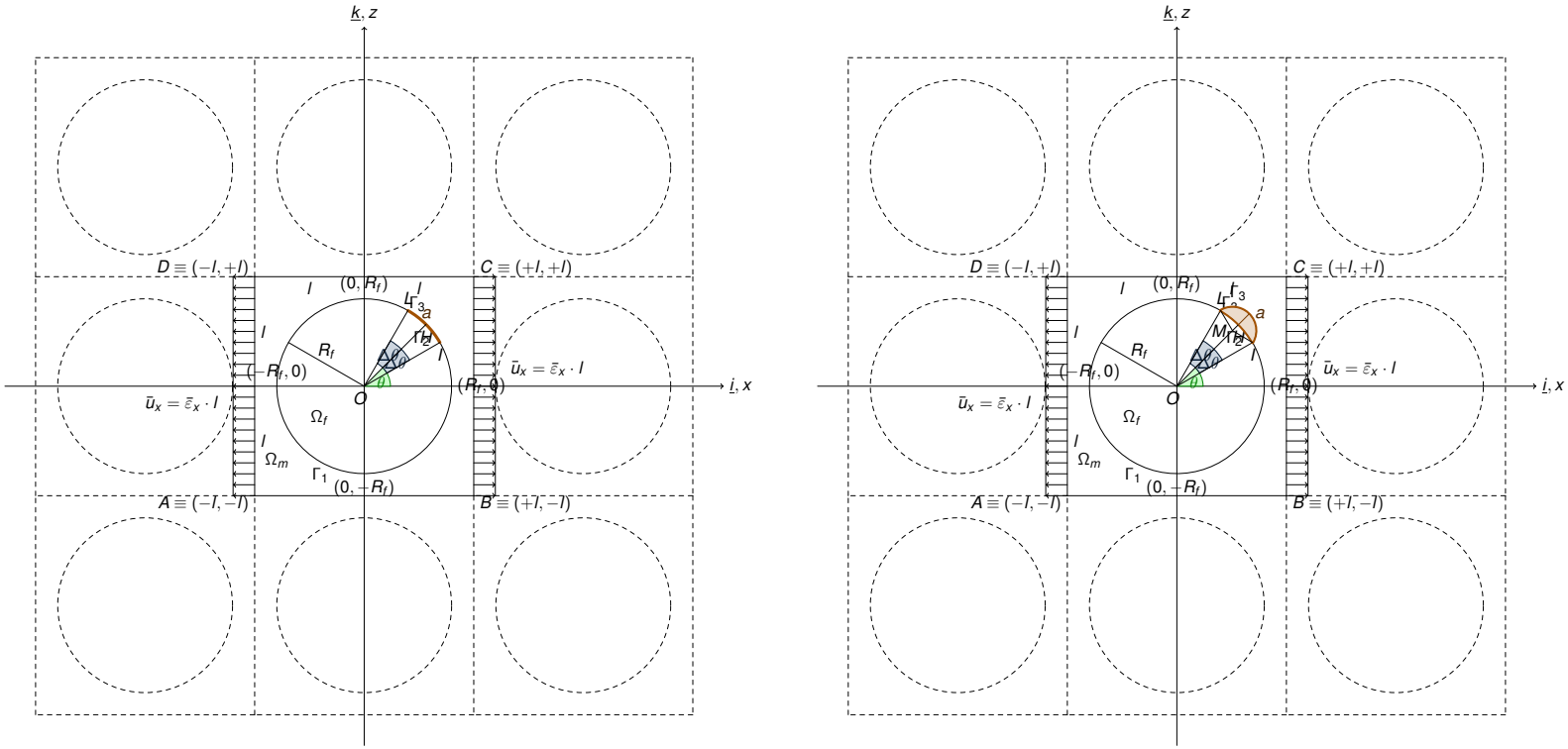
(b) Crack open in the radial direction.

