

Grid generation with Goat*



Developers: Sander Van den Kerkhof (grid generation, grid deformation), Nathan Vervloesem (grid adaptation)

*Grid Generation, Optimization and Adaptation Toolbox (goat/Goat/GOAT)

Context

- Unstructured FV framework in SOLPS ('wide grids' version)
- Emergence of novel topologies (snowflake, super-X divertor, 4X-point configuration, see talk Ivan Paradela Perez)
- Divertor shape optimization
- **Need for a gridding toolbox, capable of automated (re)meshing for arbitrary topologies**

What is GOAT?

- A flexible gridding toolbox that can deal with virtually any magnetic field and vessel geometry
- Intended use: aligned grids for plasma edge simulations
- Originally mostly developed by a one-man team in an academic context, but very happy to see other people getting involved

What is GOAT *not*?

- An industry-grade toolbox with fancy (G)UI
- Foolproof
- The solution to all of your simulation problems
- Tailored for SOLPS
- Fully mature: bugs are still found and issues may pop up (as with any code)

What is available in GOAT?

- **Three modules:**
 - **Grid generation** for arbitrary magnetic topologies and (quasi) arbitrary geometries
 - Optimization-based grid deformation
 - Grid adaptations
- **Only grid generation is currently mature enough** to be used by the public (others are at own risk)

In this presentation

- **Focus on grid generation mechanics** to understand the output you get
- No detailed info about **options/switches** → **hands-on session**
- Update on workflow integration with SOLPS-ITER

Outline

- Introduction
- Grid generator mechanics
- Workflow: from DivGeo to SOLPS simulation with kinetic neutrals & voids
- Conclusion & outlook

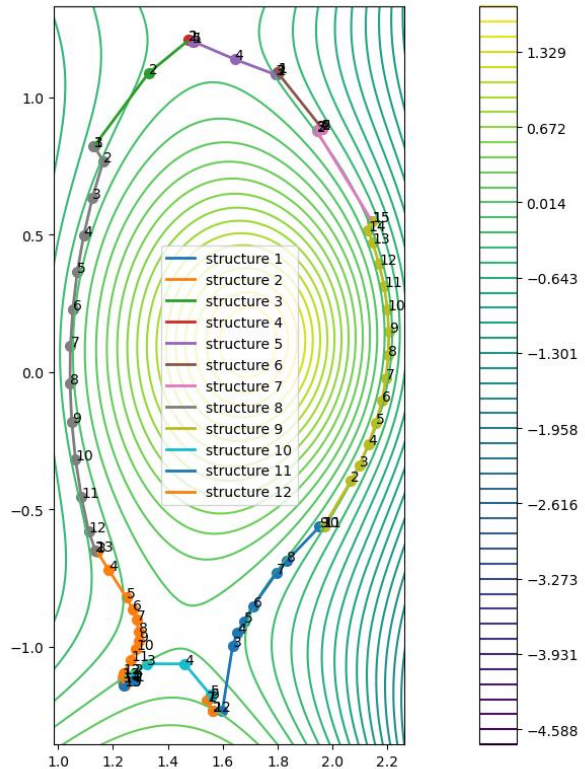
Why grid mechanics?

- Avoid trial-and-error grid generation as much as possible
- ‘Debug’ issues in the grid
- Be aware of limitations and caveats

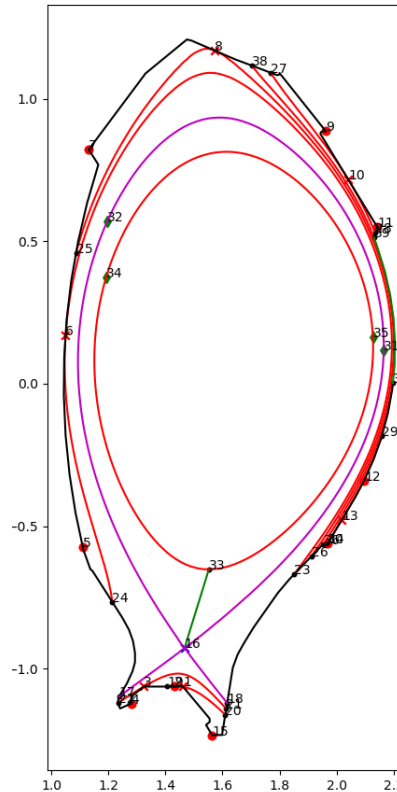
Why is gridding more complex than before?

- We're dealing with *general* topologies and geometries
- We need to be field-aligned
- We can't (yet!) allow local refinement in radial direction (global radial refinement is possible, i.e. adding more flux surfaces)
- We want to keep control on grid features near the wall

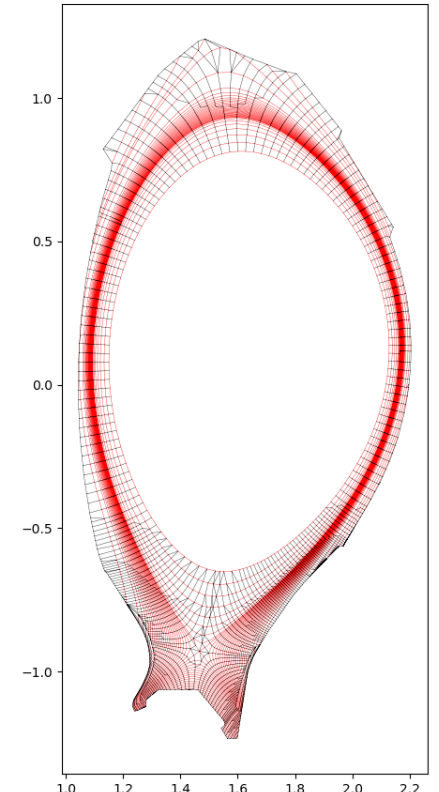
The two main steps



Input: vessel & magnetic field



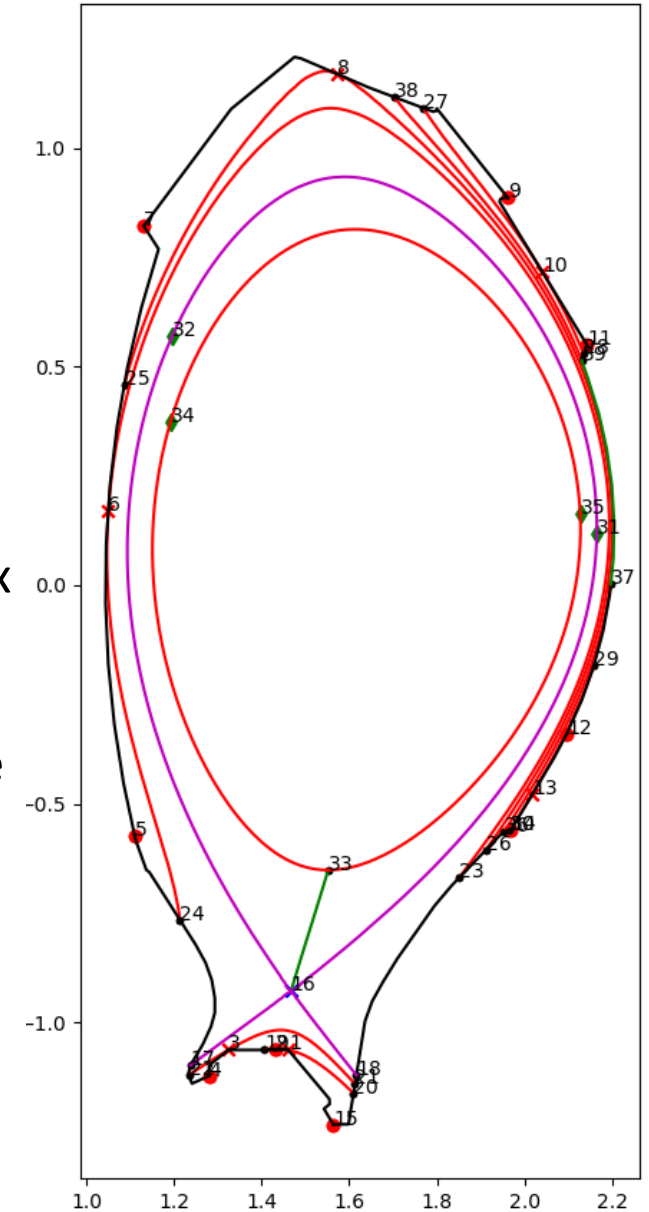
Topological mesh generation



Grid generation

Topomesh generation

- Partitions the 2D domain in **topologically rectangular regions**
- A region has 2 aligned and 2 non-aligned boundaries
- A boundary may consist of multiple topological faces or be a single vertex
→ a region may be rectangular but contain ≥ 3 topological vertices
- Why rectangular regions? → easy GG, since only one type of region to be meshed...
- **Again a two-step process:**
 - Base topomesh generation
 - Topomesh adaptations



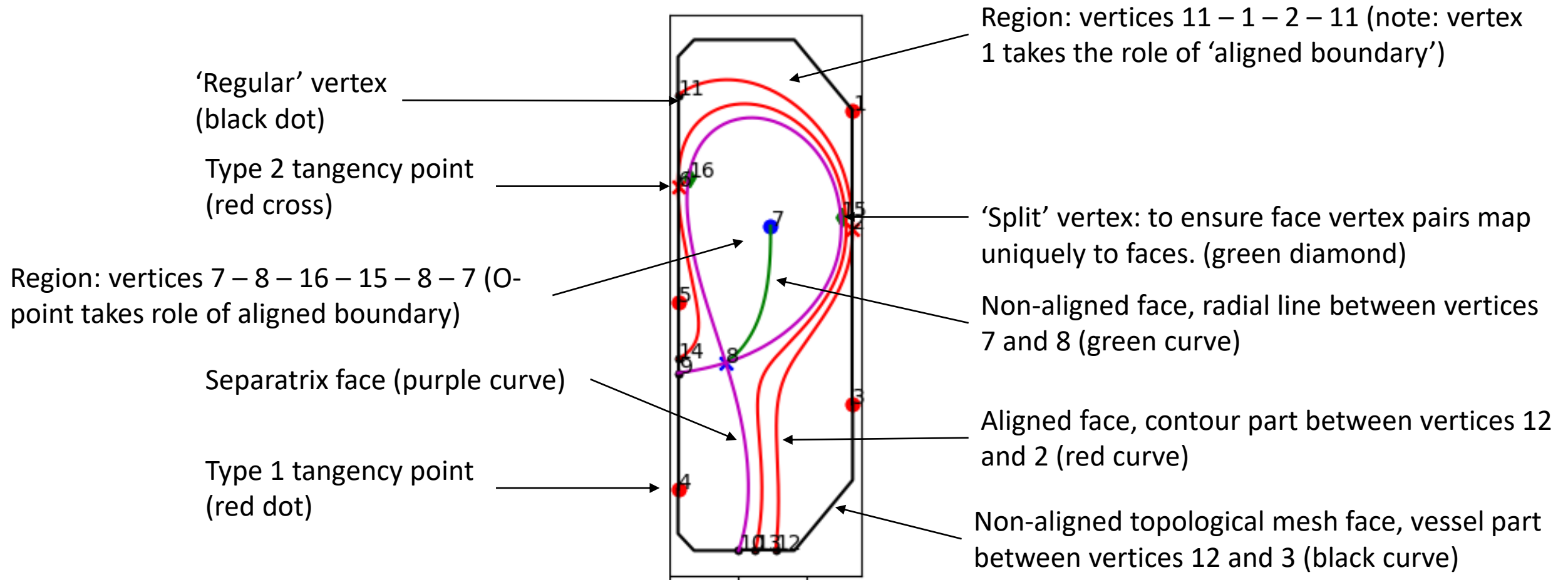
Base topomesh generation

- Inputs:
 - Magnetic field **Ψ values** on structured 2D grid
 - **Vessel polygon**: closed, nested, non-intersecting polygons
- **Identifies automatically**:
 - X-points, O-points, T(angency)-points and their contours ('topomesh vertices')
 - All related 'topomesh cells', 'topomesh tubes/regions' and 'topomesh faces'
- Note on nomenclature: we also speak of 'vertices' and 'faces' of the topological mesh, not to be confused with grid vertices and faces

