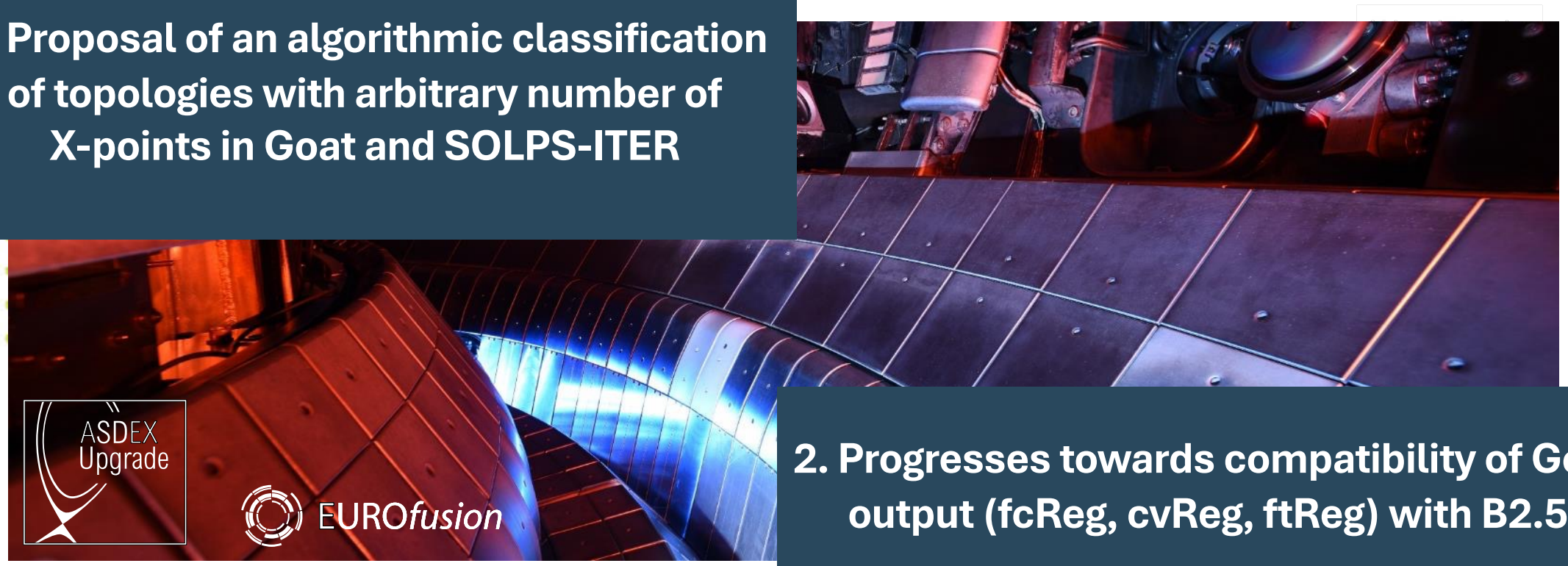
MAX-PLANCK-INSTITUT
FÜR PLASMAPHYSIK

1. Proposal of an algorithmic classification of topologies with arbitrary number of X-points in Goat and SOLPS-ITER



2. Progresses towards compatibility of Goat output (fcReg, cvReg, ftReg) with B2.5

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Problem:

- SOLPS-ITER WG can simulate arbitrary magnetic topologies

Problem:

- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies

Problem:

- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies
- … but this capability is difficult to exploit as SOLPS was only taught to expect a finite (and small) set of topologies

Problem:

- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies
- … but this capability is difficult to exploit as SOLPS was only taught to expect a finite (and small) set of topologies
  For the rest, good simulation with no or wrong output diagnostics (best case), or run which fails to start (worst case)

Problem:

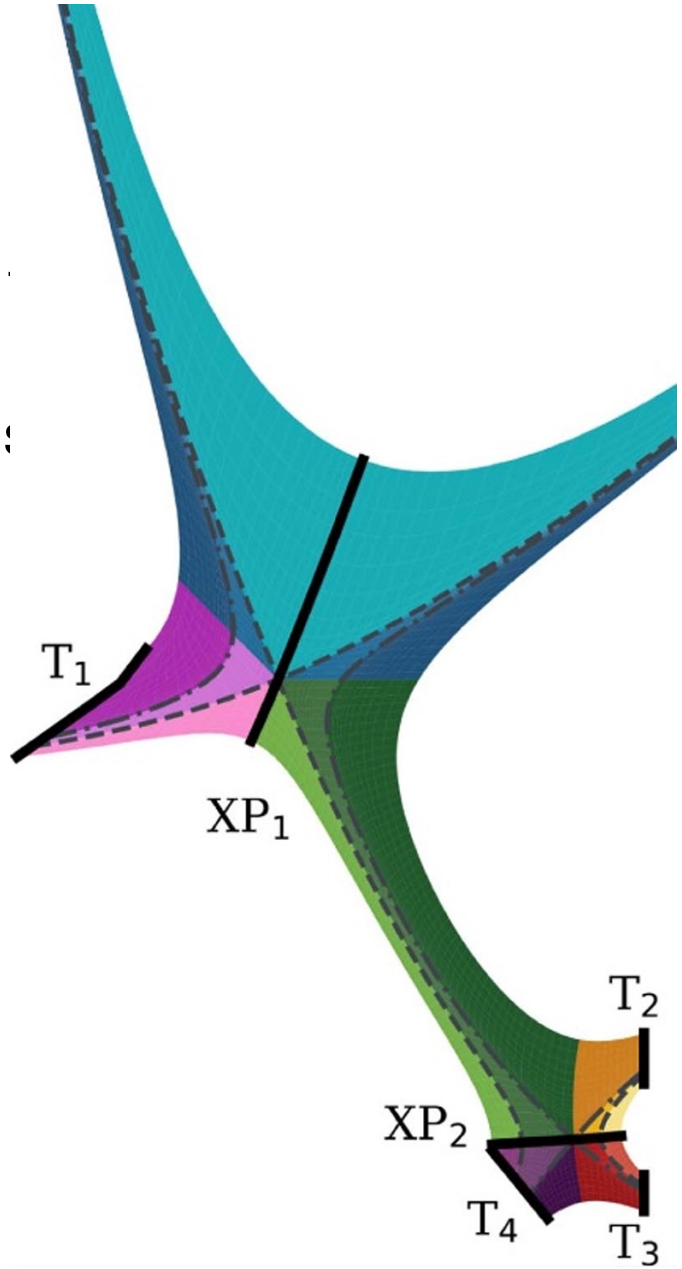
- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies
- … but this capability is difficult to exploit as SOLPS was only taught to expect a finite (and small) set of topologies
  For the rest, good simulation with no or wrong output diagnostics (best case), or run which fails to start (worst case)

- Catalogue and classify all topologies and teach SOLPS-ITER how to handle them is impossible (infinite possibilities)…

Problem:

- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies
- … but this capability is difficult to exploit as SOLPS was only
  a finite (and small) set of topologies
  For the rest, good simulation with no or wrong output diagno
  or run which fails to start (worst case)

- Catalogue and classify all topologies and teach SOLPS-ITER
  is impossible (infinite possibilities)…



Problem:

- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies
- … but this capability is difficult to exploit as SOLPS was only taught to expect a finite (and small) set of topologies
  For the rest, good simulation with no or wrong output diagnostics (best case), or run which fails to start (worst case)

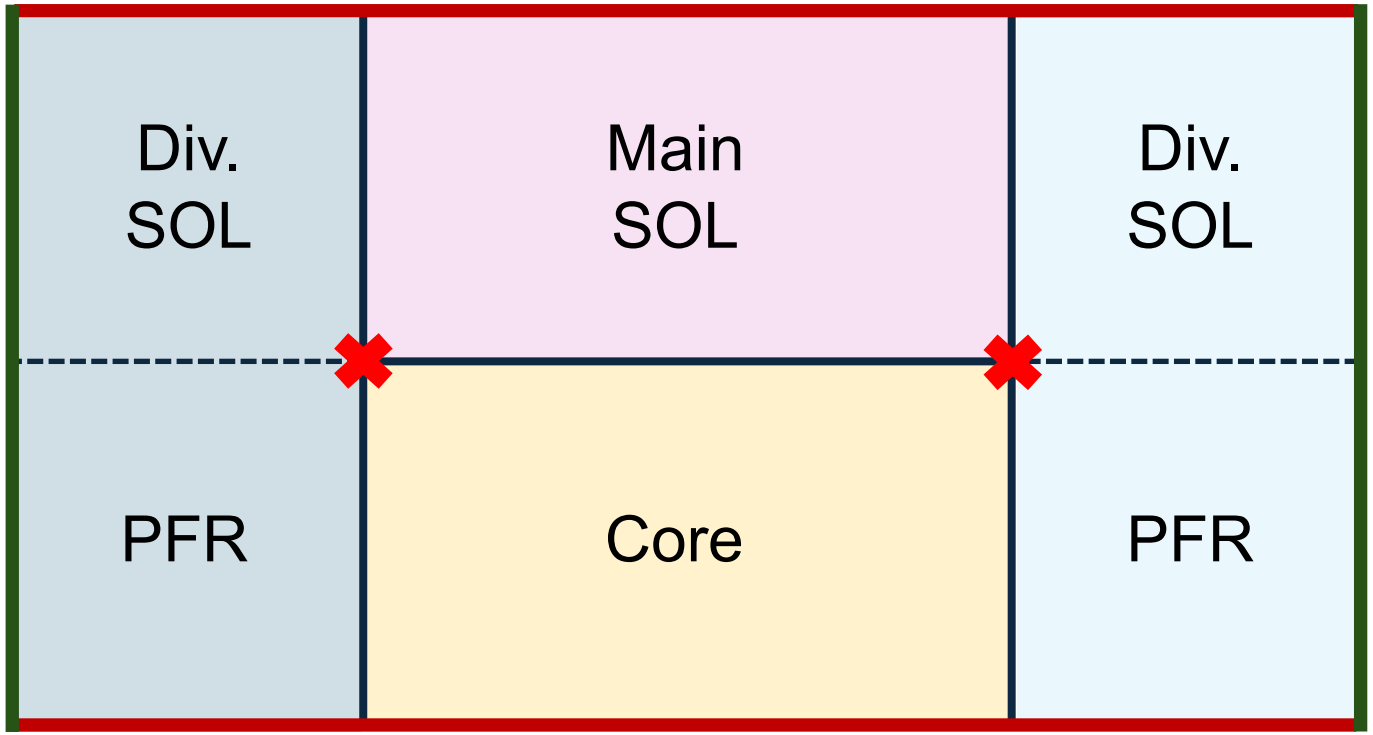
- Catalogue and classify all topologies and teach SOLPS-ITER how to handle them is impossible (infinite possibilities)…
- … but algorithmic classification IS possible if we employ a simplifying (yet relevant) assumption:
  Magnetic topology with only one O-point, with X-point(s) being only far above or far below the O-point

Problem:

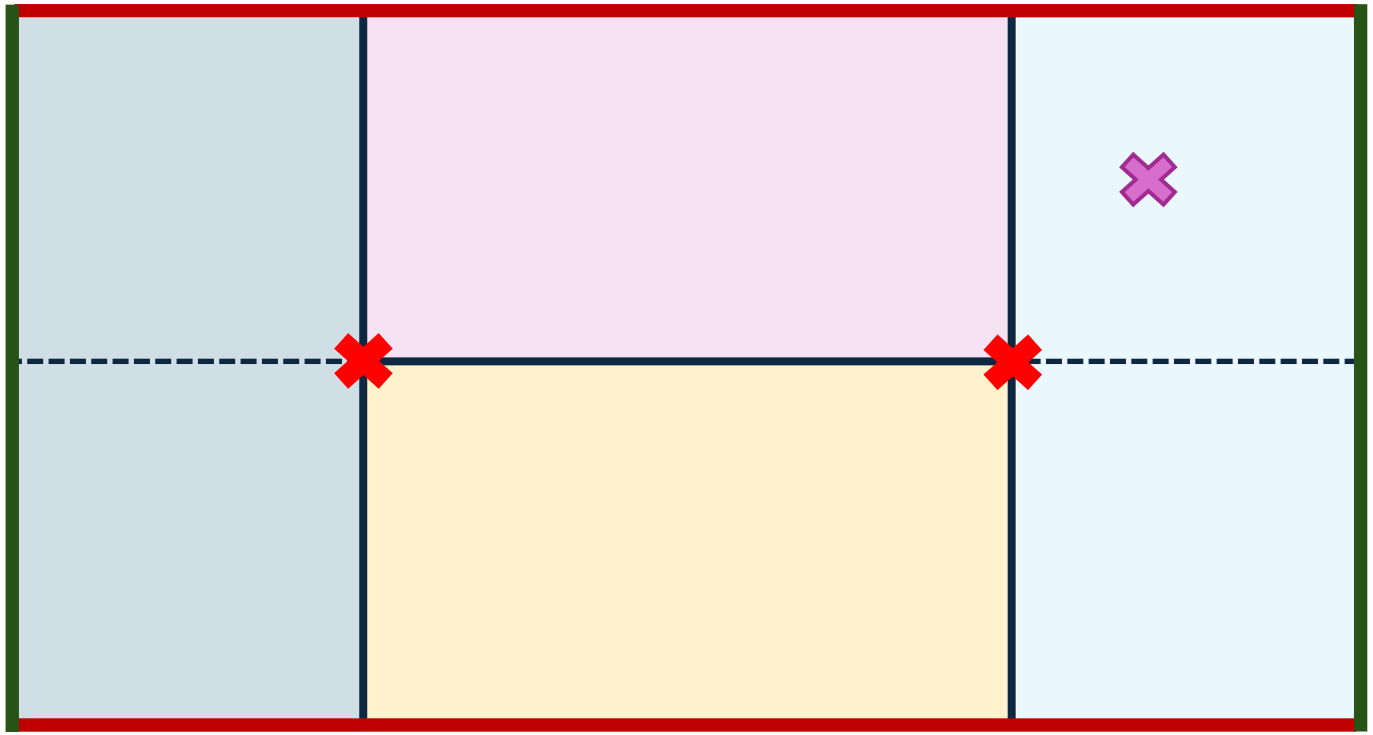
- SOLPS-ITER WG can simulate arbitrary magnetic topologies
- Goat can create grids on arbitrary magnetic topologies
- … but this capability is difficult to exploit as SOLPS was only taught to expect a finite (and small) set of topologies
  For the rest, good simulation with no or wrong output diagnostics (best case), or run which fails to start (worst case)

- Catalogue and classify all topologies and teach SOLPS-ITER how to handle them is impossible (infinite possibilities)…
- … but algorithmic classification IS possible if we employ a simplifying (yet relevant) assumption:
  Magnetic topology with only one O-point, with X-point(s) being only far above or far below the O-point
- In this case, we CAN teach SOLPS how to deal with an arbitrary number (and positions) of the X-points!