Bounded RVE model



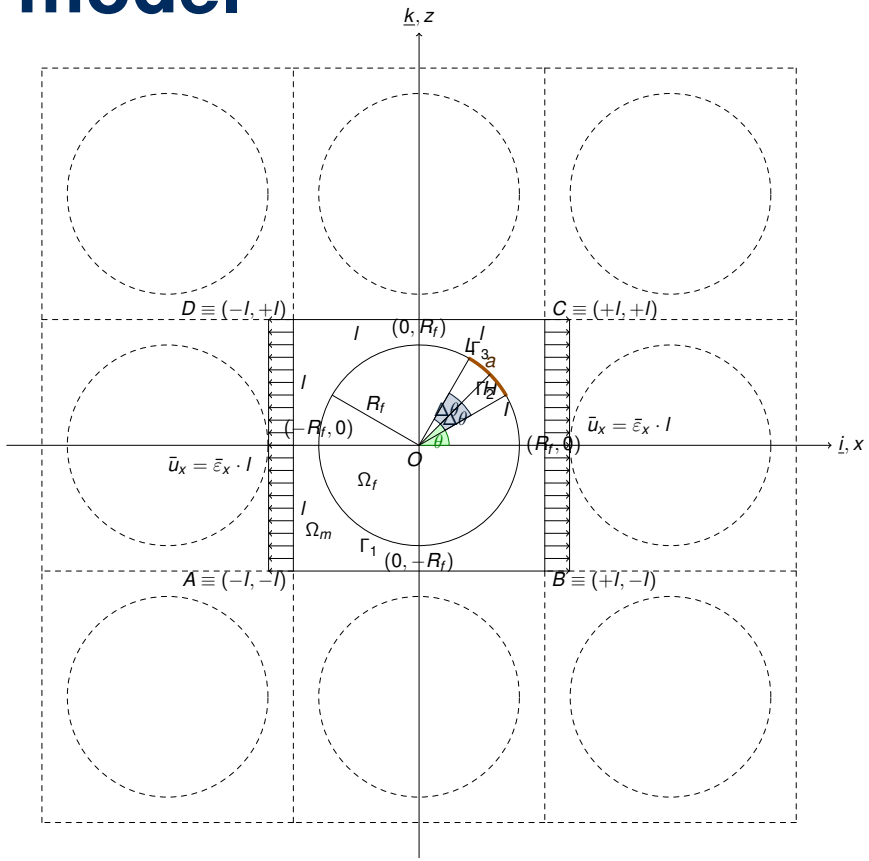
Initial state of bounded RVE model: crack closed in the radial direction.

Periodic RVE model



(a) Crack closed in the radial direction. (b) Crack open in the radial direction.

Periodic RVE model



Initial state of periodic RVE model: crack closed in the radial direction.

Summary of designed geometries

Name	Number of phases
single-RVE	2

Description

Circular fiber inside a square matrix domain.

Geometry of each phase

Fiber: circular; matrix: square with circular inclusion at its center.

Boundary conditions

Constant strain at $z = \pm l$; in order to have constant strain, the displacement has a linear functional form, i.e. $u_x|_{z=\pm l} = \bar{u}_x \frac{x}{l}$.

Imposed conditions

Constant displacement $u_x|_{z=\pm l} = \bar{u}_x = \bar{\varepsilon}_x \cdot l$ at $x = \pm l$.

Summary of designed geometries

Name	Number of phases
bounded-RVE	3

Description

Circular fiber inside a square matrix domain, bounded by two UD rectangular domains on the upper and lower side.

Geometry of each phase

Fiber: circular; matrix: square with circular inclusion at its center; UD: rectangular.

Boundary conditions

Free surface at $z = \pm l$.

Imposed conditions

Constant displacement $u_x|_{z=\pm l} = \bar{u}_x = \bar{\varepsilon}_x \cdot l$ at $x = \pm l$.

Summary of designed geometries

Name	Number of phases
periodic-RVE	2

Description

Periodically repeated unit cell, constituted by a circular fiber inside a square matrix domain.

Geometry of each phase

Fiber: circular; matrix: square with circular inclusion at its center.

Boundary conditions

Periodic boundary conditions on all sides.