Topomesh vertex & face classification

- Topomesh vertices:
 - X-points, O-points
 - Type 1 tangency points: flux surface *outside* domain
 - Type 2 tangency points: flux surface *inside* domain
- Topomesh faces:
 - Aligned: separatrices and other flux surfaces, aligned vessel parts (see later)
 - Non-aligned: vessel, 'radial' lines (lines between MF extrema)
- Regions: arbitrary classifications (not necessary for base topomesh construction, but useful for adaptations later)

Example: TCV



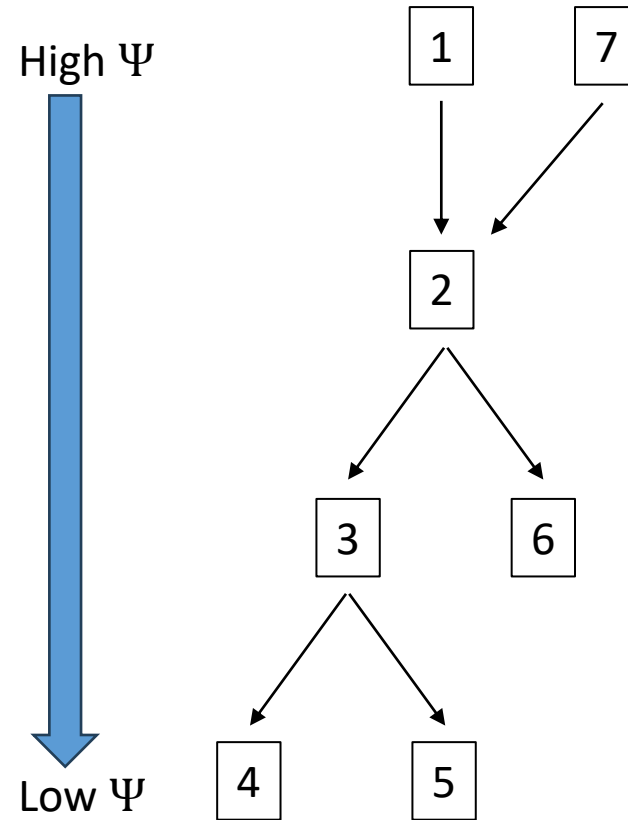
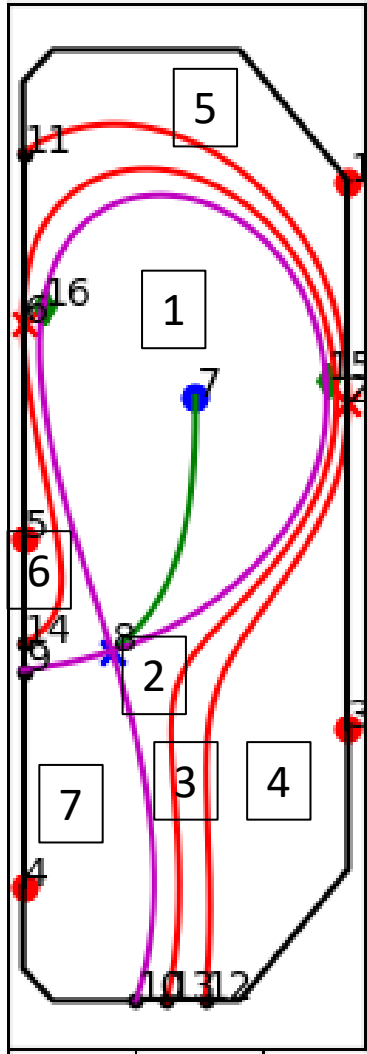
Intermezzo: topomesh from a graph POV

- Topomesh regions can be seen as graph nodes, interconnecting edges are topomesh faces (example will follow)
- This graph is the true and complete representation of interconnection
- As such, this generalizes the classical 'rectangular computational domain' representation

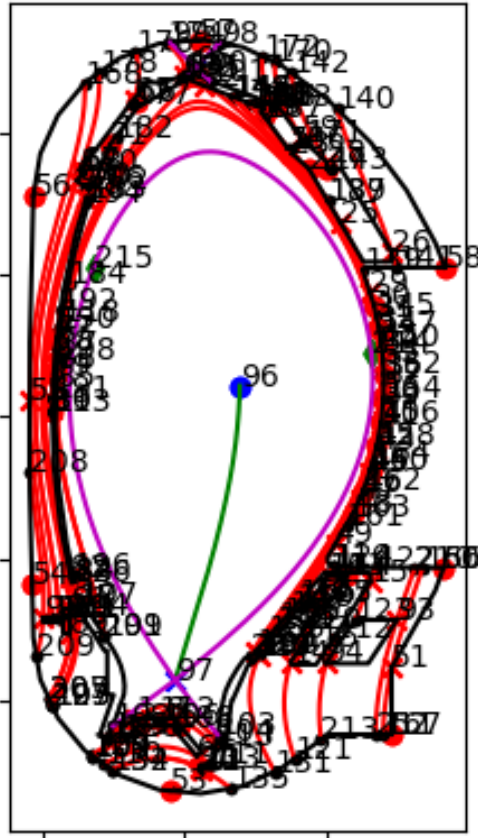
Intermezzo: topomesh from a graph POV

- It has some important properties
 - There is a sense of direction (gradient of Ψ)
 - It may contain cycles (e.g. nested vessel polygons), but if we use the direction we can make it a directed acyclic graph (DAG)
- This idea is exploited in many GOAT features:
 - Topomesh adaptations (especially region merging, see later)
 - Grid generation: propagation of vertex distribution
 - Identification of region types (e.g. 'core', 'wide grid/far SOL', ...) in conjunction with vertex/face classification

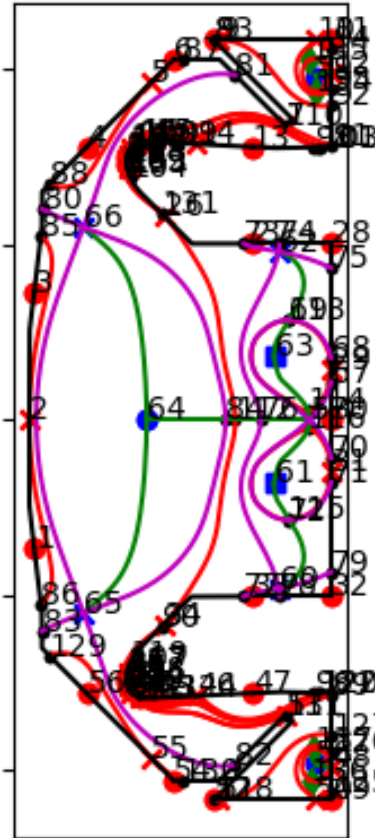
Example: TCV region graph



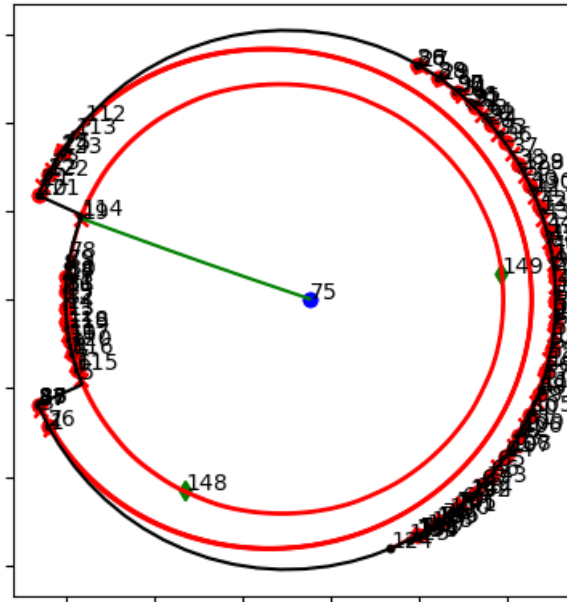
Other examples of base topomeshes



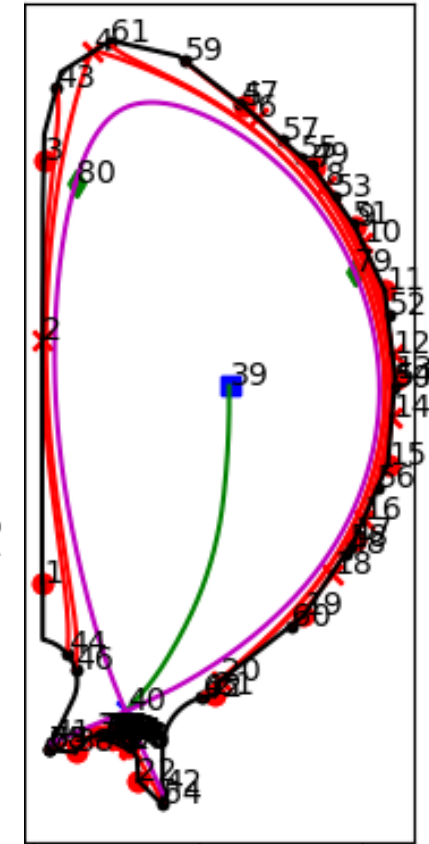
ASDEX with subdivertor structures



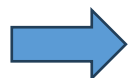
MAST



Aditya



ITER



Topomesh adaptations are required to get simulation-ready grids

Topomesh adaptations: overview

- Flux surface (contour) insertion (e.g. core, or as auxiliary for other methods)
- Region removal
- Region merging
- Aligned vessel part insertion