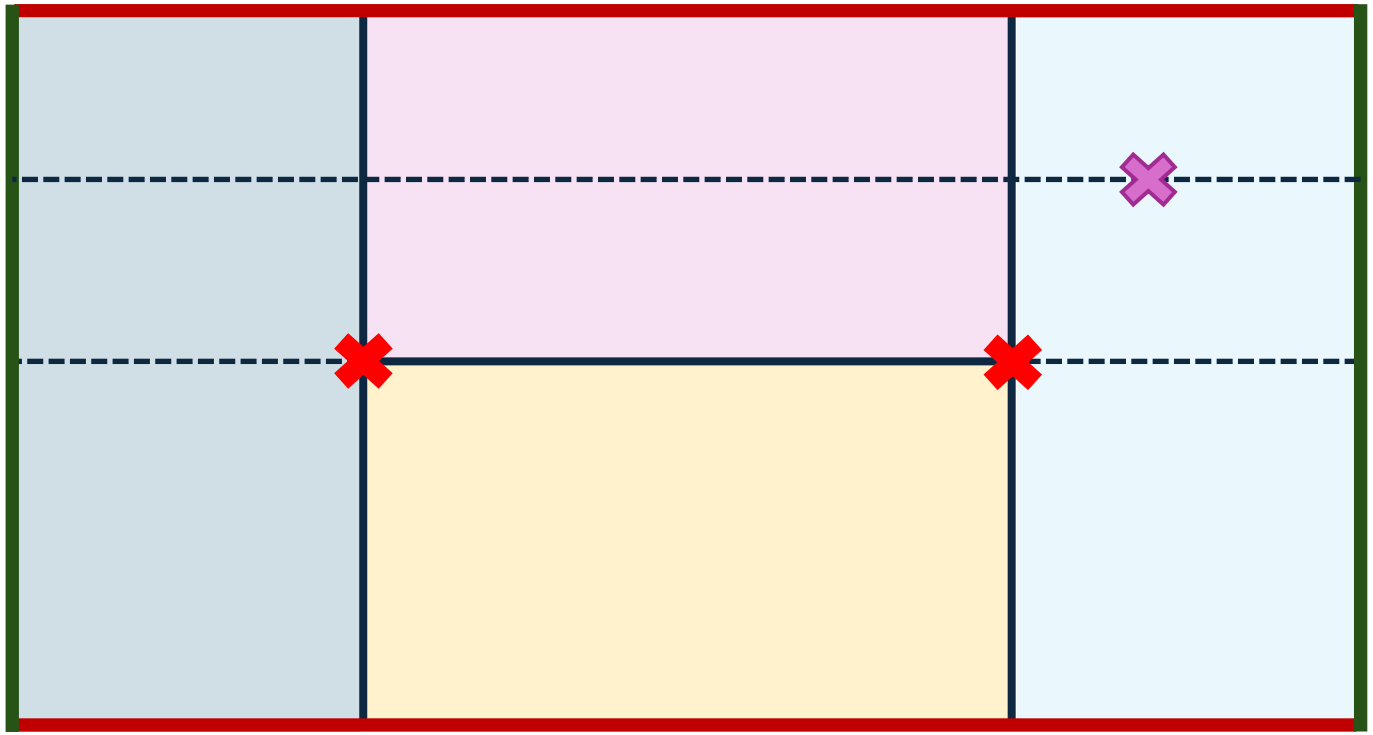
Basic single null scenario
(X-point only at one side of the plasma, lower OR upper)