Imposed conditions

Constant displacement $u_x|_{z=\pm l} = \bar{u}_x = \bar{\varepsilon}_x \cdot l$ at $x = \pm l$.

BODY-FITTED GRIDS

The issues

- Mesh characteristics (elements' size and distribution, irregularities) affect the behaviour of numerical models used to simulate fracture propagation
- The presence of curved boundaries induce irregularities in the mesh, using quadrilaterals as well as triangular elements

The issues

- Abaqus softwares' package provides powerful algorithms for the Finite Elements Method (global to local node mapping, application of load and boundary conditions, solvers)
- On the other hand, the meshing capabilities offered by Abaqus are restricted
  - Using Abaqus/CAE, mesh parameterization is not possible and the meshing process is controlled only through a few parameters, no access to the algorithms is provided
  - Abaqus Python scripting grants some degree of parameterization, but it allows the automation of CAE commands; thus, there is no direct access to the meshing algorithms
  - Abaqus input file scripting provides greater access to the meshing algorithms and allows for parameterization of simple geometries; complex arrangements are much more difficult, or even impossible, to parameterize using this kind of scripting

The issues

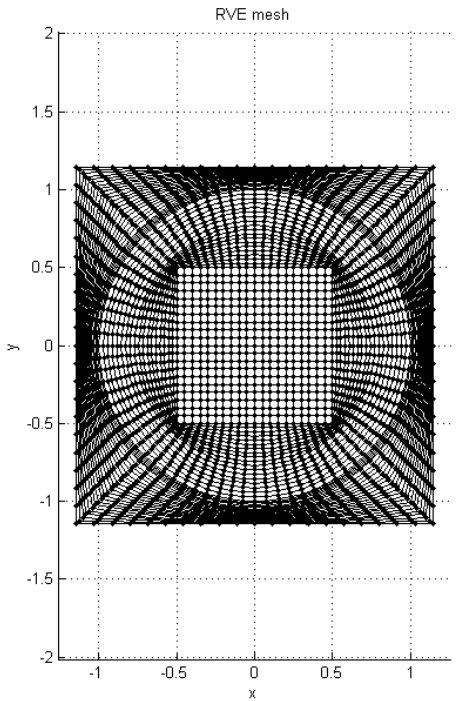
- Tools for the analysis and control of the mesh are limited in Abaqus
- No field variable is provided to measure the quality of the mesh

The objectives

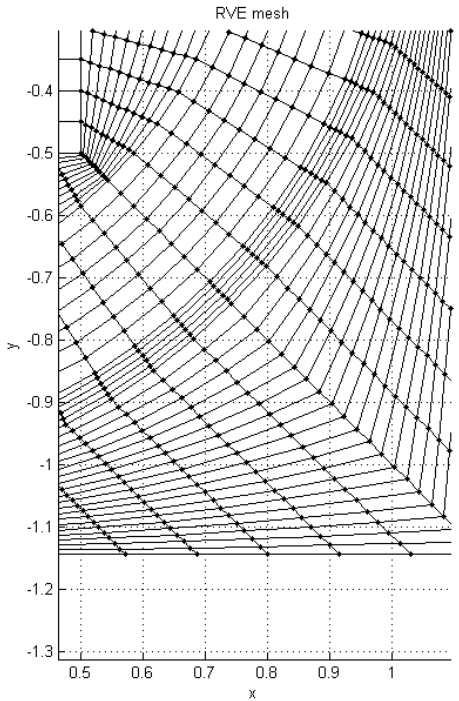
- Develop a toolbox of algorithms to:

1. Develop a toolbox of algorithms to:
2. generate meshes of curvilinear domains in 2D and 3D
3. provide control of the mesh down to the single node/element
4. allow for efficient parameterization of the grid generation process
5. provide quantitative tools for the analysis of mesh quality properties