Flux surface insertion: core and PF

- Core (PF) flux surfaces can be introduced at some psi value between a min/max point (tangency point) and the X-point:

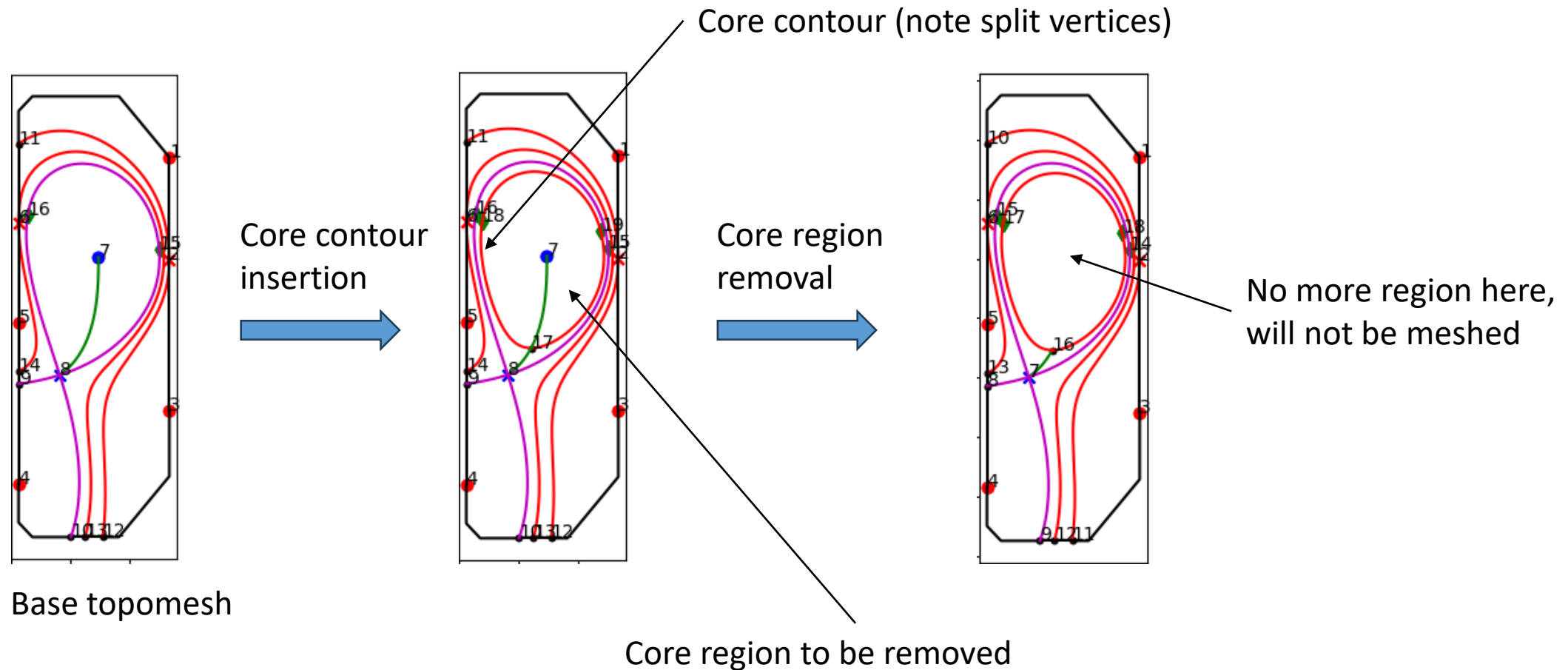
$$\begin{aligned}\Psi_C &= \lambda(\Psi_{ext} - \Psi_{xp}) + \Psi_{xp} \\ \Psi_{PF} &= \lambda(\Psi_{tp1} - \Psi_{xp}) + \Psi_{xp}\end{aligned}$$

- Note that PF region is only recognized as a region with a TP type 1 point that has two separatrix legs as boundary! Often not used

Region removal

- Core region: region with an O-point (so classical core region remains if core boundary is introduced)
- ‘Wide grid’ region: any region that doesn’t have a separatrix segment
- Region adjacent to structure: any region that has a radial boundary face that has (part of) a specified structure
- Can always be extended to remove arbitrary regions (requires further development)

Example: removing the core from TCV



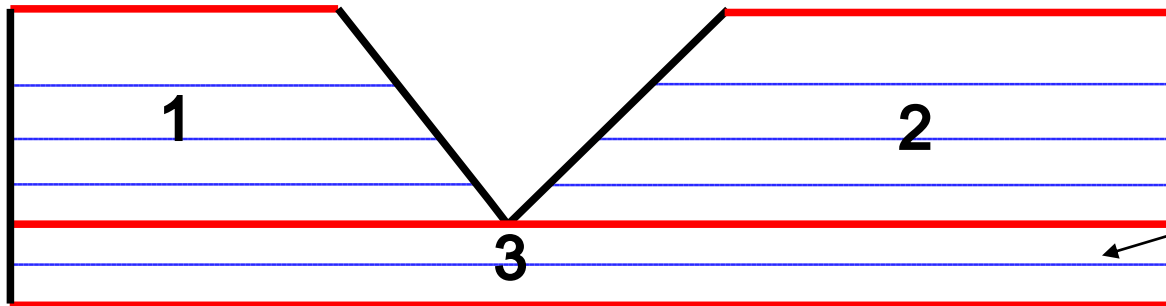
Region merging

- Idea: merge regions together if characteristic radial length is smaller than some value
- Should resolve:
 - Many small tubes close together, e.g. due to tangency points
 - Nearly connected X-points (e.g. DDN to CDN)
- But at a cost:
 - Final grid radial resolution is then locally bounded!
 - Some vessel parts may be treated as aligned, leading to ‘interesting’ features
 - Examples follow!

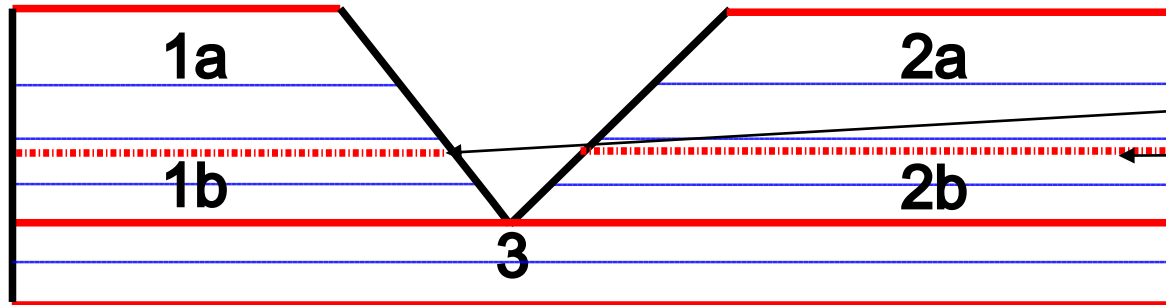
Region merging

- Algorithm:
 - Until all regions are sufficiently large, or no regions can be merged anymore, do the next steps
 - Find a set of neighbouring regions where there is at least one region 'too small'
 - If there are regions that aren't too small in this set, split them by inserting a contour – this minimizes resolution loss
 - Merge the resulting regions
- This algorithm does **not** allow merging over a separatrix, see later