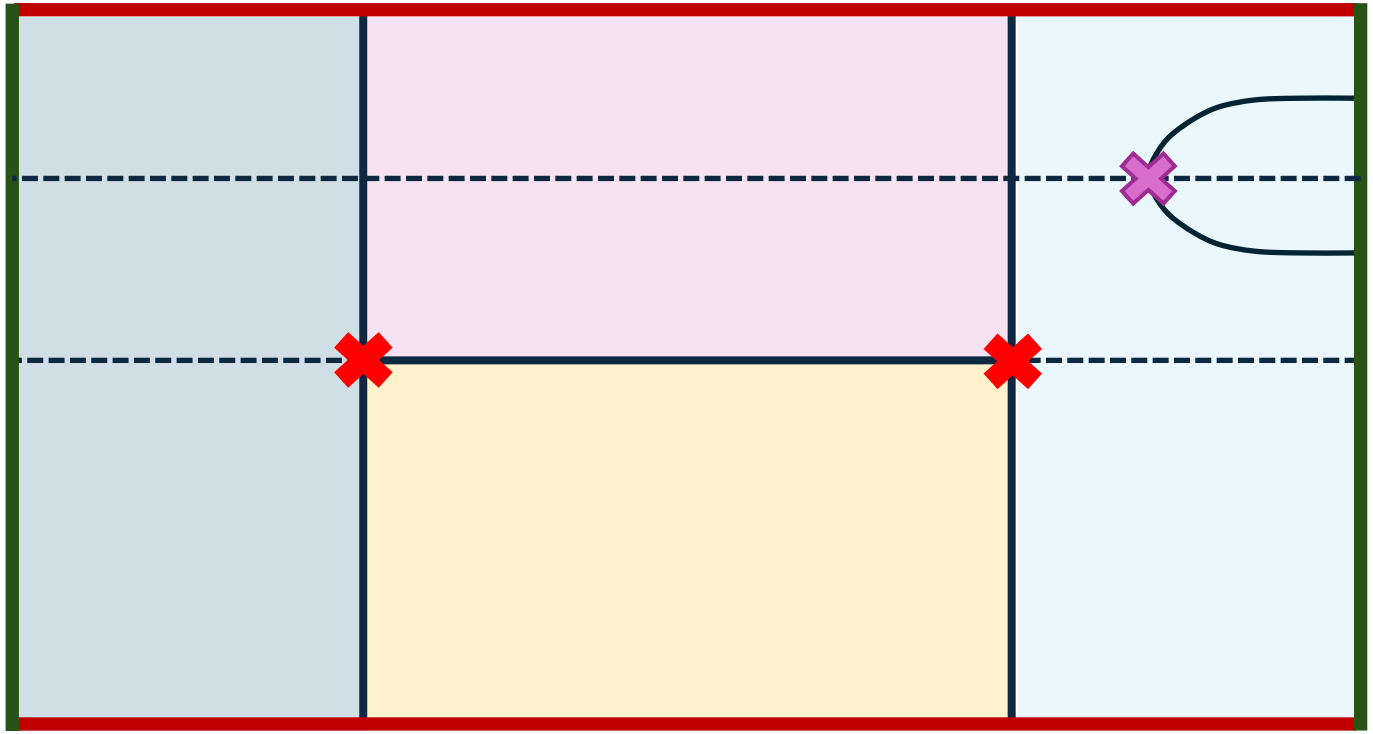
Let’s add an additional X-point



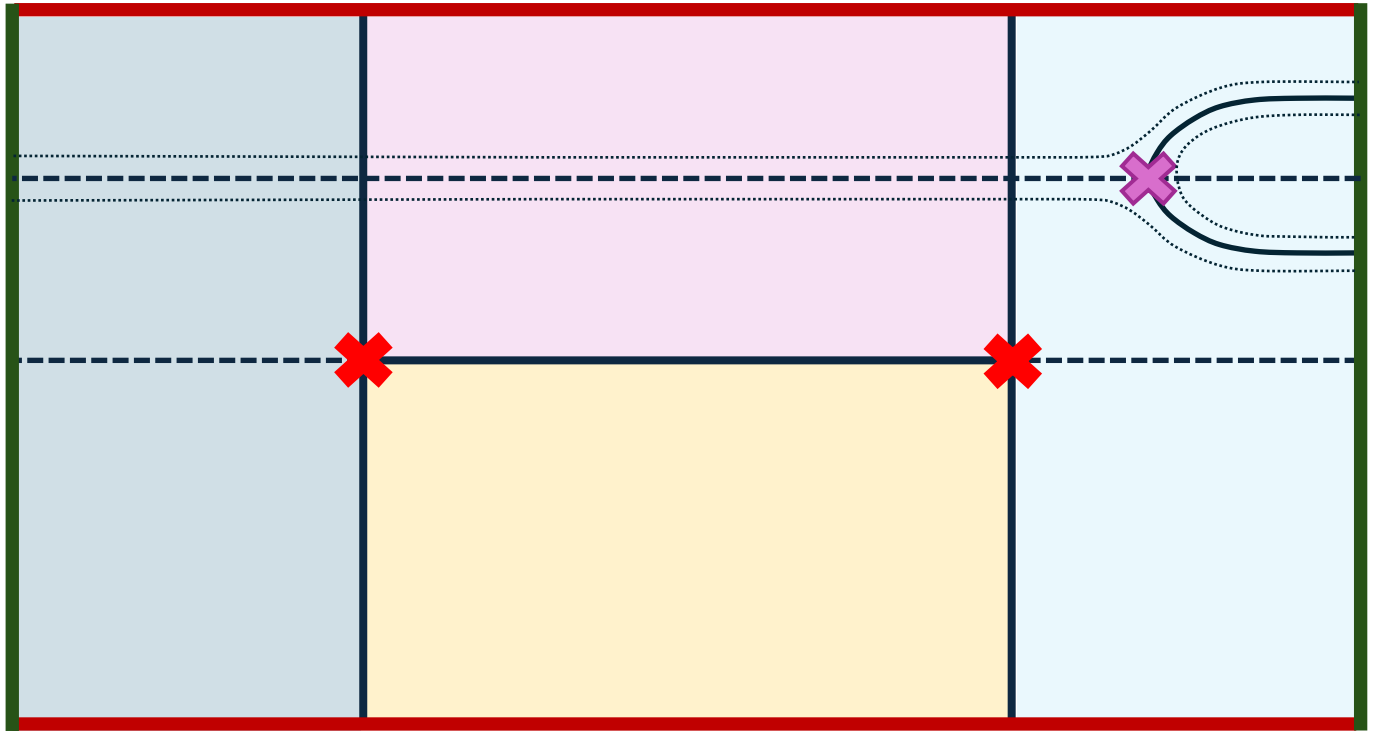
Let’s draw the corresponding separatrix



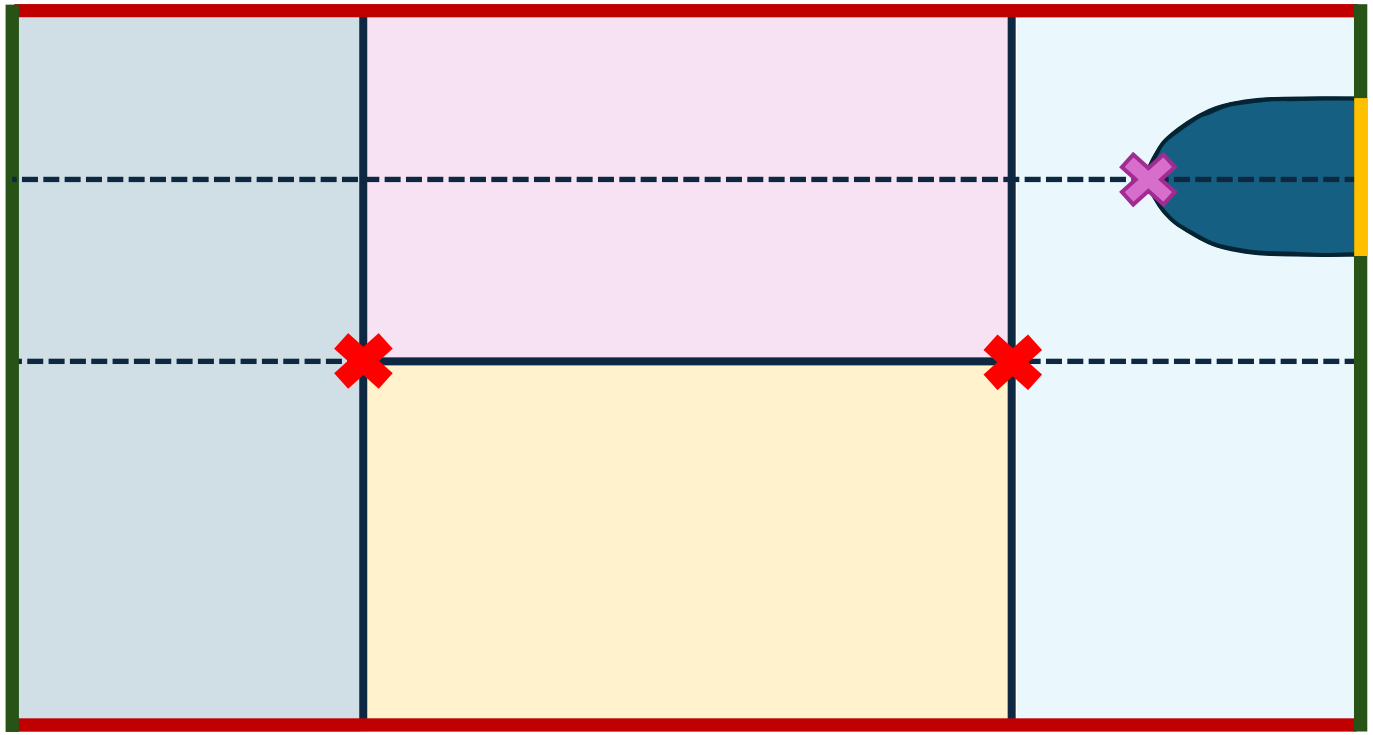
Let’s add the additional leg branches



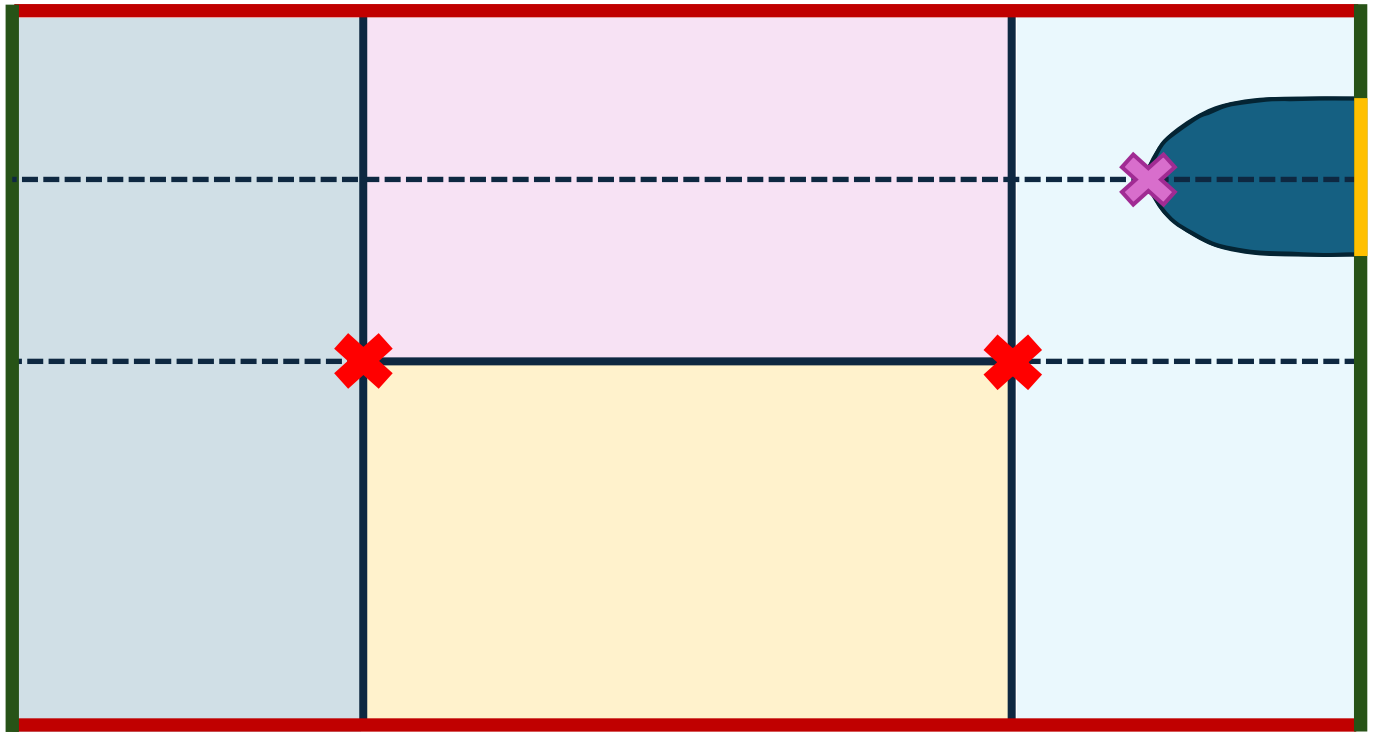
Topologically distinct region:
not connected anymore to plasma upstream (and to other target)



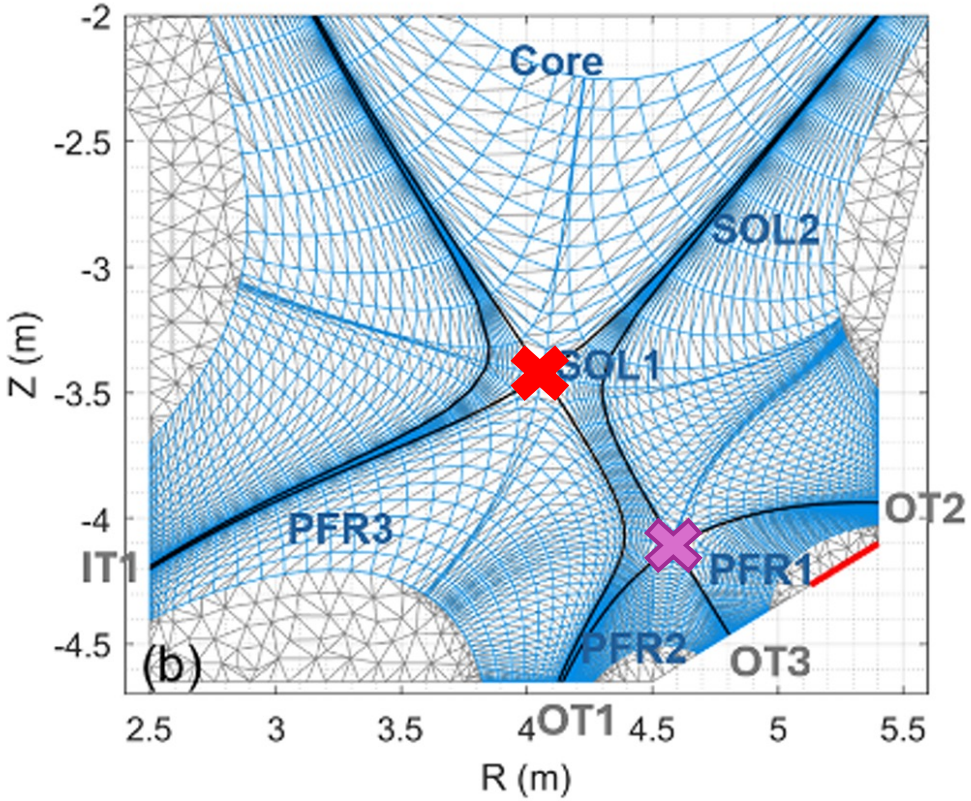
Therefore:
new volume region (disconnected PFR)

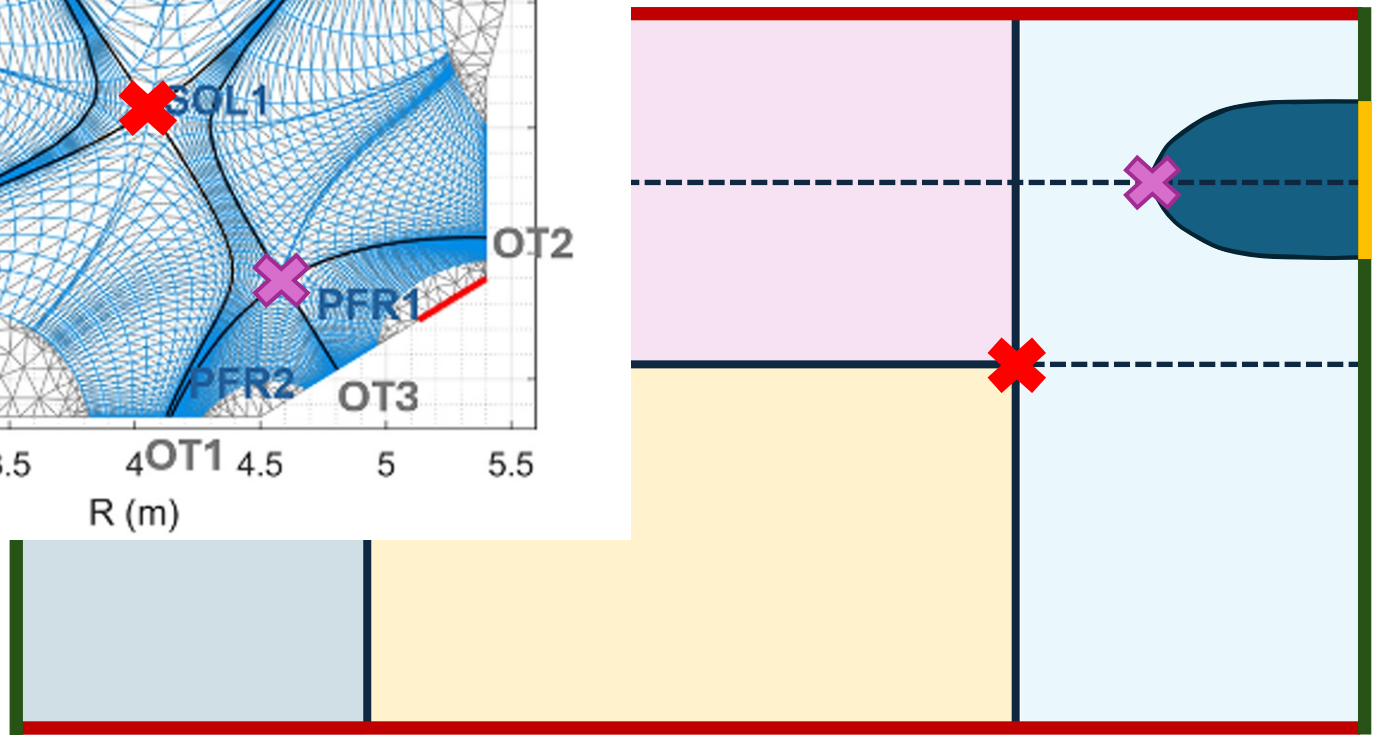


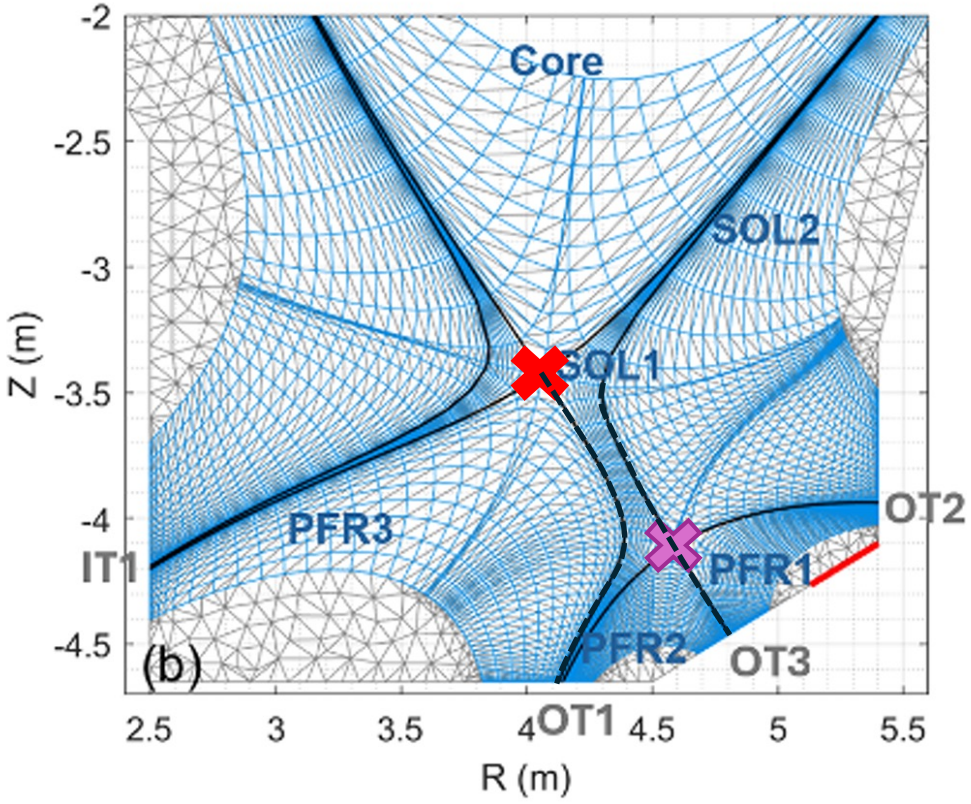
Therefore:
new volume region (disconnected PFR)