The result

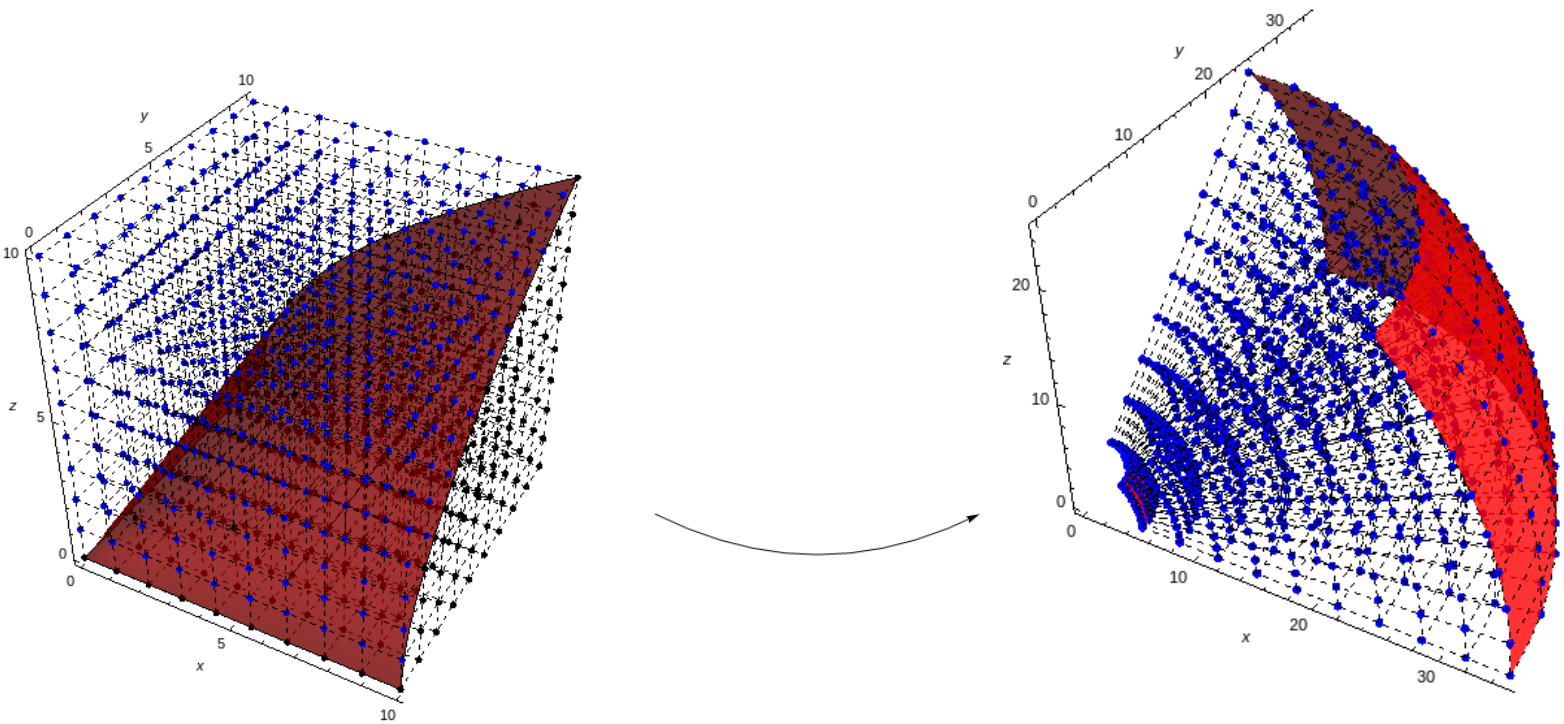


The result



Fundamental ideas

One-to-one mapping



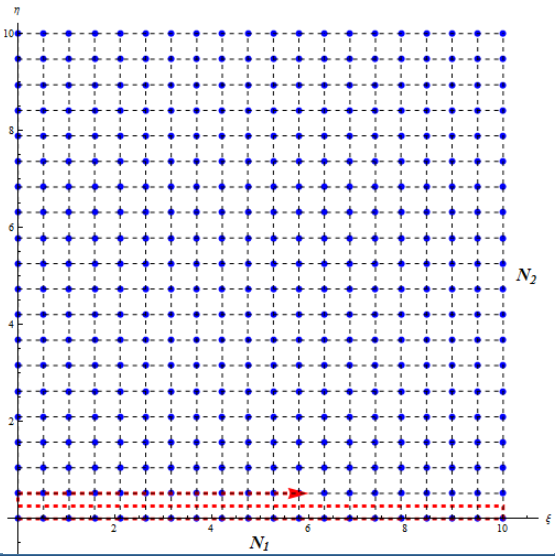
Computational domain

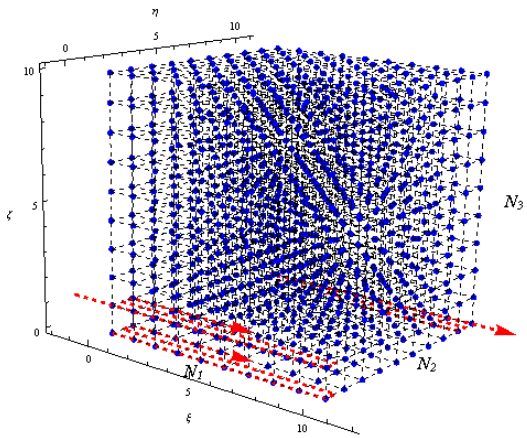
Physical domain

Fundamental ideas

Helical numbering

- N_1, N_2, N_3 number of grid points respectively along $\xi-$, $\eta-$, $\zeta-$ direction in computational space
- Total number of points $N = N_1 \cdot N_2$ in 2D and $N = N_1 \cdot N_2 \cdot N_3$ in 3D
- Helical numbering: given $n \in [1, N]$, $i \in [1, N_1]$, $j \in [1, N_2]$, $k \in [1, N_3]$, $n = i + (j-1) \cdot N_1$ in 2D and $n = i + (j-1) \cdot N_1 + (k-1) \cdot N_1 \cdot N_2$ in 3D





Mesh generation

Numerical strategy for structured grid generation in curved geometries - 3D (2D)

1. The boundary is generated patching analytical parameterizations
2. The boundary is split into a set of 8 corners (c_i), 12 edges (e_i), 4 surfaces (f_i)
3. Interior nodes are created applying transfinite interpolation using multi-dimensional linear Lagrangian or Hermite interpolants

$$P_1(x, p_j) = \sum_{j=1}^{n} p_j \prod_{k=1,\ k\neq j}^{n} \frac{x - x_k}{x_j - x_k}$$

$$P_2(x, y, p_j, q_j) = P_1(x, p_j) \otimes P_1(y, q_j) \quad P_3(x, y, z, p_j, q_j, r_j) = P_1(x, p_j) \otimes P_1(y, q_j) \otimes P_1(z, r_j)$$