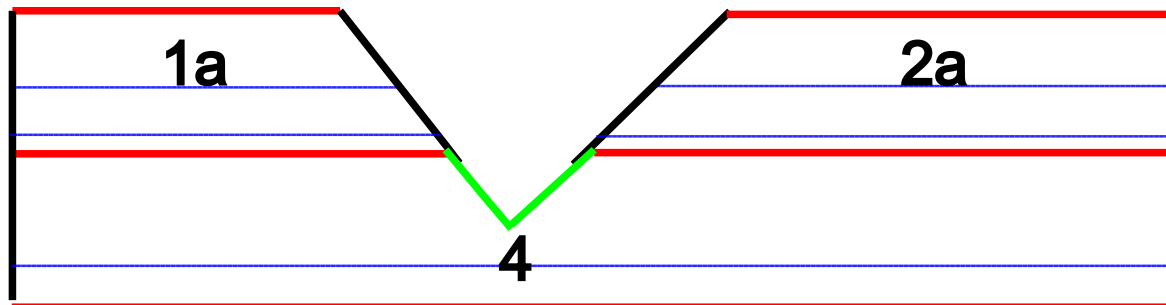
Example



Region 3 is deemed too small, but not regions 1 and 2



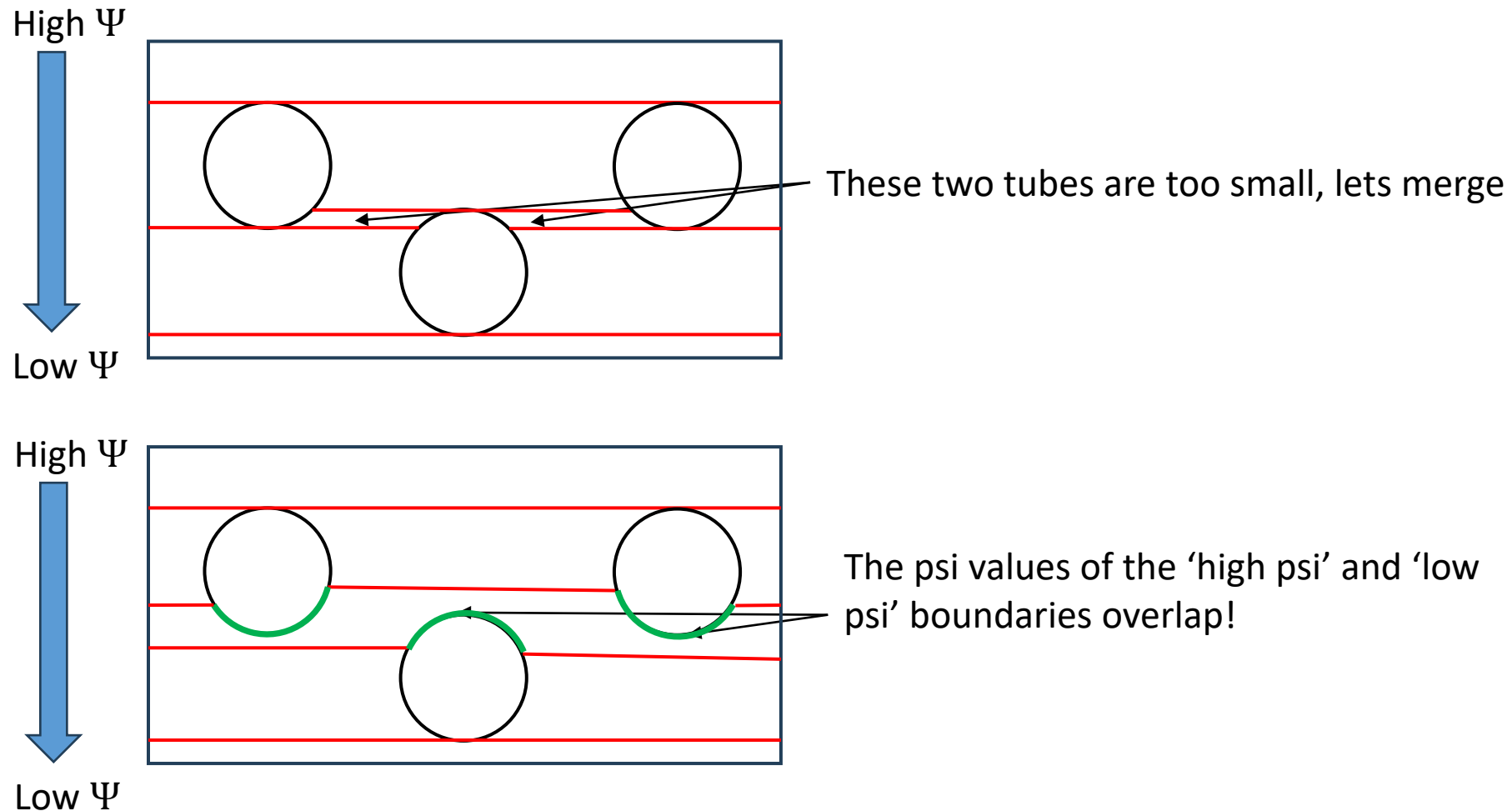
Split regions 1 and 2 by inserting new contours (red dashed lines) at minimal required distance from contour to be deleted



Merge regions 1b, 2b, 3 into region 4 by removing contour and inserting aligned vessel part (green)

Note: contours that intersect the aligned vessel will not be part of the grid!
→ shows resolution decrease

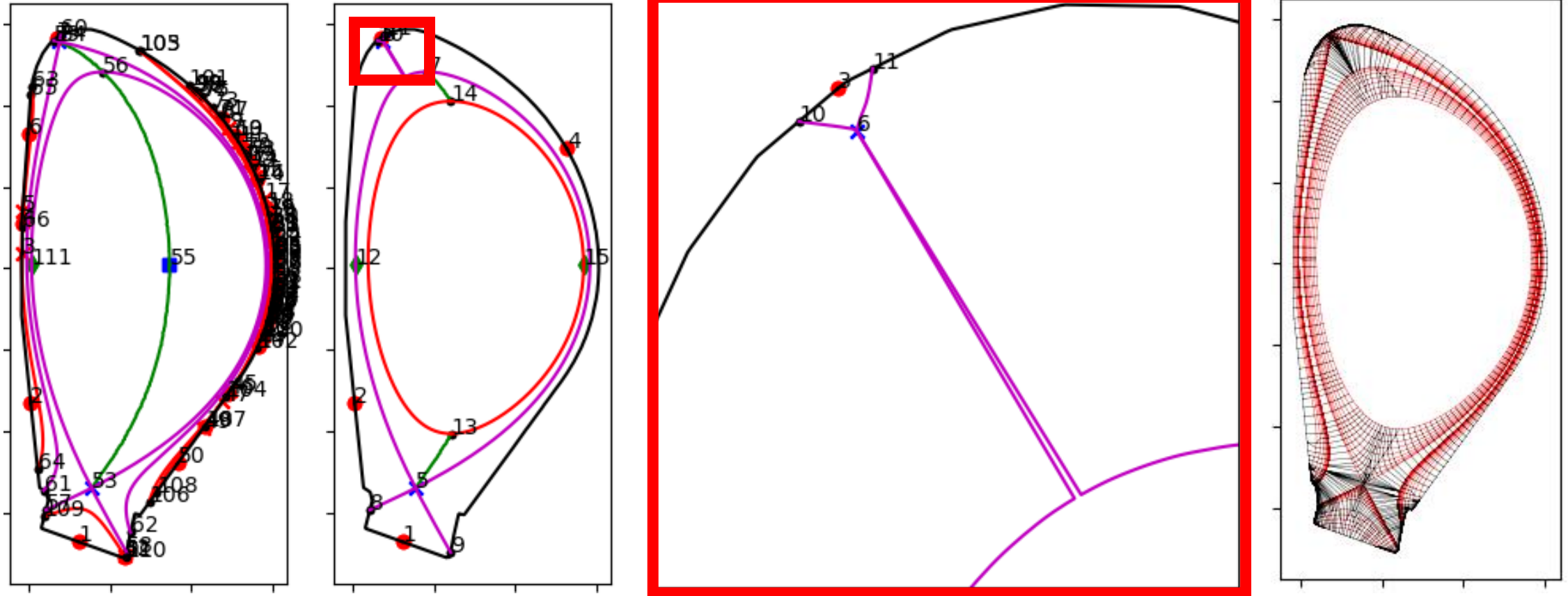
Example: plasma flow around cylinders



Region merging: dealing with separatrices

- Merging away a separatrix is tricky: no more ‘access’ to some regions (e.g. core in classical configuration) – we don’t (yet) allow this
- Hence, the only legal operation is ‘collapsing’ a tube with two separatrices into one separatrix with multiple and/or high-order X-points
- Note that this works *regardless of the actual radial distance between separatrices*

Example: DEMO



→ Bad topomesh, bad grid

→ Tube merging limits radial resolution

Region merging: limitations & caveats

- Merging away internal structures is not supported (nor desirable?)
- Very small features with O- and X-points cannot be merged away (possible to extend algorithm, for now: coarsen magnetic field...)
- Psi values are not constant anymore on an aligned boundary
- The topomesh graph representation still stands, regardless of psi values

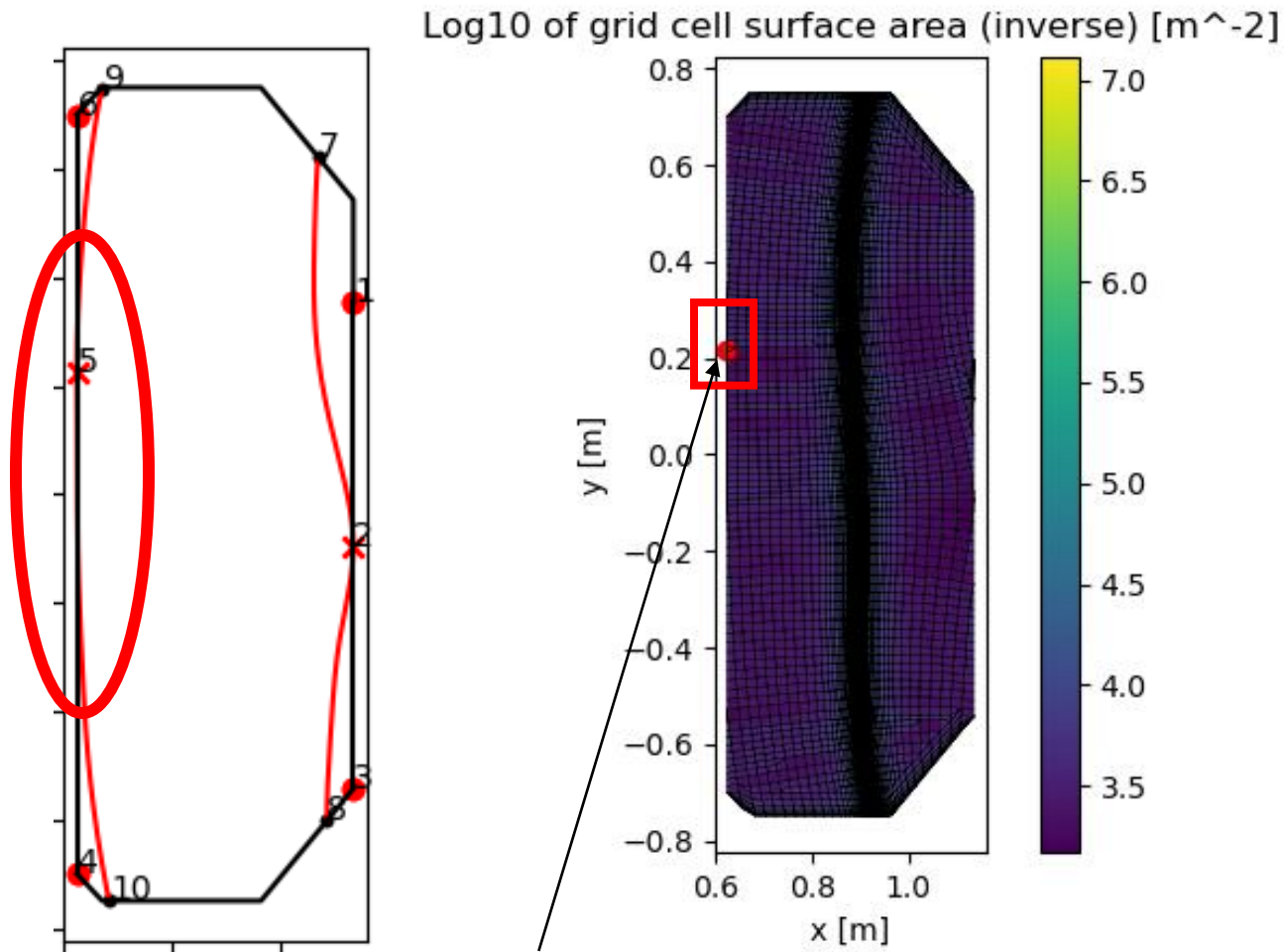
Aligned vessel part insertion @type 2 T-points

- Motivation:
 - Often, large part of the vessel is almost aligned with the magnetic field at type 2 T-points
 - This may result in very small cells/very 'flat' triangles locally, resulting in locally bad grid quality
- Therefore, need to change topological mesh: 'replace' type 2 T-point by aligned vessel part

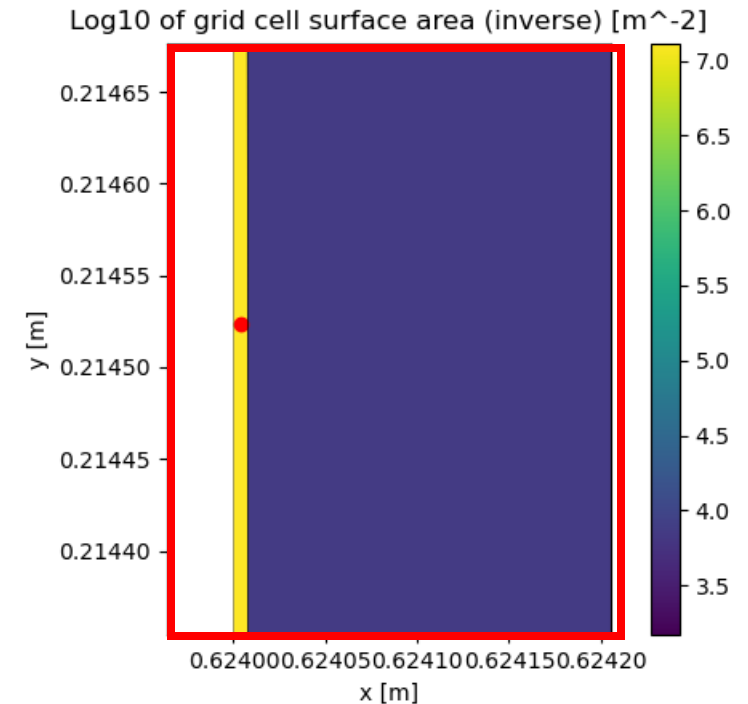
Algorithm

- If the local angle at a type 2 T-point is too small, try to split the neighbouring tubes
- If this is successful, merge the resulting tube set over the original tangency point contour, introducing an aligned vessel part at the original T-point location
- Note: this does *not* imply any change of the geometry or local magnetic field!