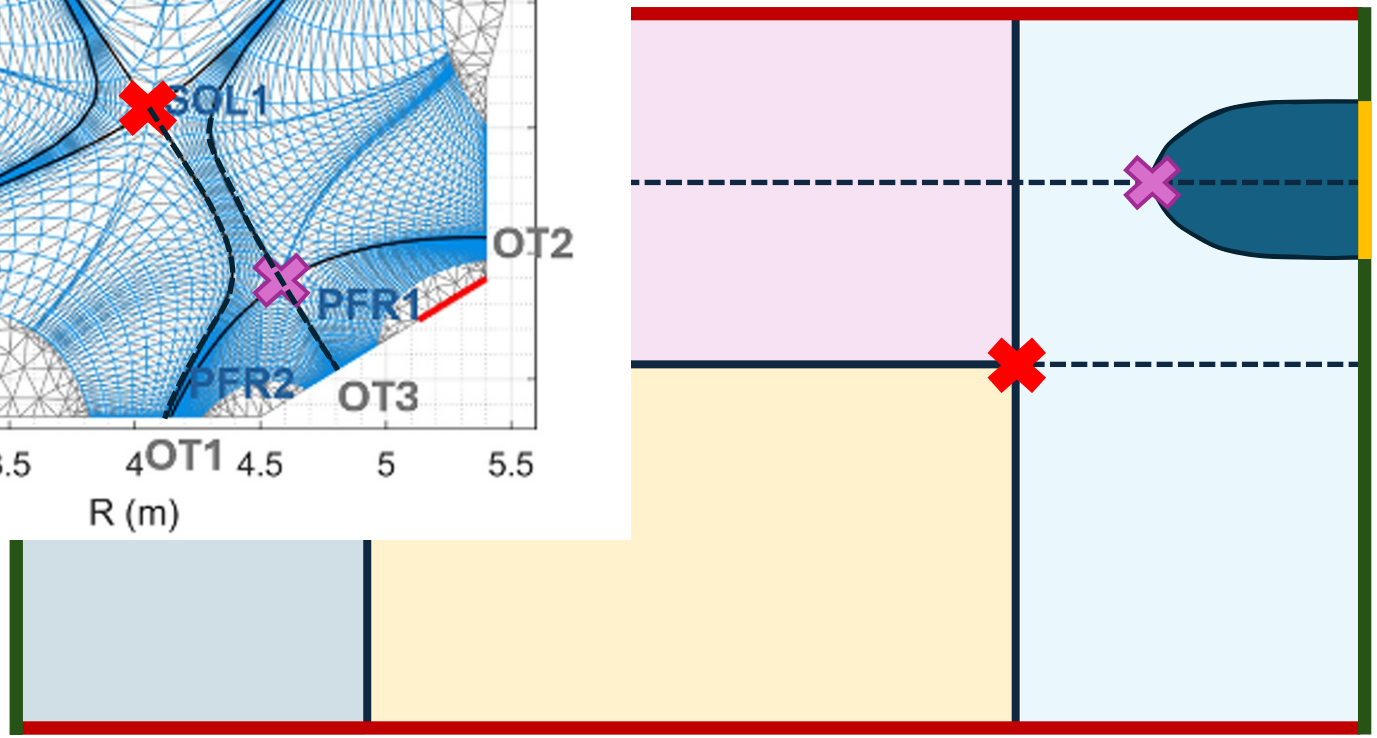


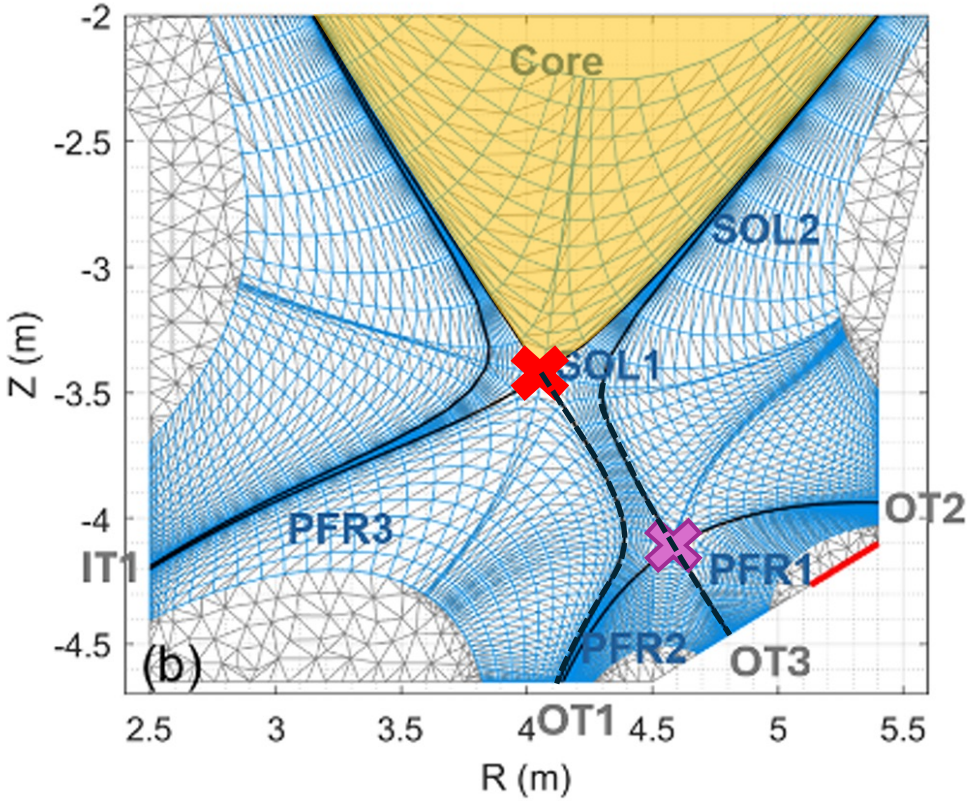
That’s it! We have a LFS SF- !

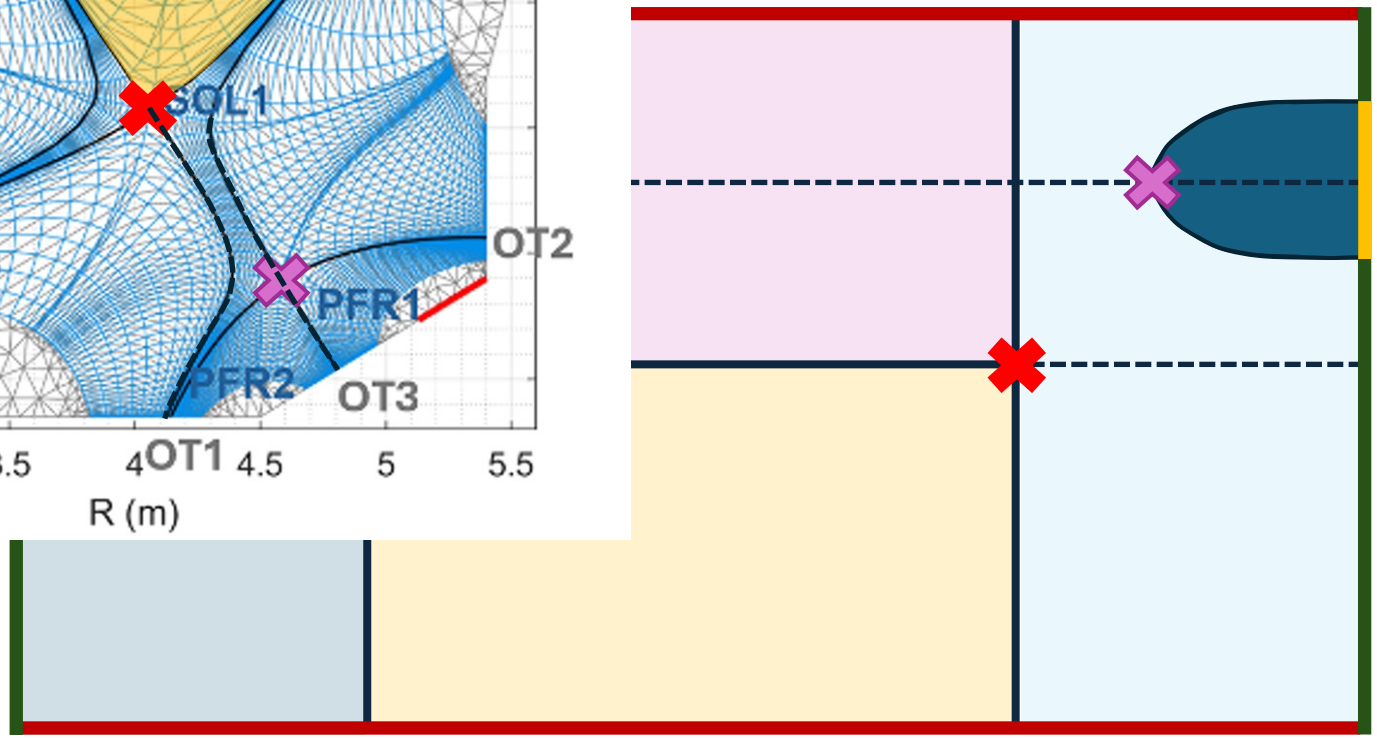

Core
SOL2
SOL1
PFR3
PFR1
PFR2
OT2
OT3
OT1
IT1
(b)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

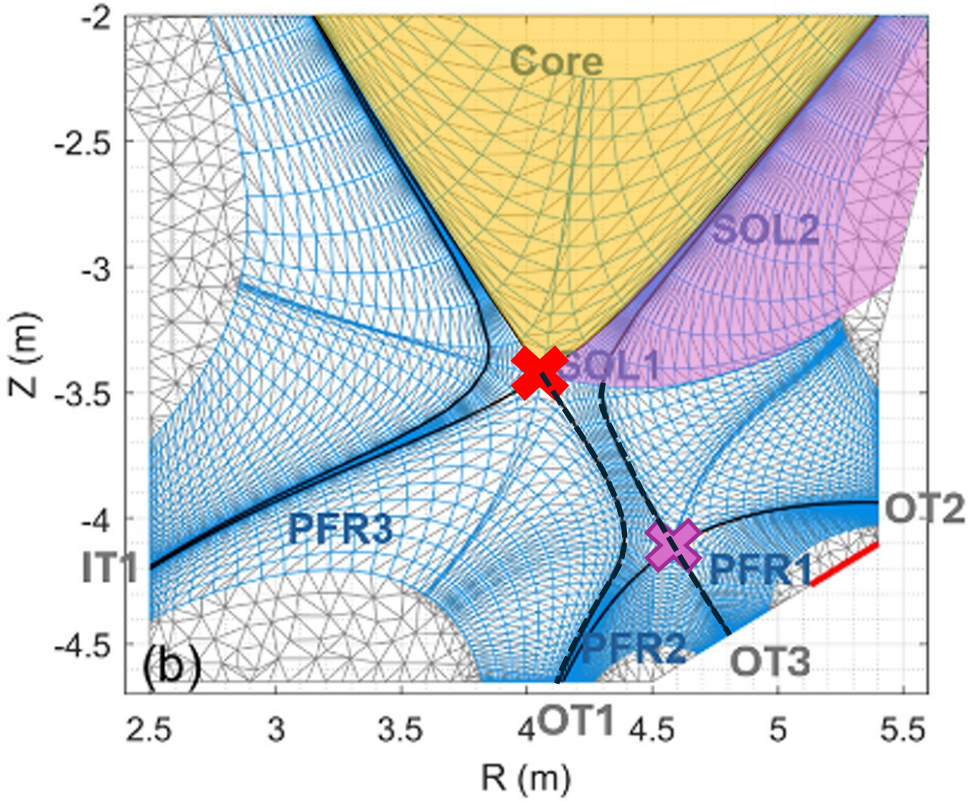


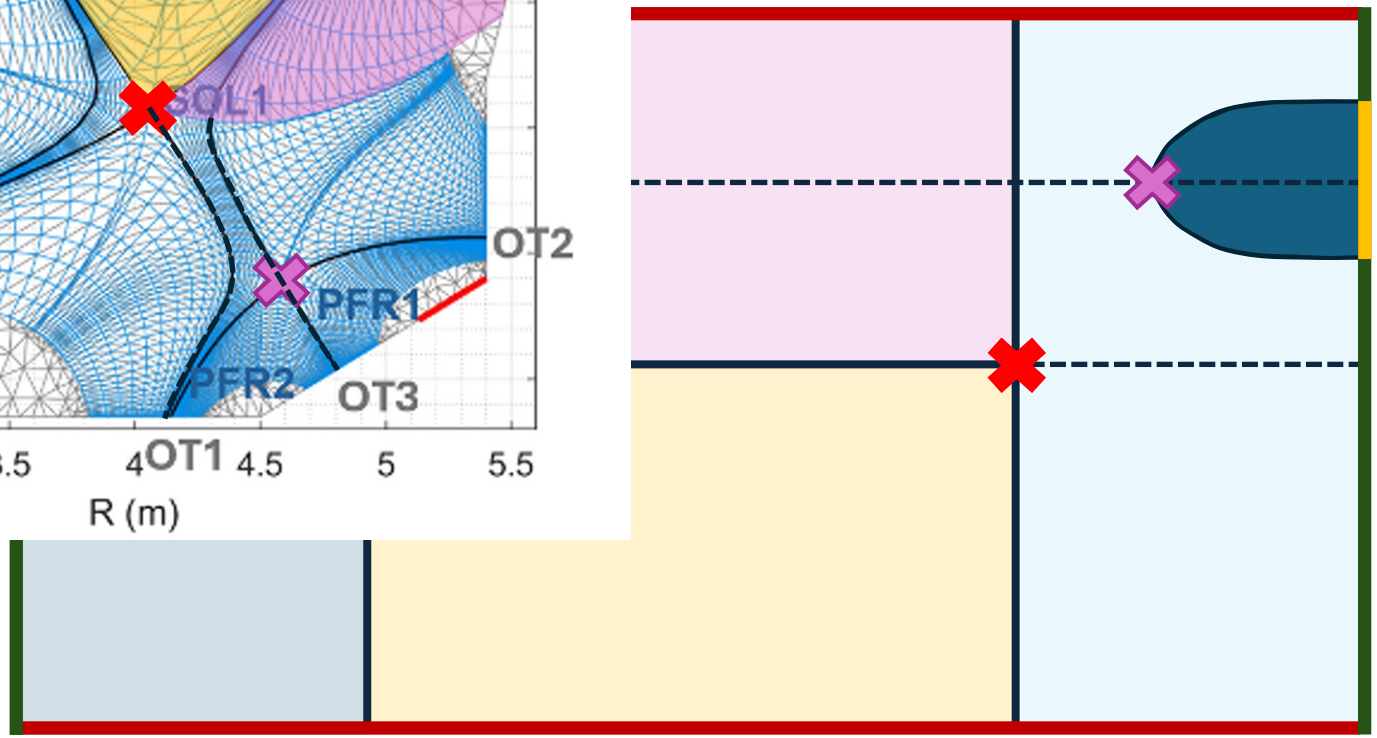

Core
SOL2
SOL1
PFR3
IT1
PFR1
OT2
PFR2
OT3
OT1
(b)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