$$\begin{aligned} r_i(\xi, \eta, \zeta) = {} & P_1(\xi, f_5, f_3) + P_1(\eta, f_2, f_4) + P_1(\zeta, f_1, f_6) - P_2(\xi, \eta, c_1, c_2, c_3, c_4) + \\ & - P_2(\xi, \eta, e_5, e_6, e_7, e_8) - P_2(\xi, \zeta, e_2, e_4, e_{10}, e_{12}) - P_2(\eta, \zeta, e_1, e_3, e_9, e_{11}) + \\ & + P_3(\xi, \eta, \zeta, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) \end{aligned}$$

4. The mesh is smoothed applying elliptic mesh generation

$$g^{11}\underline{r}_{,\xi\xi} + 2g^{12}\underline{r}_{,\xi\eta} + g^{22}\underline{r}_{,\eta\eta} + 2g^{13}\underline{r}_{,\xi\zeta} + g^{33}\underline{r}_{,\zeta\zeta} + 2g^{23}\underline{r}_{,\eta\zeta} = 0$$

Mesh generation

Numerical strategy for structured grid generation in curved geometries More

- For both transfinite interpolation and elliptic mesh generation, Dirichlet boundary conditions are assigned, i.e. boundaries are fixed
- Elliptic mesh generation is solved iteratively: at step n, the contravariant metric components are computed based on configuration at step $n-1$ and afterwards the configuration is updated
- Partial differential equations are discretized using second order central finite difference scheme for second derivatives:

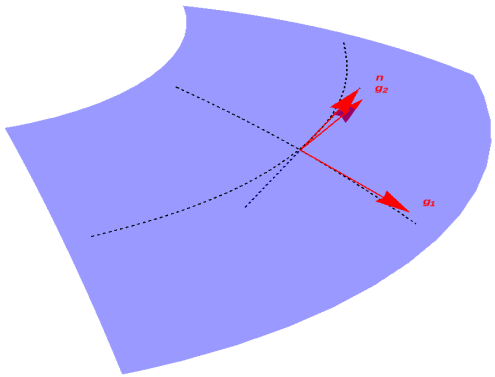
$$\frac{\partial^2 u}{\partial (q^i)^2}|_{q^i} = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta q^i)^2} - \frac{(\Delta q^i)^2}{12}\frac{\partial^4 u}{\partial (q^i)^4}|_{q^i} + \dots$$

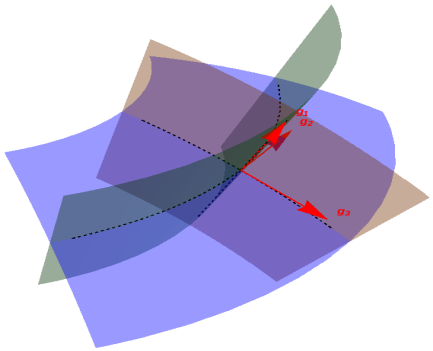
An algebraic system of equations $\underline{\underline{A}}\underline{x} = \underline{b}$ is obtained; it is solved using built-in libraries in Matlab, and implementing an iterative Jacobian-Free Newton-Krylov (JFNK) method in C++

- Management of boundaries is done exclusively using indices, not using coordinates: indices of nodes on corners, edges, faces are stored at the beginning of computation

Differential geometry

Computation of geometric properties of mappings





- Given analytical parameterization, using symbolic computing (Mathematica)
- Given discrete data, using numerical computing (Matlab, C++)
- For 2D and 3D geometries, surfaces in 3D space

Differential geometry

$$\underline{R} = r_i = r_i(q^j)$$

$$\underline{g}_i = \frac{\partial \underline{R}}{\partial q^i}$$

$$\underline{g}_3 = \underline{g}_1 \times \underline{g}_2 \quad \text{for surfaces}$$

$$g_{ij} = \underline{g}_i \cdot \underline{g}_j$$

$$g = det(g_{ij})$$

$$\underline{g}^i = \frac{\underline{g}_j \times \underline{g}_k}{\sqrt{g}}$$

$$g^{ij} = \underline{g}^i \cdot \underline{g}^j$$

$$v_i = \underline{v} \cdot \underline{g}_i \quad v^i = \underline{v} \cdot \underline{g}^i$$

$$\underline{v} = \sum_i v_i \underline{g}^i \quad \underline{v} = \sum_i v^i \underline{g}_i$$

$$[ij, k] = \underline{g}_{i,j} \cdot \underline{g}_k$$

$$\Gamma^k_{ij} = \underline{g}_{i,j} \cdot \underline{g}^k$$

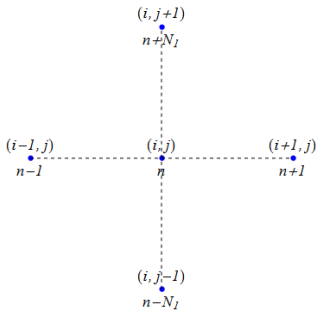
$$R^l_{ijk} = \Gamma^l_{ik,j} - \Gamma^l_{ij,k} + \Gamma^l_{js}\Gamma^s_{ik} - \Gamma^l_{ks}\Gamma^s_{ij}$$

$$R_{iklm} = g_{ls} R^s_{ijk}$$

$$R_{ij} = g^{lm} R_{iljm}$$

$$R = g^{ij} R_{ij}$$

Differential geometry



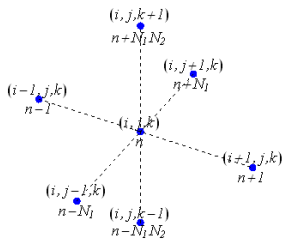
- First derivatives computed with second order finite differences

$$\frac{\partial u}{\partial q^i}|_{q^i} = \frac{u_{i+1} - u_{i-1}}{2\Delta q^i} - \frac{(\Delta q^i)^2}{6}\frac{\partial^3 u}{\partial (q^i)^3}|_{q^i} + \dots$$