Example: TCV ECWC case

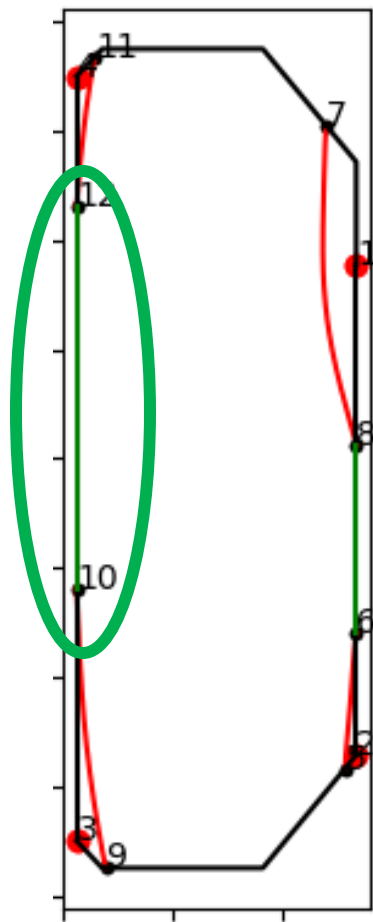


Red dot indicates smallest cell

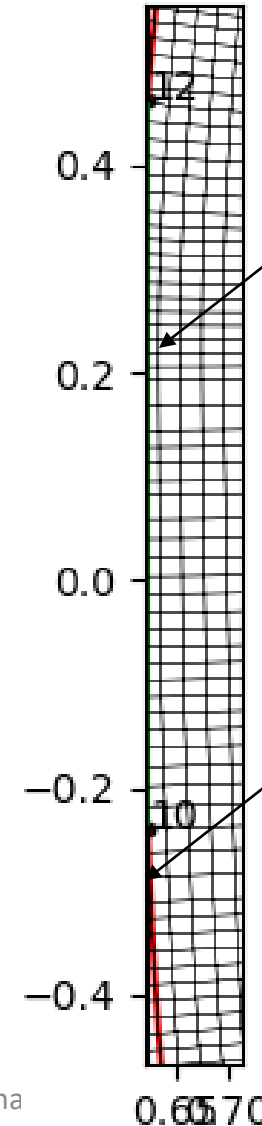
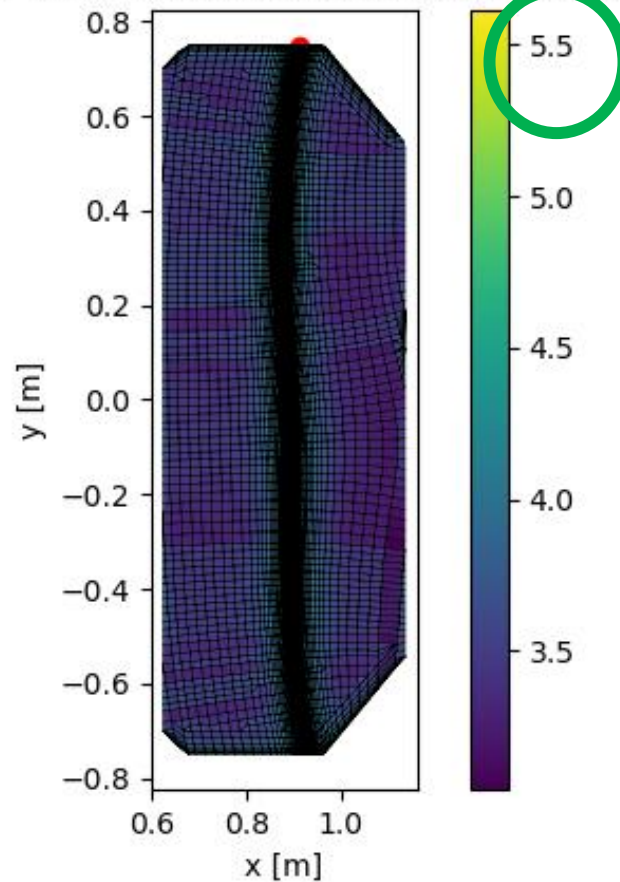


There's actually cells there!
Expected: there is a region that runs from vertices 4 – 5 – 10 – 4

Example: TCV ECWC case with AVP



Log10 of grid cell surface area (inverse) [m⁻²]



Due to AVP: only cut cells near topomesh face 10 – 12

Only triangular cells again near vertex 10, but sufficiently large

Limitations and caveats

- Contour insertion to split a tube may not be possible (e.g. tubes with overlapping psi values...)
- Arbitrary large angles are not possible (good values are $0 - 10^\circ$)
- 'Cut cells' will occur there

Key takeaways: topomesh generation

- Base topomesh doesn't require (much) user input
- **Topomesh adaptations are necessary** to enable simulation-ready grids:
 - Core region removal
 - Region merging
 - Aligned vessel part insertion
- Important rules of thumb:
 - **Bad topomesh → bad grid**
 - **Region merging and AVP limits local radial resolution**

Grid generation: easy! ...or not?

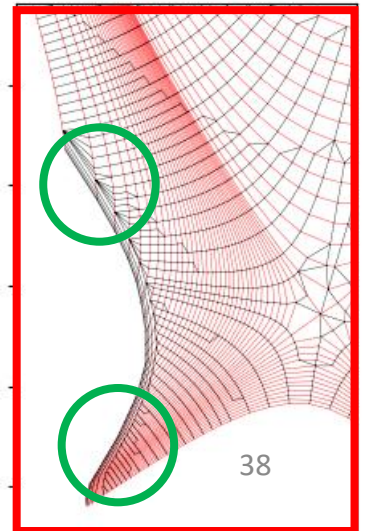
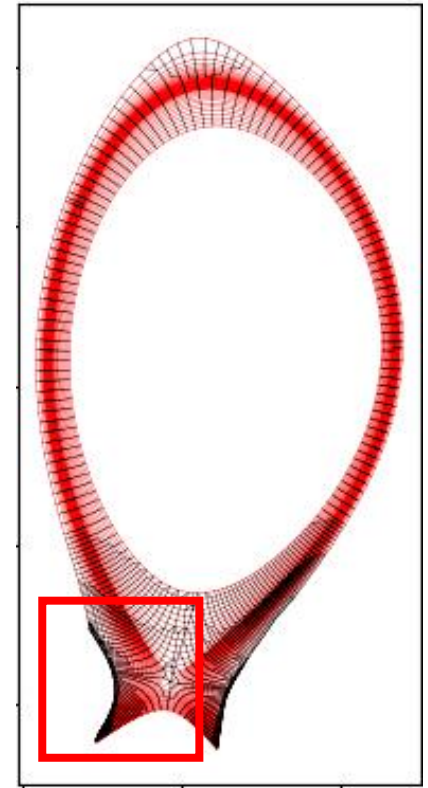
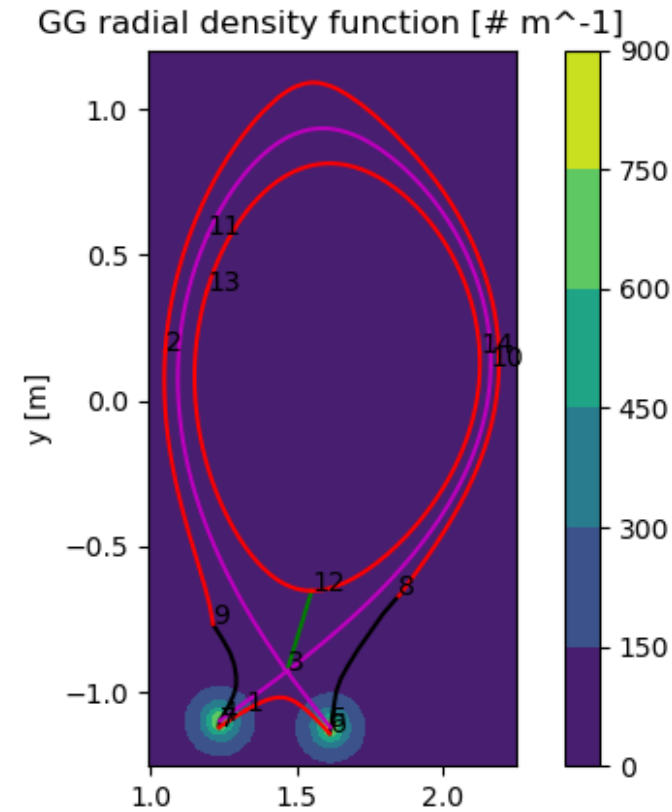
- Though topologically, everything is now rectangular, it certainly isn't in real space
- Discretization effects may lead to intersecting faces
- How to ensure smooth transitions between topomesh regions?
- Multiple options possible, here: currently most performant one