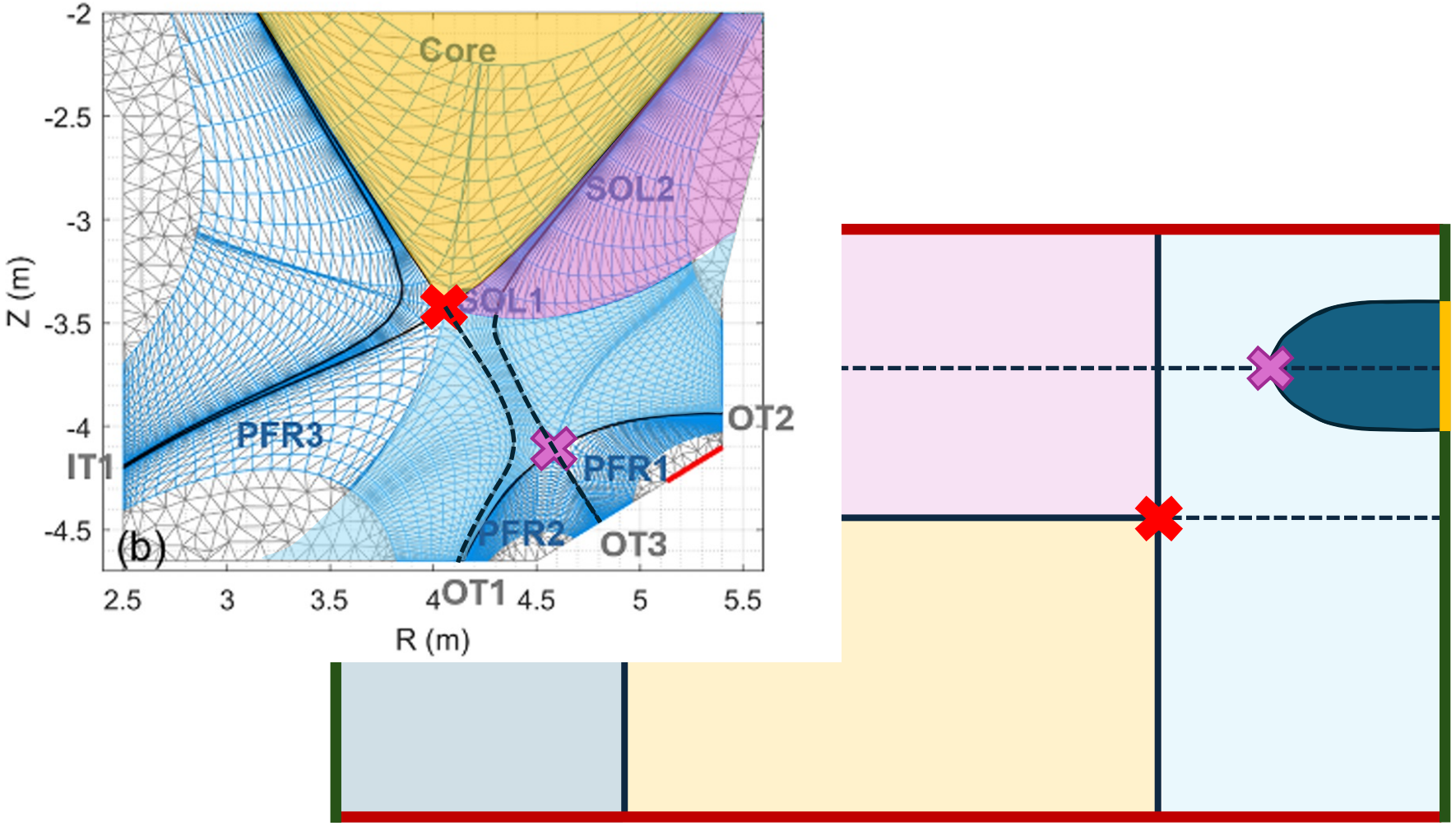


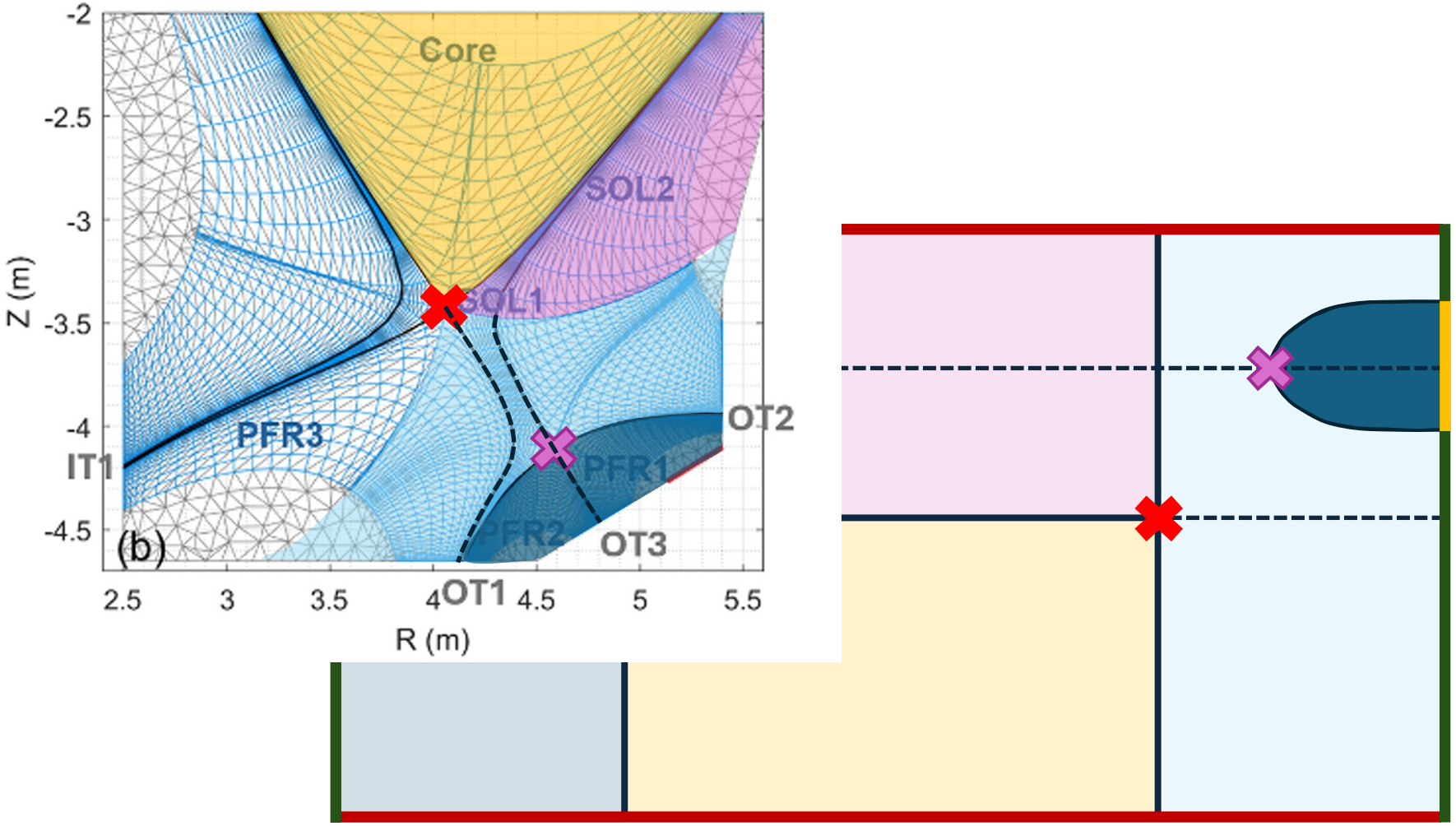

Core
SOL2
SOL1
PFR3
IT1
PFR1
OT2
PFR2
OT3
OT1
(b)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5



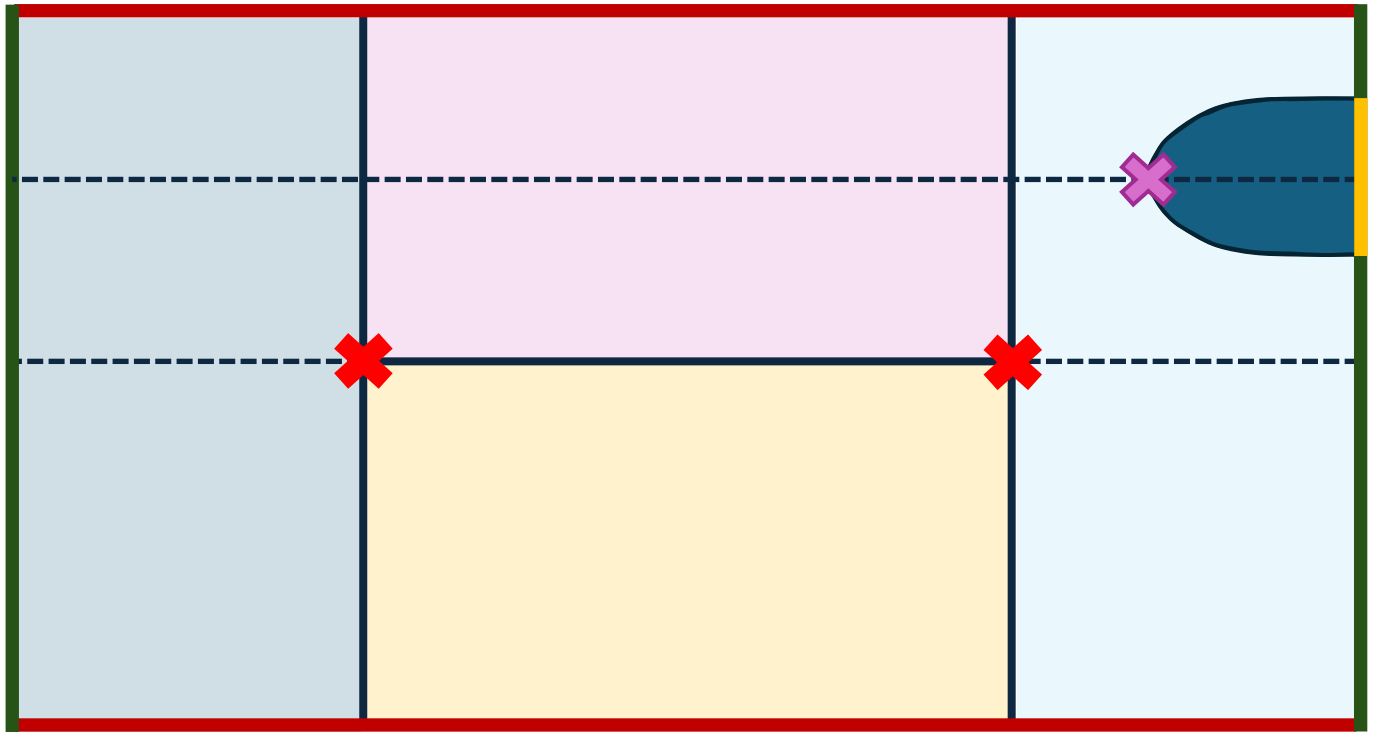

Core
SOL2
SOL1
PFR3
PFR1
PFR2
OT2
OT3
OT1
IT1
(b)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5



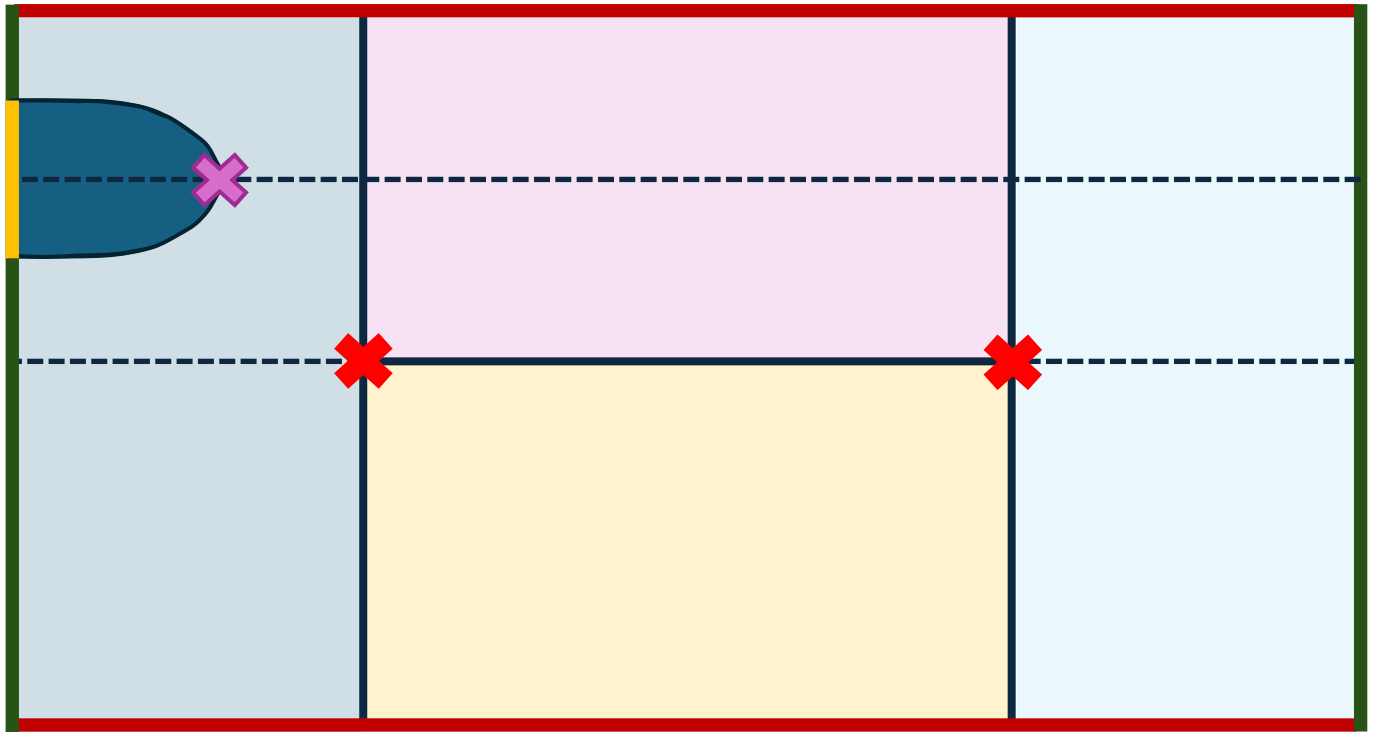

Core
SOL2
SOL1
PFR3
PFR1
PFR2
OT2
OT3
OT1
IT1
(b)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5


Core
SOL2
SOL1
PFR3
IT1
PFR1
PFR2
OT2
OT3
OT1
(b)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