- Right- and left- sided second order finite differences at boundary

$$\frac{\partial u}{\partial q^i}|_{q^i} = \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2\Delta q^i} + \frac{(\Delta q^i)^2}{3}\frac{\partial^3 u}{\partial (q^i)^3}|_{q^i} + \dots$$

$$\frac{\partial u}{\partial q^i}|_{q^i} = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2\Delta q^i} + \frac{(\Delta q^i)^2}{3}\frac{\partial^3 u}{\partial (q^i)^3}|_{q^i} + \dots$$



- Δq^i are assumed to be constant in space and time but could be different between different directions, as considered in the computational space

Additional details

LULEÅ UNIVERSITY OF TECHNOLOGY

Doc MASE

UNIVERSITÉ DE LORRAINE | EEIGM

THANK YOU!

APPENDICES

Differential geometry

Details on Matlab implementation

- In array *firstdevneighbours* of dimension $N \times 4D$, neighbours and type of finite difference scheme (encoded with an integer mask, from 1 to 3) are stored for each grid site
- Each geometric quantity is stored in array of dimension $N \times D^{n_{indices}}$; i.e. tensor components are stored in columns $N \times 1$ stacked together in a single global array
- Symmetry of geometric quantities (for example, $g_{ij} = g_{ji}$) is exploited: components that can be recovered by symmetry are not saved to disk

Back to Differential geometry

Differential geometry

Details on C++ implementation

Back to Differential geometry

Mesh generation

Numerical strategy for structured grid generation in curved geometries - 2D

1. The boundary is generated patching analytical parameterizations
2. The boundary is split into a set of 4 corners (c_i) and 4 edges (e_i)
3. Interior nodes are created applying transfinite interpolation using multi-dimensional linear Lagrangian interpolants

$$P_1(x, p_j) = \sum_{j=1}^{n} p_j \prod_{k=1\ k\neq j}^{n} \frac{x - x_k}{x_j - x_k} \qquad P_2(x, y, p_j, q_j) = P_1(x, p_j) \otimes P_1(y, q_j)$$

$$r(\xi, \eta) = P_1(\xi, e_2, e_4) + P_1(\eta, e_1, e_3) - P_2(\xi, \eta, c_1, c_2, c_3, c_4)$$

4. The mesh is smoothed applying elliptic mesh generation

$$g^{11}\underline{r}_{,\xi\xi} + 2g^{12}\underline{r}_{,\xi\eta} + g^{22}\underline{r}_{,\eta\eta} = 0$$

Back to Mesh generation

Geometry and mesh generation

Numerical strategy for monolithic domains

- Cells are indexed with the same criteria applied to nodes and can be accessed using look-up tables of such indices
- 3D geometries can be costructed directly using 3D transfinite interpolation and 3D elliptic smoothing or can be generated as extrusion of 2D geometries
- If needed, cells can be accessed and modified, i.e. creating triangles/tetrahedra instead of squares/cubes

Back to Mesh generation

Geometry and mesh generation

Numerical strategy for domains with random inclusions

- 2D and 3D Ising model is applied to generate a porous geometry, controlling the dimensions of structures through the 'temperature' of the sytem
- Nodes with positive magnetization are assigned to one material domain and those negative to the other material (or viceversa)
- Boundaries of the 2 domains are identified checking magnetization of neighbours
- Both domains are then separately smoothed using elliptic mesh generation iterations

Back to Mesh generation


Doc
MASE
UNIVERSITÉ
DE LORRAINE
EEIGM
LULEÅ
UNIVERSITY
OF TECHNOLOGY
Education and Culture
Erasmus Mundus