Grid generation: main algorithm

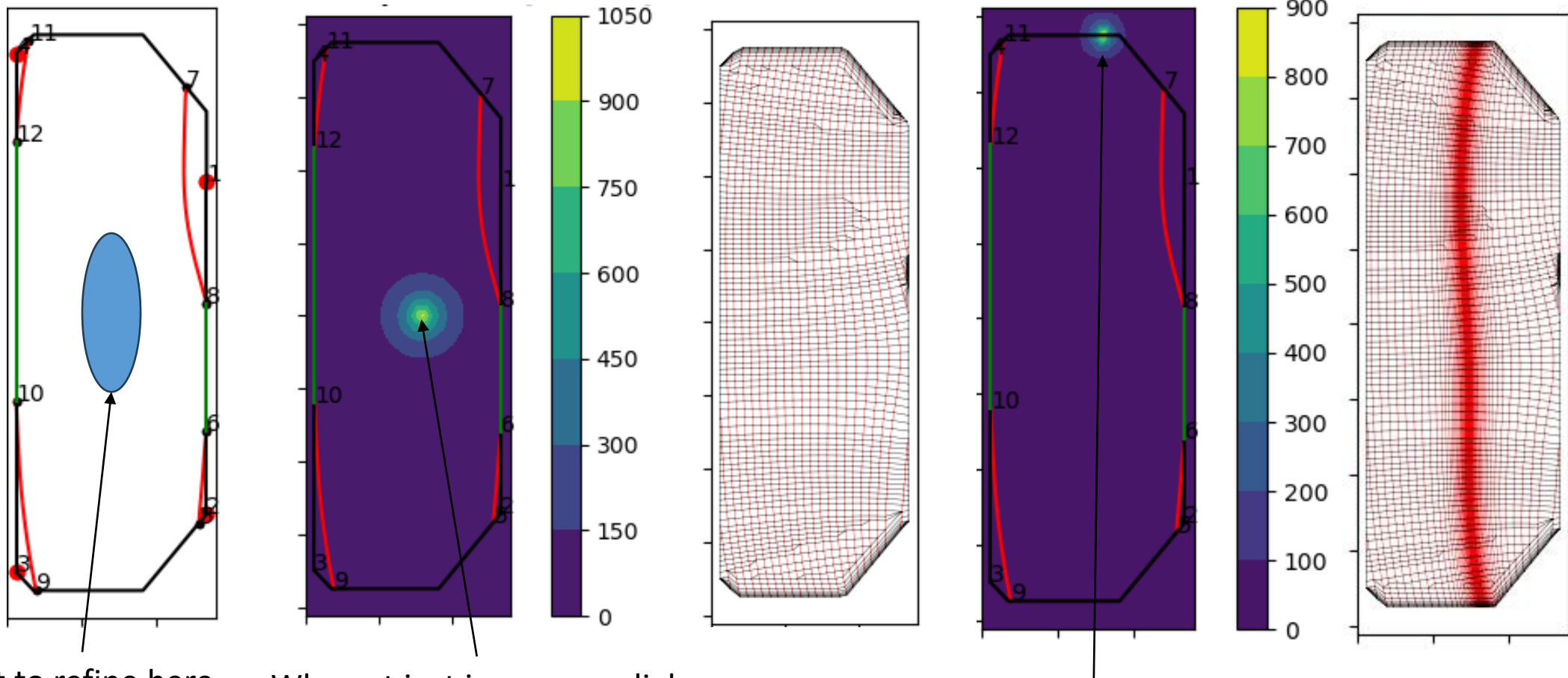
- First, we need to determine the **radial resolution** (we cannot locally refine that one yet) → to trace flux surfaces per region
- Then, we construct **flux tubes** per pair of flux surfaces for each region
- If needed, **extend flux tubes** with vessel parts
- Afterwards, we **distribute vertices on flux surfaces per flux tube**, propagated by tracing radial lines (cf. CARRE)
- Finally, we construct **grid cells and faces**

Radial resolution

- Determined by distribution function [# lines per meter radial length]
- Important: distribution function is only evaluated at non-aligned surfaces of a region, so matters only there!



Example: TCV ECWC case



We want to refine here
since plasma heating is
applied there through EC

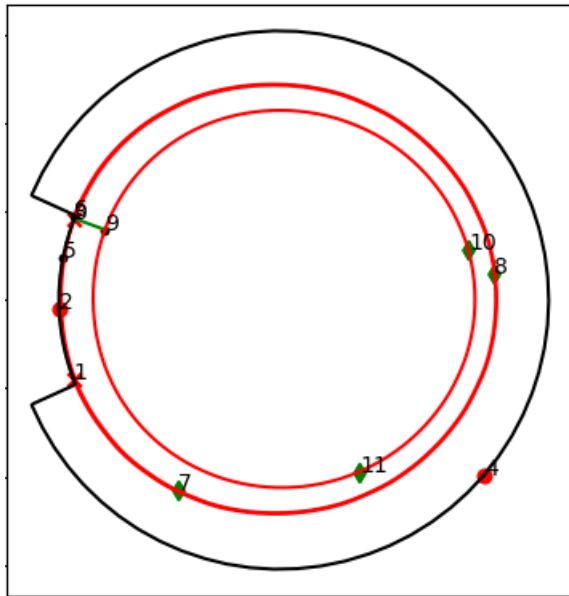
Why not just increase radial
distribution function here?

Nothing happened! Refine at non-aligned face

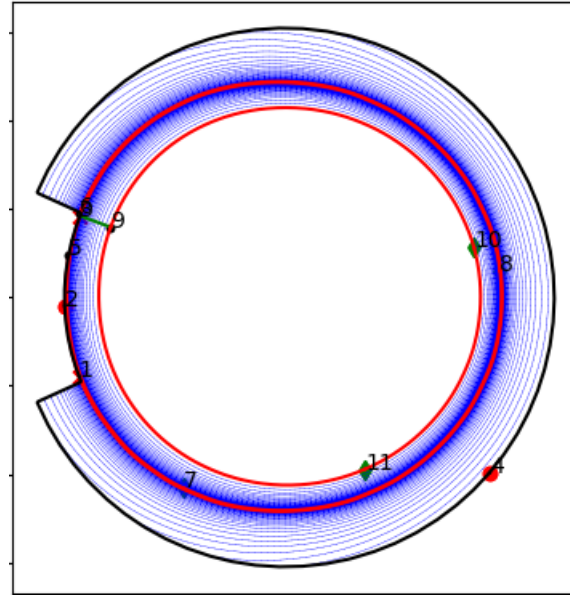
Final notes on radial resolution

- Note that in general no smooth transitions can be expected between topomesh regions in radial resolution
- Local radial refinement would resolve several issues & seemingly unexpected behavior (cfr. Previous example)

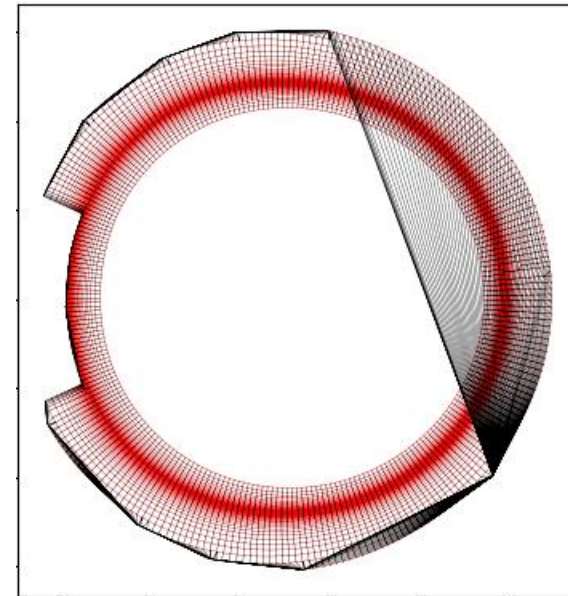
Flux tube construction & extension: why?



Aditya topomesh after adaptations, looking good



Traced flux surfaces for gridding, looking good

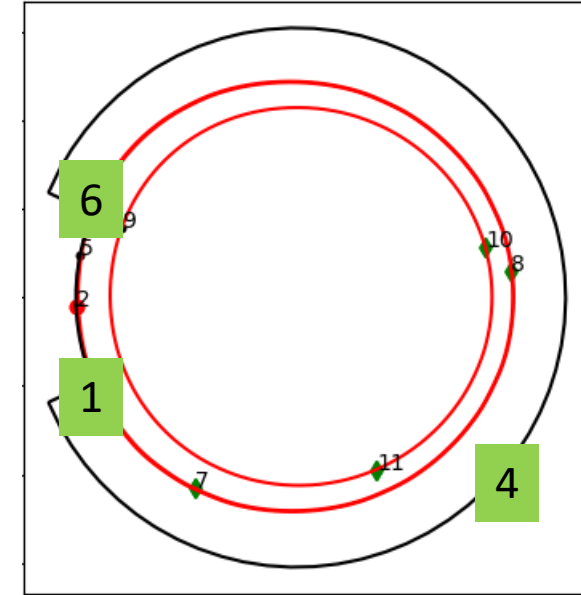
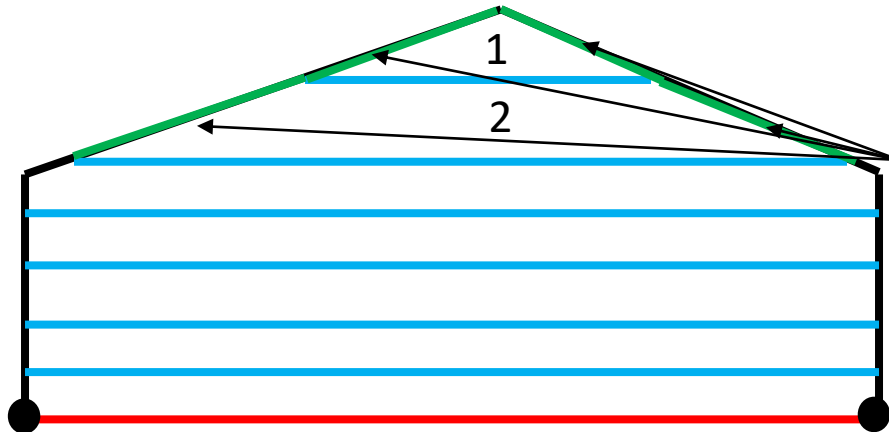
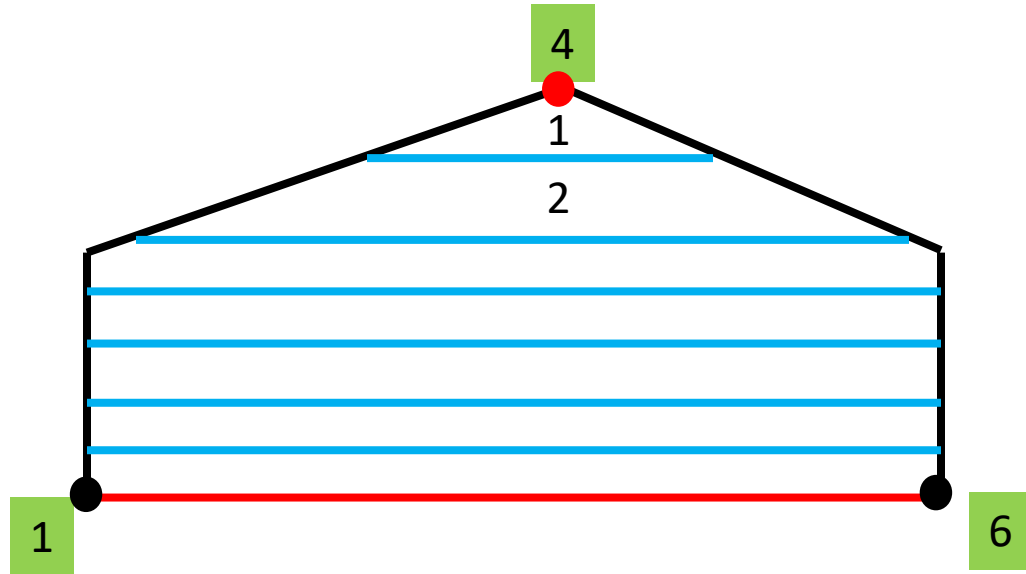


The actual grid without tube extension – what?!