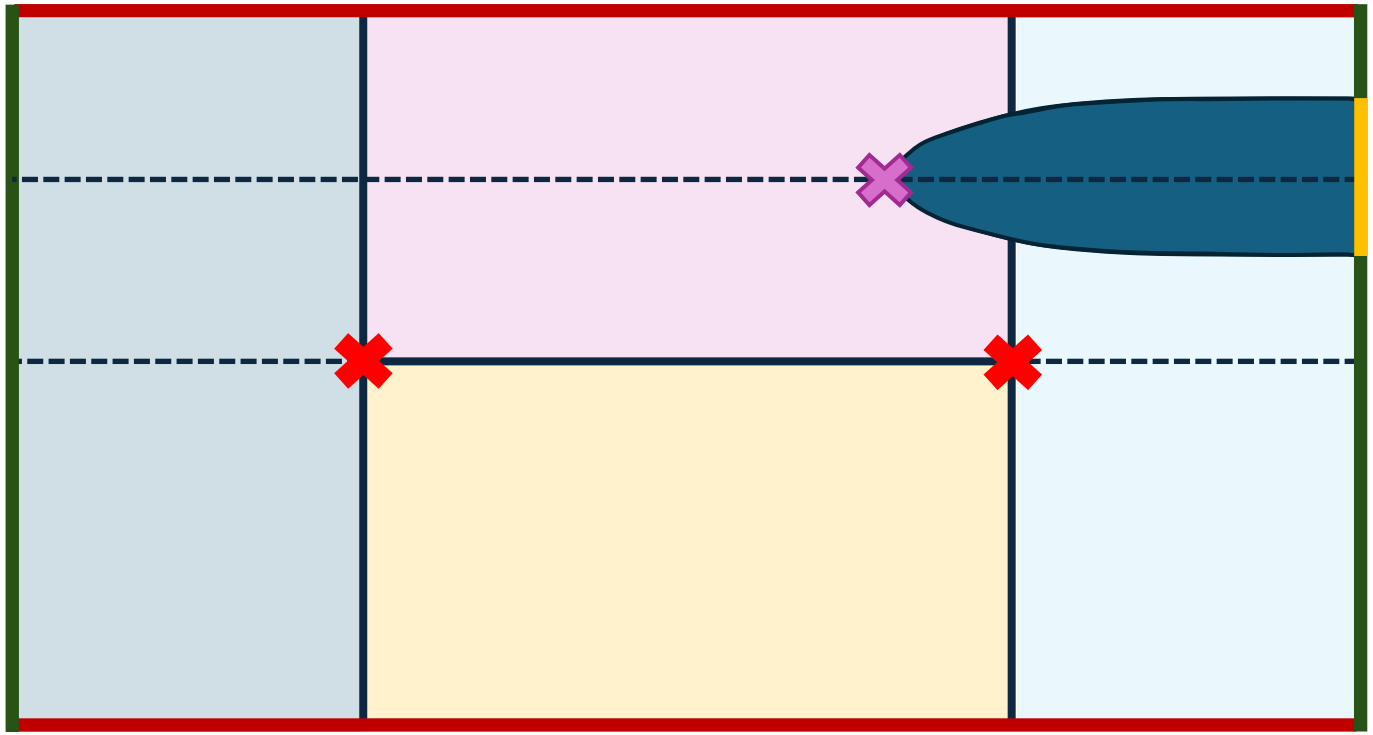
LFS SF- with X-point in divertor SOL



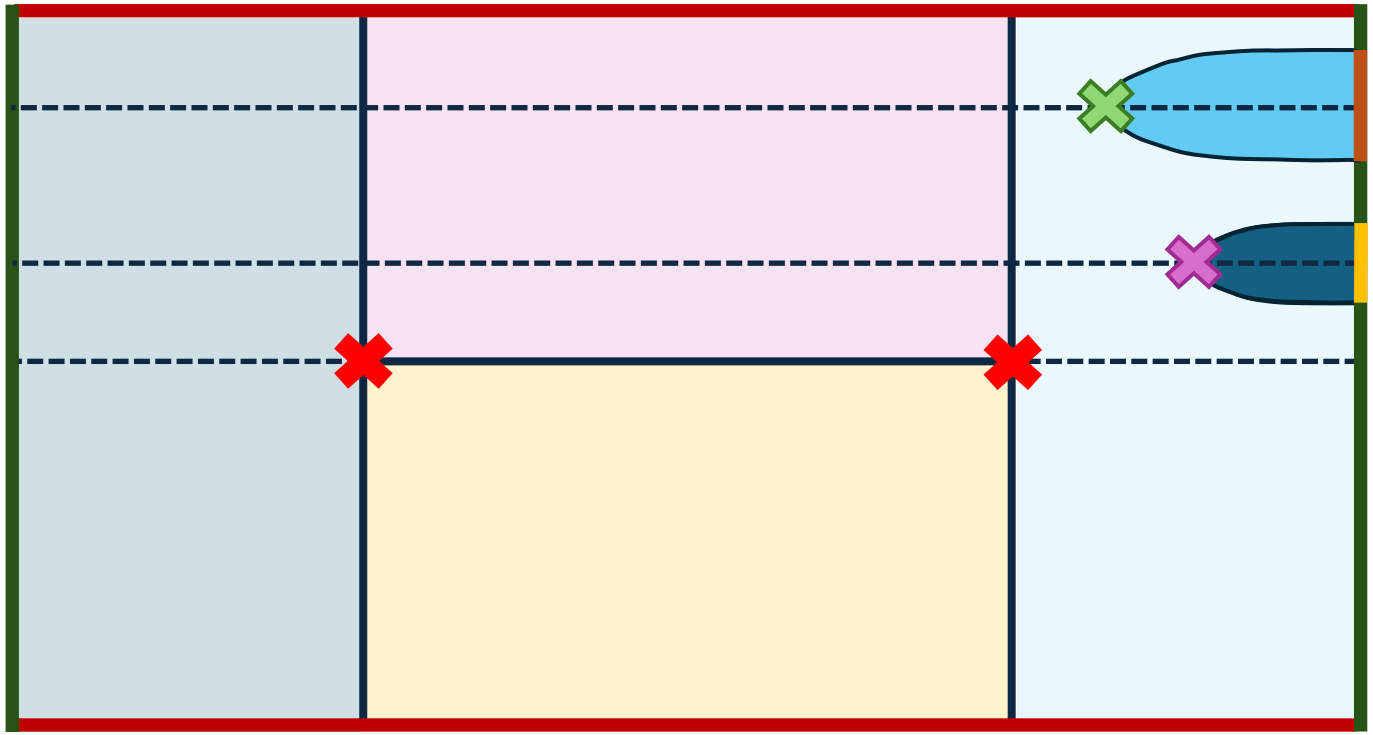
HFS SF- with X-point in divertor SOL



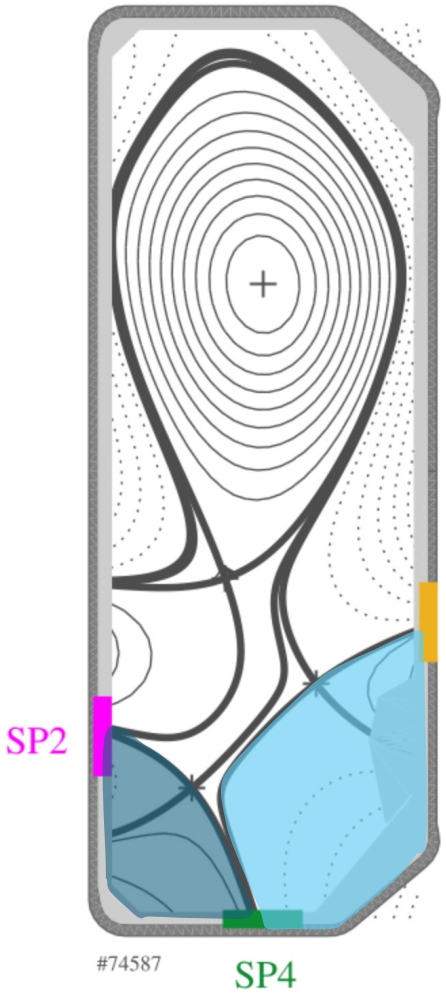
LFS SF- with X-point in main SOL

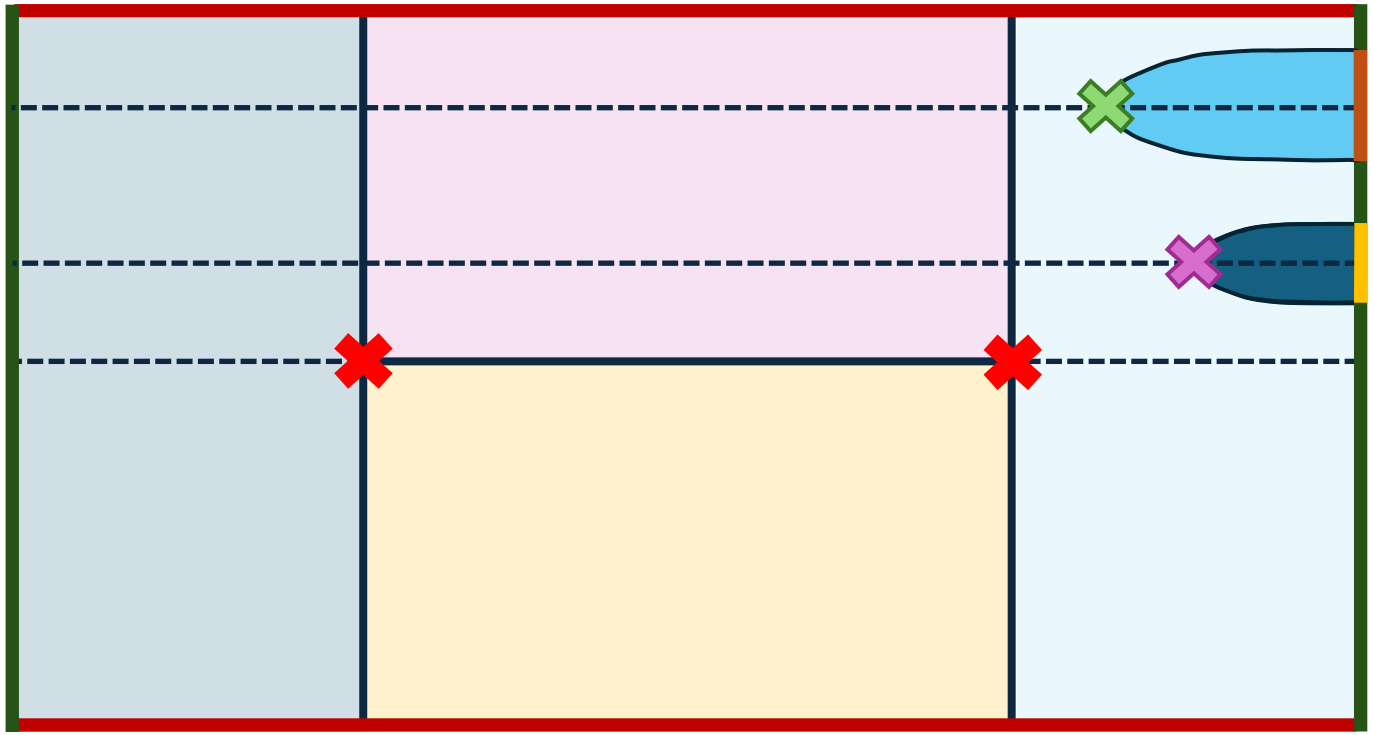


Multiple additional X-points at same side

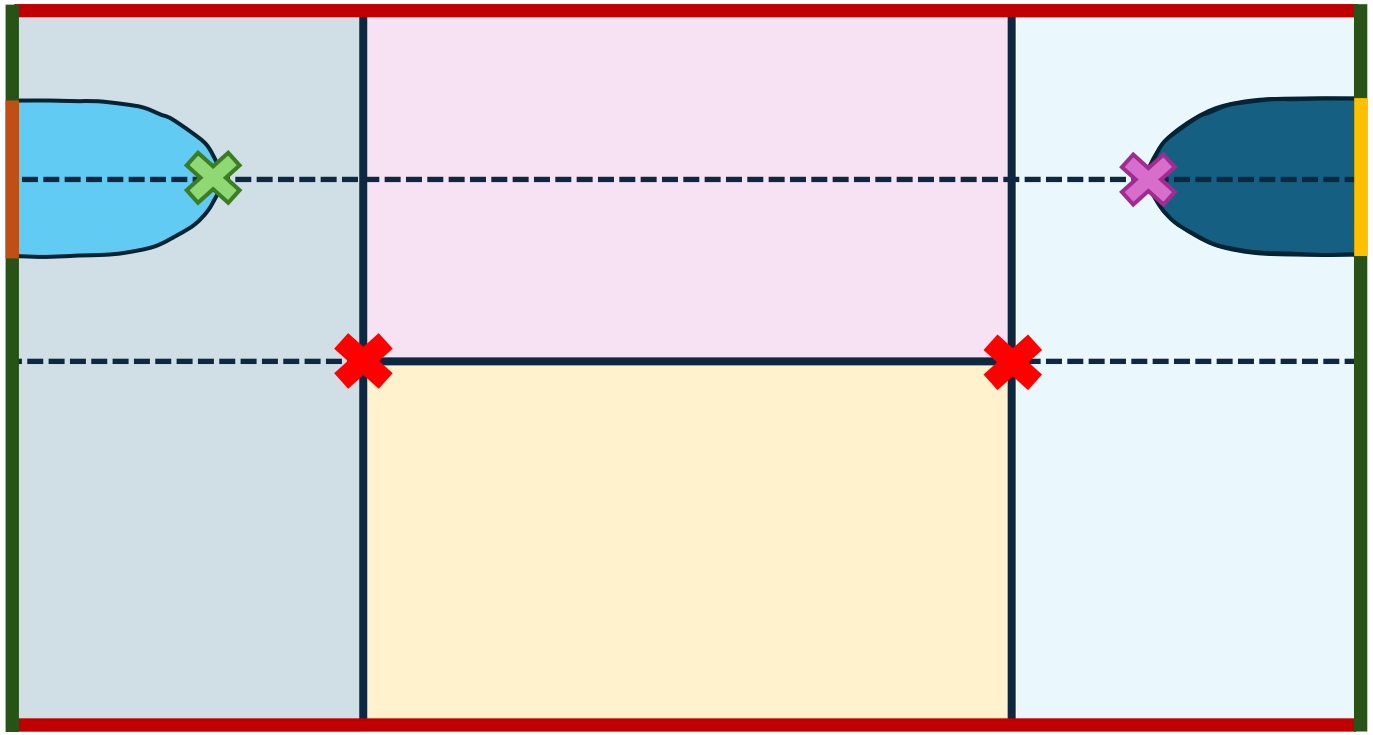