Reason: combined effect of nearly aligned vessel, coarse radial resolution, high poloidal resolution, and vessel & magnetic field curvature

Flux tube construction & extension: how?

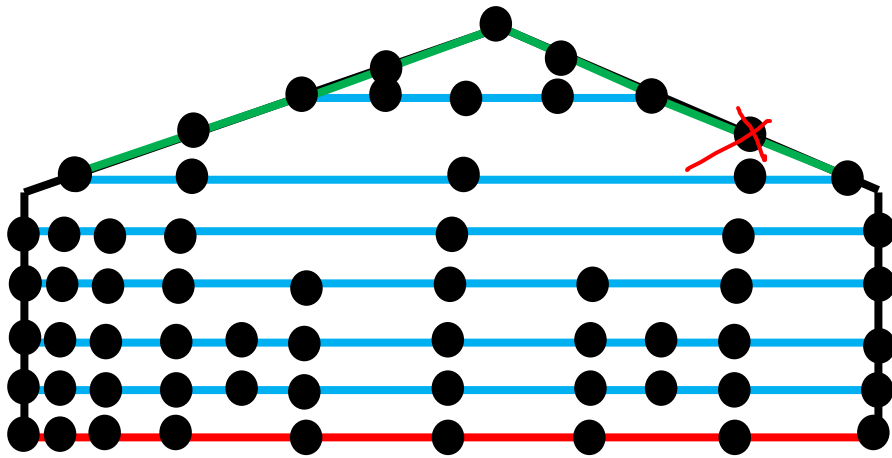


Cfr. Aligned vessel parts: 'extend' the high psi/ low psi surface of a tube with a vessel part, that is treated as 'aligned'

→ Vertices can be distributed on vessel parts now

→ **Cut-cell like geometries**

Vertex distribution



Start: bottom tube (high ψ to low ψ) with initial (coarse) vertex distribution

Step 1: refine/coarsen high Ψ surface locally & apply BL

Step 2: trace radial lines and find intersections with low Ψ boundary
→ these become vertices on low Ψ boundary

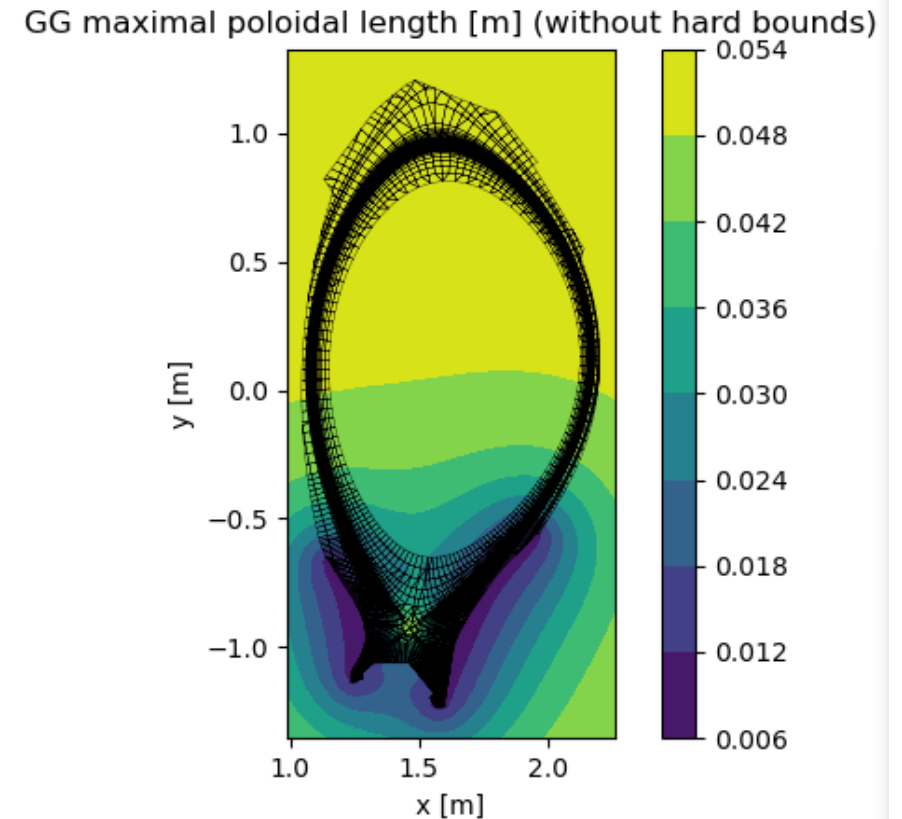
Repeat steps 1 and 2 for all next tubes

Note that the tube extensions allow vertices on the original vessel boundaries! Otherwise, no vertices would be present there

Note that boundary layers are not applied at tube extensions!
Would lead to small triangular cut cells

How to manipulate the poloidal resolution?

- Initial distribution (cfr. radial resolution) will be overridden by orthogonal line tracing
- **Use refiners for local refinement:** specify L_{min} , L_{max} and boundary layers
- L_{min}/L_{max} are again distribution functions



How to construct grid faces?

- Aligned faces are trivial, but non-aligned faces not
- Current criteria (but extension possible!):
 - **Non-intersecting faces/cells** if possible → achieved by building vertex visibility graph that will indicate which faces are feasible (if none, warnings occur)
 - **Prioritize orthogonality** – will lead to mostly orthogonal faces due to orthogonal line tracing

Caveats for poloidal vertex distribution

- Flux tube extension: currently, criterium to extend is fully based on vessel segment length, which is not always easy to work with (threshold is a constant – see also tutorial)
- Boundary layers are not imposed on extended segments: would lead to many small triangles
- L_{min}/L_{max} only specify *bounds* → actual refinement may vary
- No cell quality metrics accounted for during face/cell construction

Outline

- Introduction
- Grid generator mechanics
- Workflow: from DivGeo to SOLPS simulation with kinetic neutrals & voids
- Conclusion & outlook

Prerequisites

- Development currently in experimental git branches, based on 3.2.1:
 - SOLPS: feature/svdk/wg-workflow
 - DivGeo: feature/svdk/wg-workflow
 - Eirene: feature/svdk/goat-integration
 - B25: goat-integration-clean-rebased
 - Goat: feature/goat-uinp-integration
- See also hands-on session this afternoon

Before...

- Manual adjustment of boundary conditions required
- fcReg hell: only defined for certain topologies (and only useful for post-processing – solver itself doesn't care)
- Voids & Eirene hard to support: triang/Uinp route completely based on structured grid format & carre setup

Idea

- Workflow: DivGeo → GOAT → triang/Uinp → SOLPS-ITER
- In Uinp, developed goat route to determine input files + adjusted triang script to identify this route (the user can press 'enter' and files will come out) – of course to be checked by the user for correctness!
- Other workflow as before: triang for triangulation of voids, stitching of voids & plasma grid for Eirene by b2us, ...
- Disclaimer: still issues possible, experimental!

Dealing with void regions

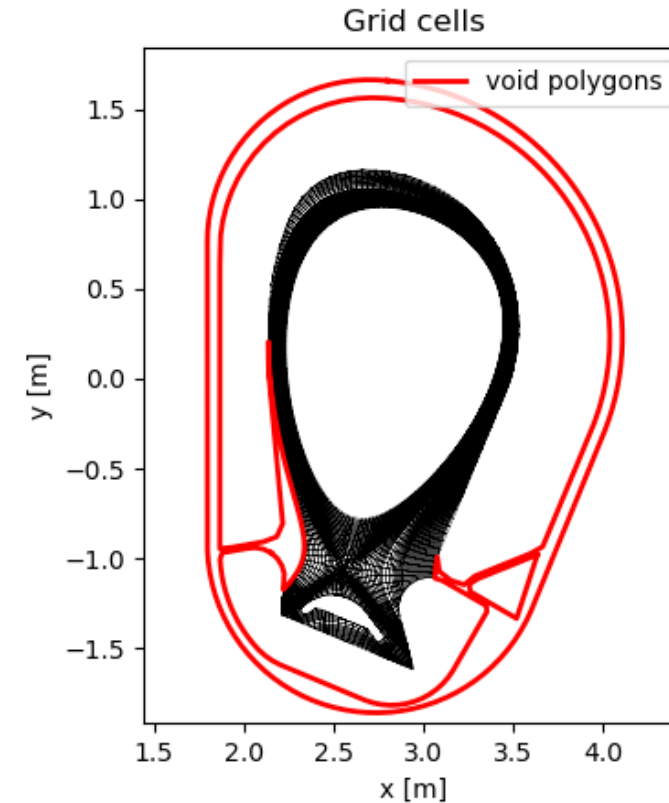
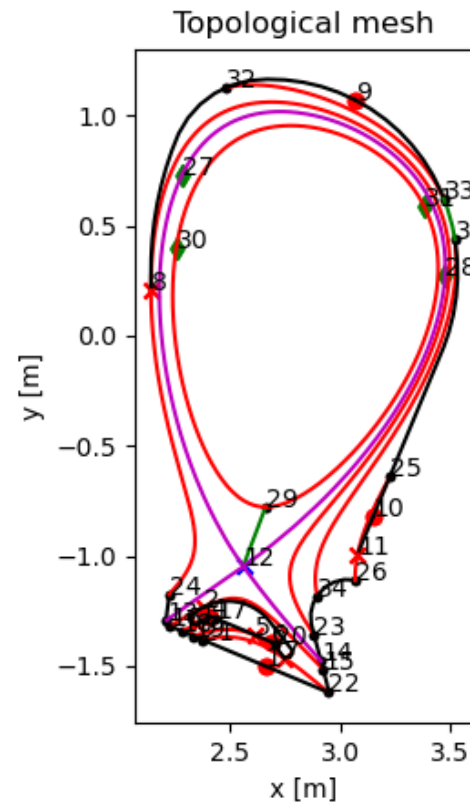
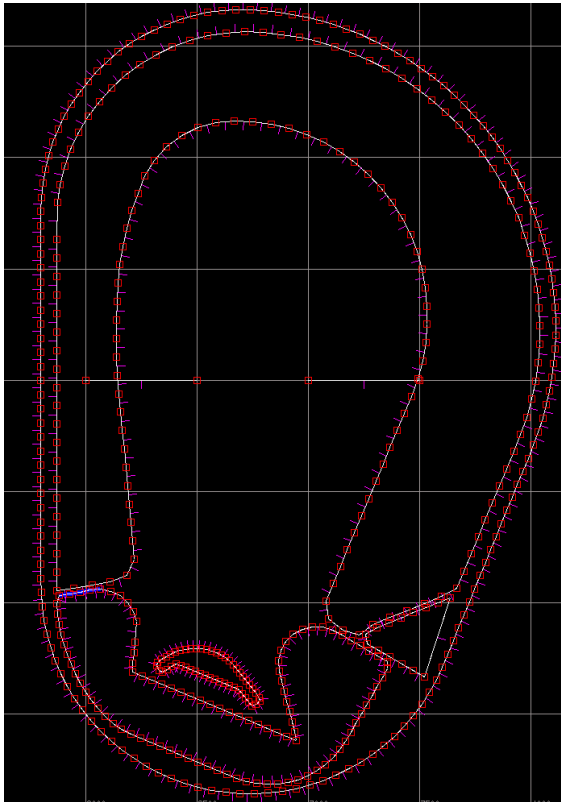
- **Goat identifies the void polygons** and writes to fort.78 and fort_goat.78 file (the former for tria, the latter for Uinp)
- Goat can also remove self-intersections from the void polygon by moving vertices (to some extent)
- Requires input via DivGeo – need to map void polygon edges back to elements (hence fort_goat.78 file)
- **Natural support of additional void regions** etc, no tricks needed