Multiple additional X-points at same side

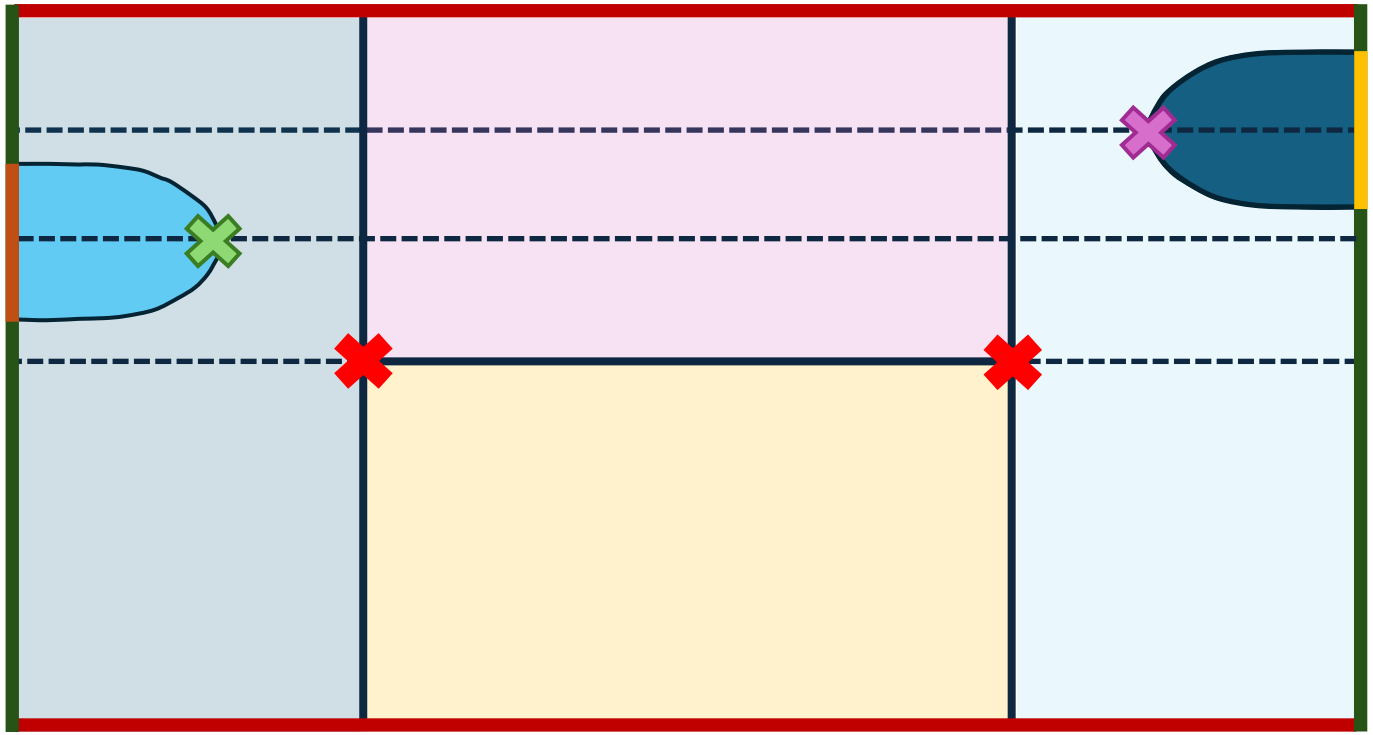




Connected additional X-points at opposite sides

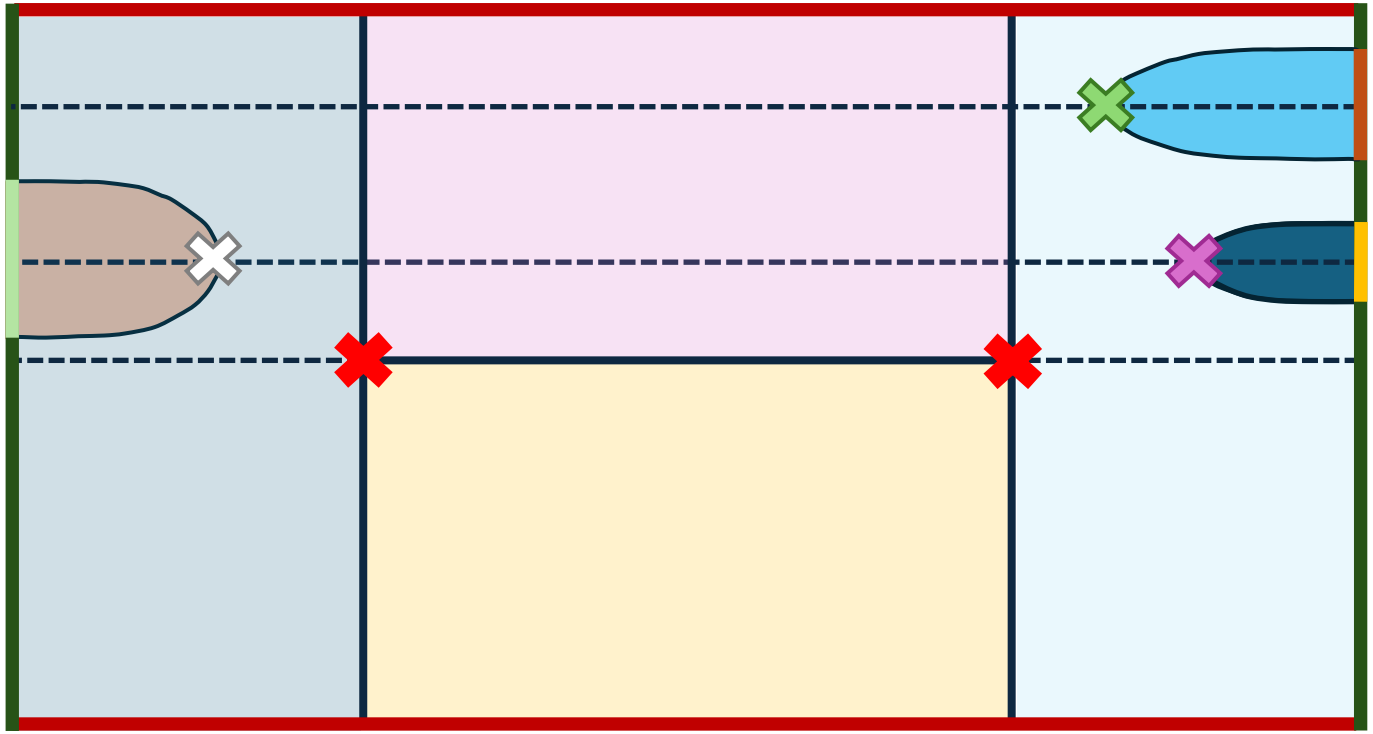


Disconnected additional X-points at opposite sides

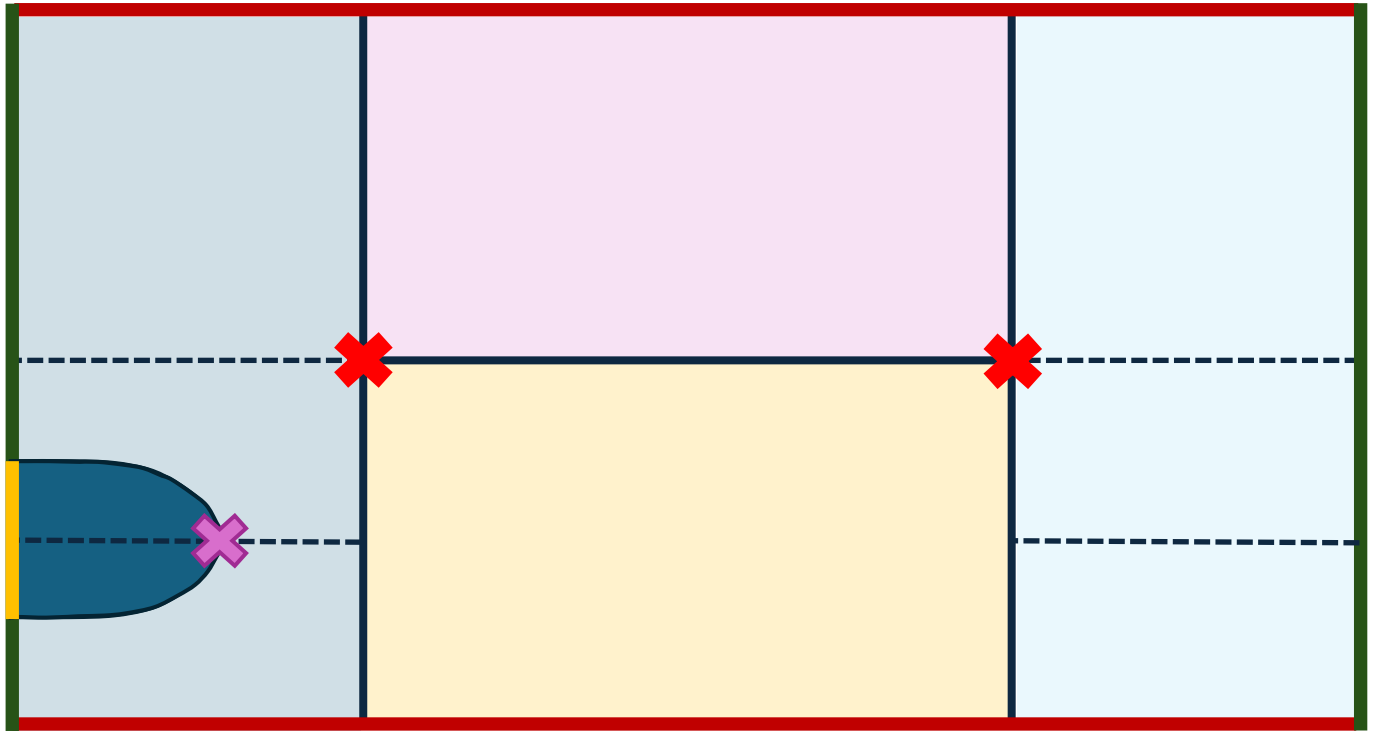


… basically any configuration you wish

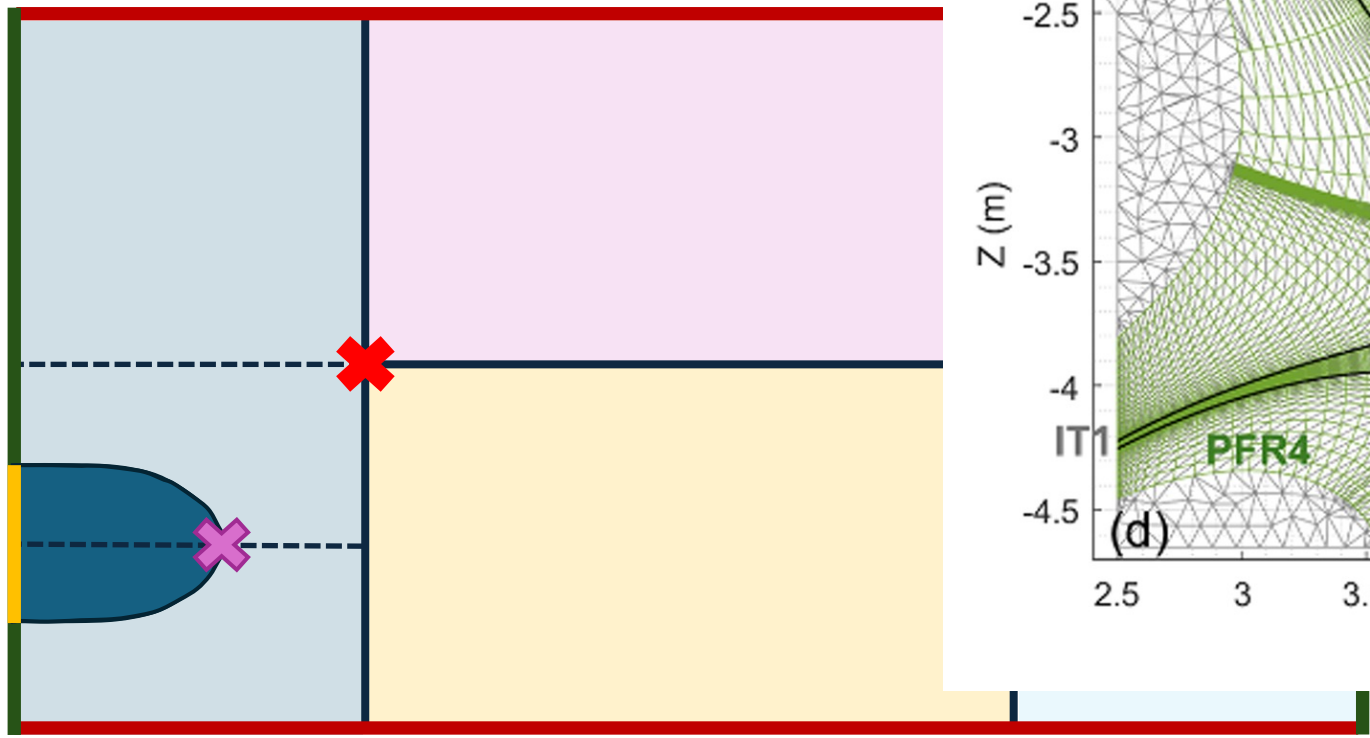
(and that your coil operators will hate)