DivGeo setup

- Very similar to Carre2 workflow (see also talk Wouter Dekeyser)
- Only need to specify that you are using goat grid generator + in output mode: uncheck all checks to be done (many are irrelevant)
- Support of additional void regions by face label convention:
 - Negative face labels to indicate void interface
 - Positive face labels to indicate vessel structure
- Detailed instructions for technically working case: see hands-on session

Goat

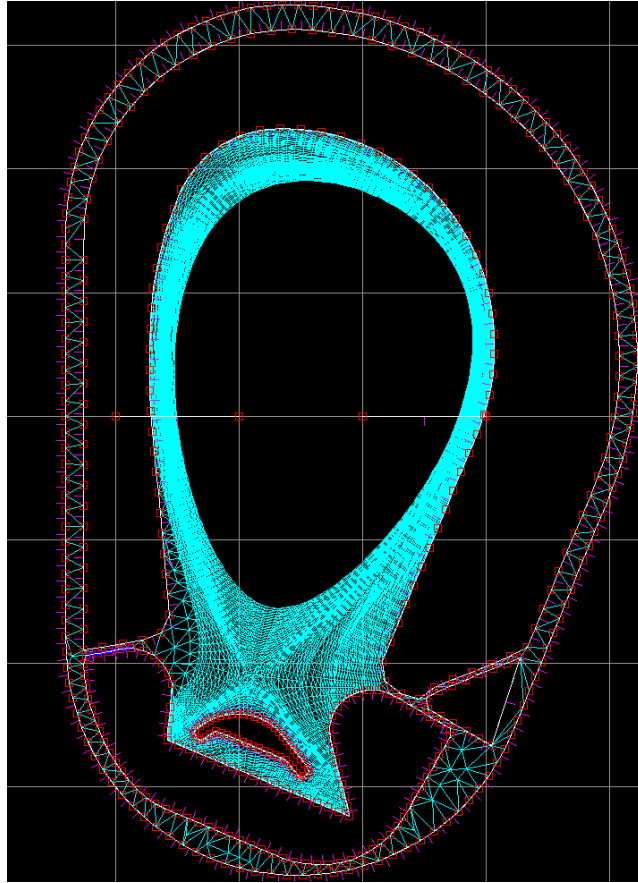
- Just generate the grid, fort.78 and fort_goat.78 follow automatically



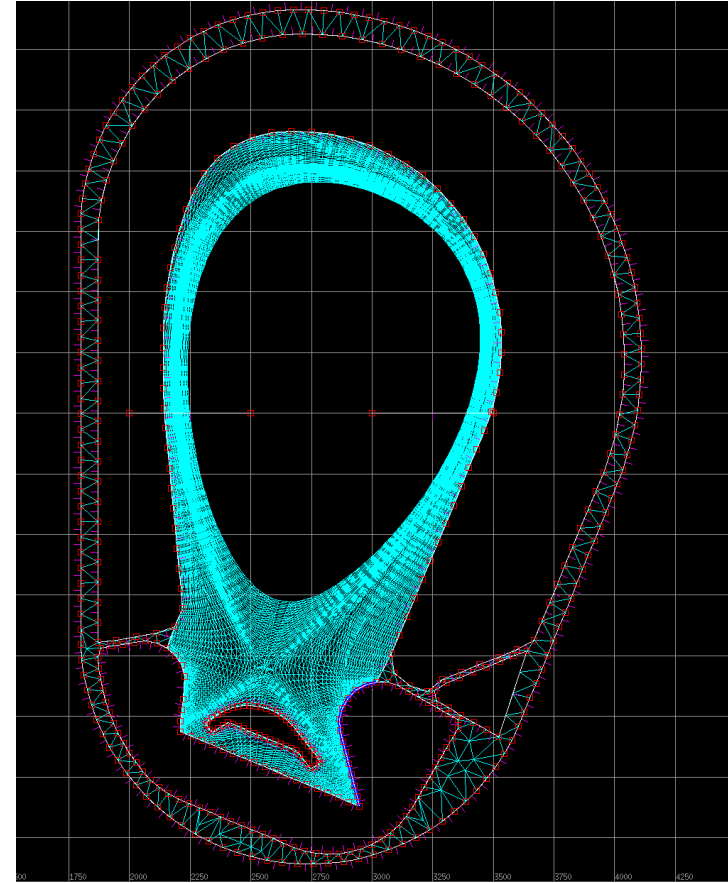
Triang

- Make sure grid files are copied in your working directory and use standard names (otherwise you may have to specify):
 - 'traduit.out.b2us' for the grid file
 - 'fort_goat.78' for the void polygon file for Uinp
 - 'fort.78' for void polygon for tria
- Run triang, press enter *a lot* → ready for coupled run, visualize templates as usual

Examples: see hands-on session



Voids through goat



Voids through DivGeo

Final notes on SOLPS-ITER

- Post-processing output for general topologies not yet supported in B2.5 (requires further discussion)
- All topological information readily available in goat output and provided through b2fgmtry
- Block 3B surfaces don't coincide with DivGeo elements anymore
- Due to precision effects, additional fort.36 file to increase precision of surfaces specified in block 3B of input.dat (automatically generated by Uinp if following goat route)

Outline

- Introduction
- Grid generator mechanics
- Workflow: from DivGeo to SOLPS simulation with kinetic neutrals & voids
- Conclusion & outlook

Conclusion

- Grid generator capable of generating wide and narrow grids for arbitrary topologies
- Interfacing with SOLPS-ITER, including void support, is progressing
- Time now for community to use grids and suggest new gridding features if needed

Outlook

- Support for local radial refinement (splitting of flux tubes) in solver would alleviate many issues:
 - Flux expansion effects near X-points
 - Cut cell approach at nearly aligned wall segments
 - Resolution of local phenomena (e.g. ECWC heating)
 - ...
- Grid adaptation/deformation may further improve grid quality

Back-up slides

Base topomesh: theory

Extrema tracing: basic idea

- An extremum has conditions $\frac{\partial \Psi}{\partial x} = 0, \frac{\partial \Psi}{\partial y} = 0$ ($\nabla \Psi = 0$)
- Extremum type (max, min, saddle) is determined by determinant of $\nabla^2 \Psi$:
- A Newton solver of the KKT conditions will only converge to a single local optimum and will likely miss saddle points
- Because the dimensionality is low (2D, max 3D), we can simply evaluate $\nabla \Psi$, trace contours of $\frac{\partial \Psi}{\partial x} = 0, \frac{\partial \Psi}{\partial y} = 0$, and then find extrema as intersections of these contours
- At intersections, we can do a Newton solve to further refine, but often not necessary. Evaluating Hessian locally gives then type of optimum
- Note that derivative evaluation requires at least $C = 2$ continuity of magnetic field description

Extrema tracing

- To ensure we can trace field lines from and to extrema (and other special points), we insert a boundary layer in R- and Z-direction at traced extrema (in a smart way, because this is actually non-trivial if you don't want overlapping saddle point domains)
- Flux surfaces and contours of $\frac{\partial \Psi}{\partial x} = 0, \frac{\partial \Psi}{\partial y} = 0$ are then added to the topomesh
- Do note that for linear fields, these derivatives may be zero everywhere, which will lead to interesting results – best to add small magnetic field component to avoid this

Tangency point tracing

- Tangency points can be done continuously or discretely, but the discrete option is the most robust/best (does require vessel refinement, see vessel options)
- Tangency point conditions: $V(x, y) = 0$ (i.e. vessel contour), $\frac{\partial \Psi}{\partial x} \frac{\partial V}{\partial x} + \frac{\partial \Psi}{\partial y} \frac{\partial V}{\partial y} = 0$ (i.e. vessel normal is perpendicular to magnetic field tangent)
- Distinction between type 1 and type 2 tangency points:
 - Type 1: contour is in exterior of domain
 - Type 2: contour is in interior of domain
 - Determined based on how field values vary along vessel & vessel outward pointing normal (or in continuous way: FORMULAS HERE)

Tangency point flux surfaces

- Quite tricky to trace and insert (may intersect multiple times with vessel boundaries due to discretization effects)
- Traced from tangency point to first 'real' intersection with vessel – otherwise many unnecessary regions pop up and some assumptions are broken

Face, cell and tube construction

- Normally, face and cell construction should never be an issue
- Tube construction can go wrong if for some reason cells exist with more than 2 radial faces – warnings will be thrown in topomesh creation, errors in grid generation
- After this step, the basic topological mesh is determined, which can be modified in next steps