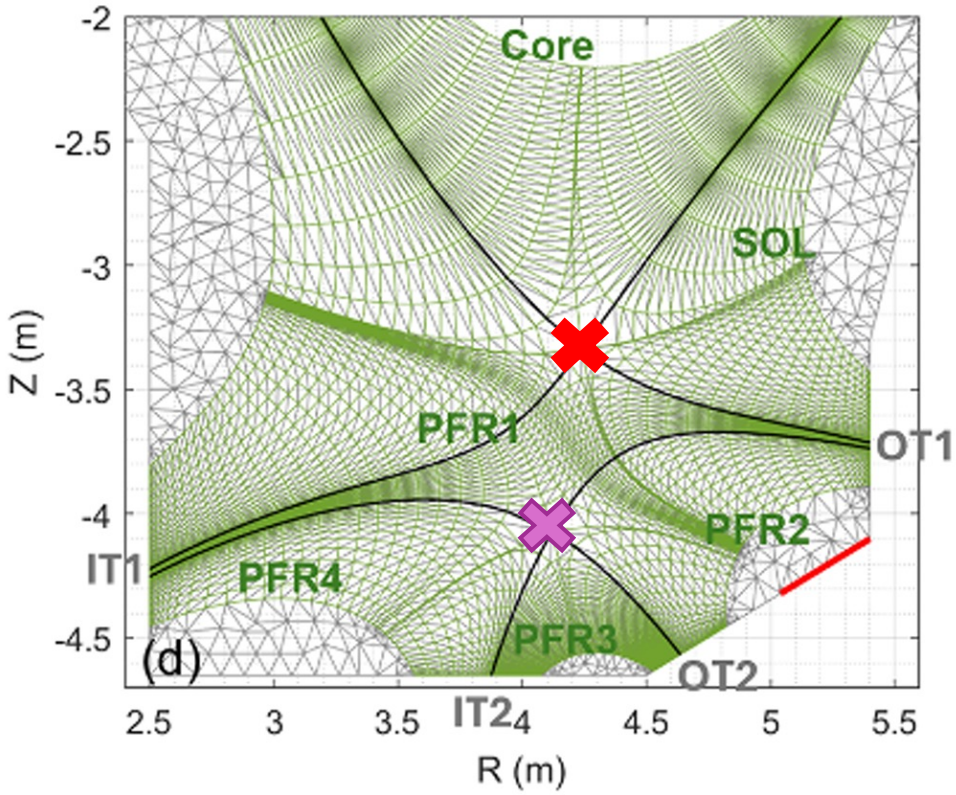


HFS SF+

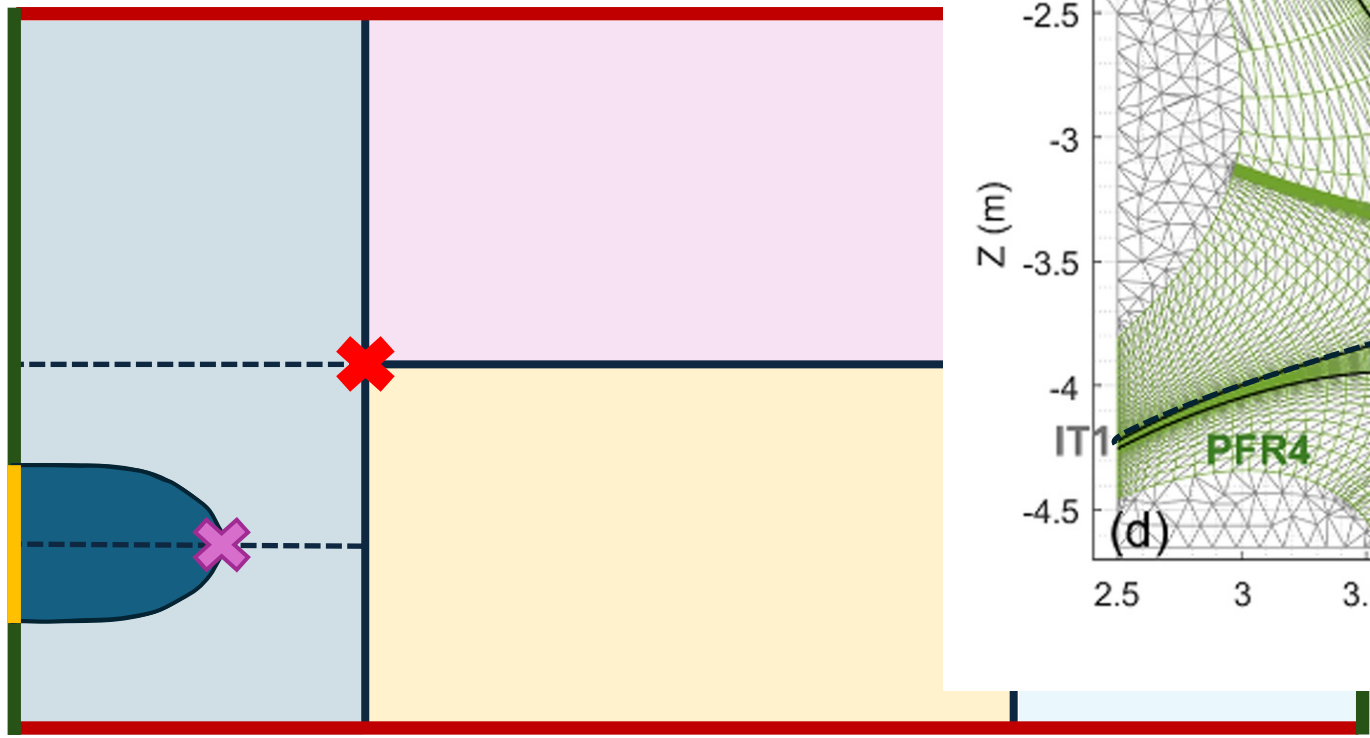


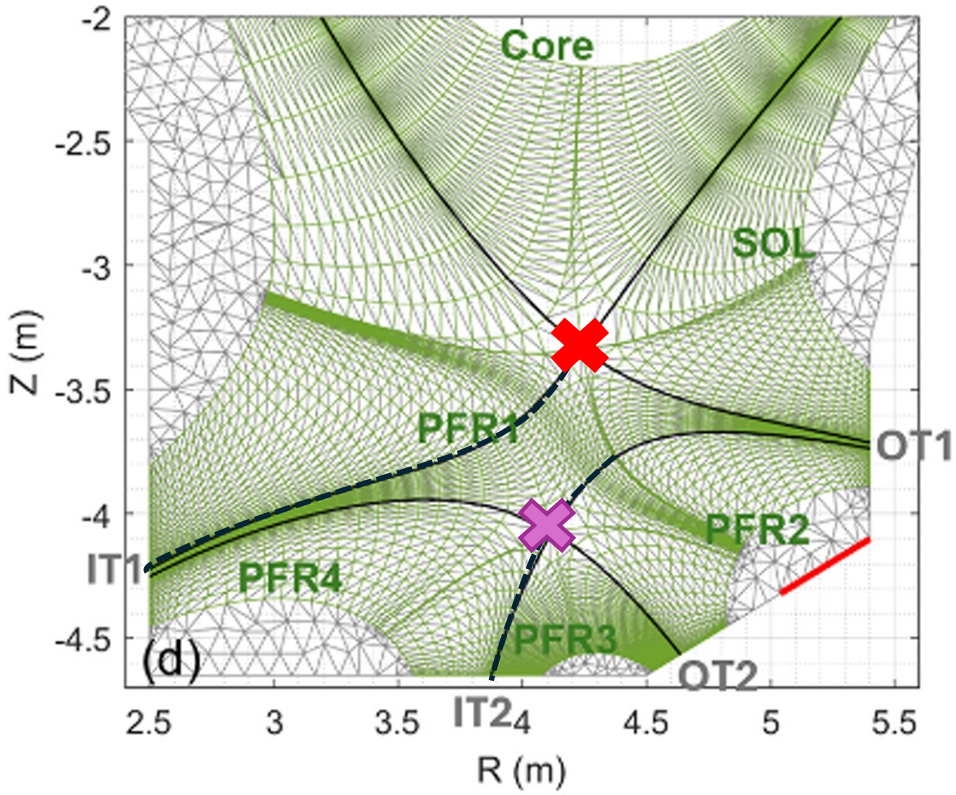
HFS SF+



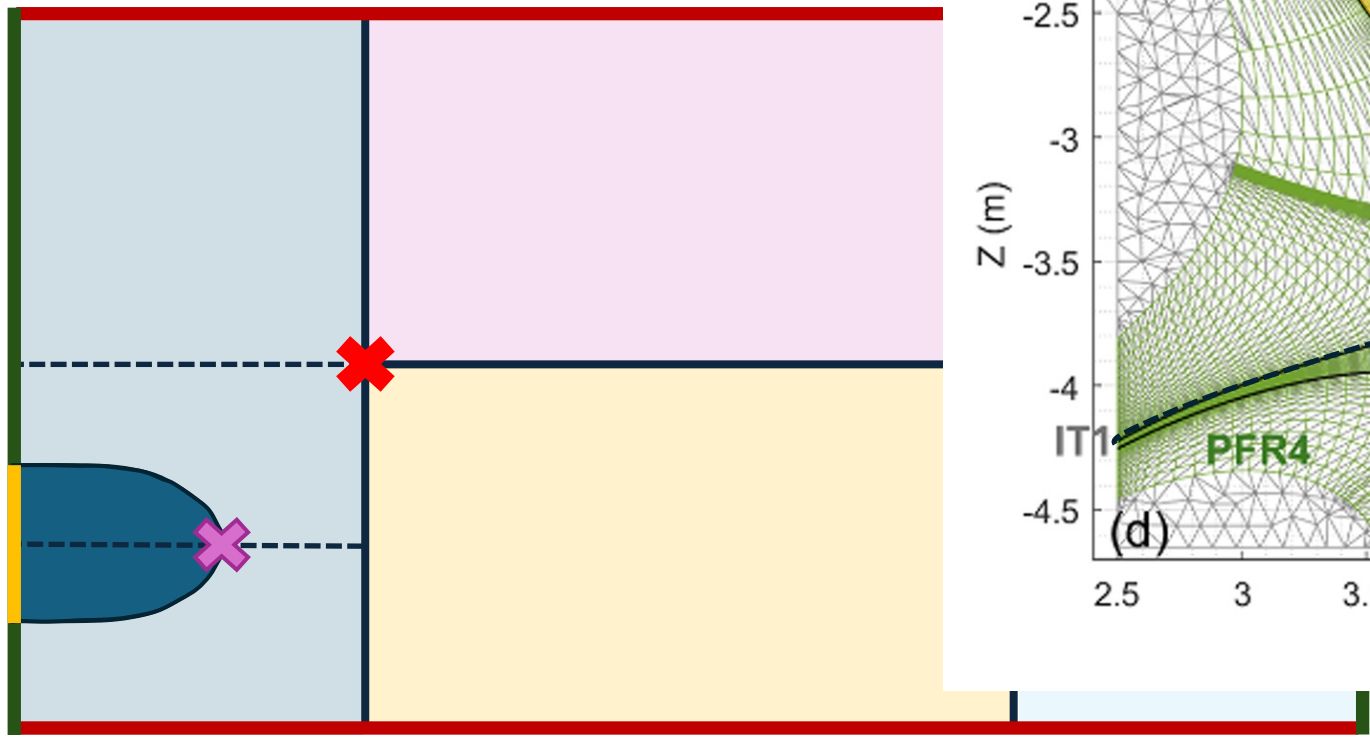

Core
SOL
PFR1
PFR2
PFR3
PFR4
OT1
OT2
IT1
IT2
(d)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

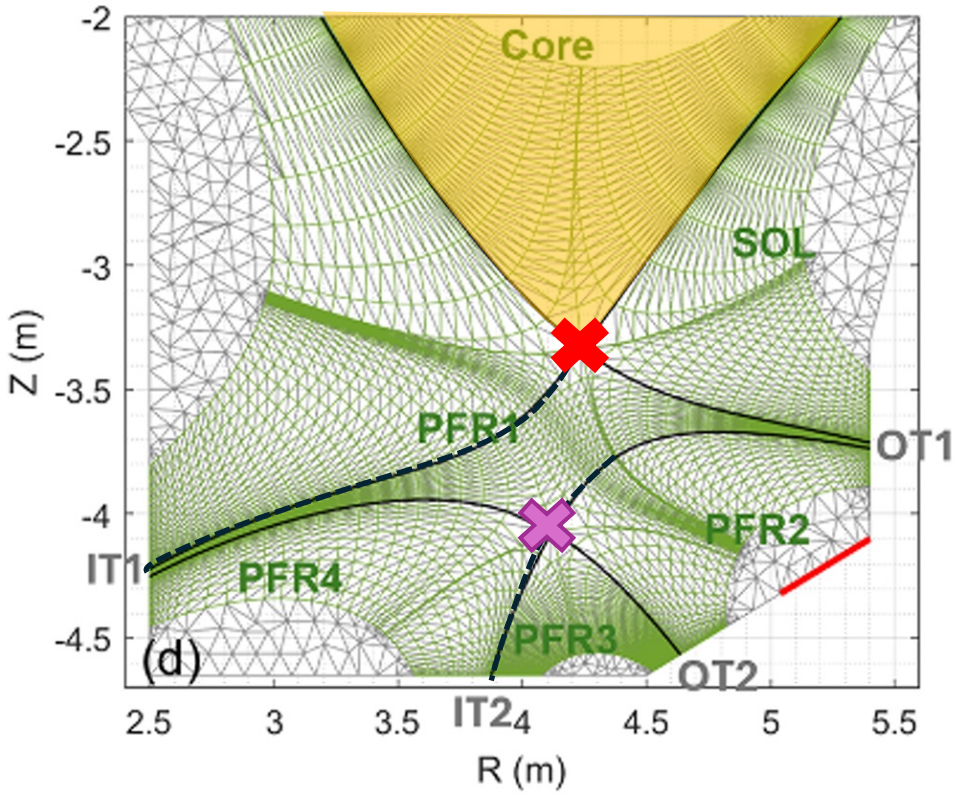
HFS SF+



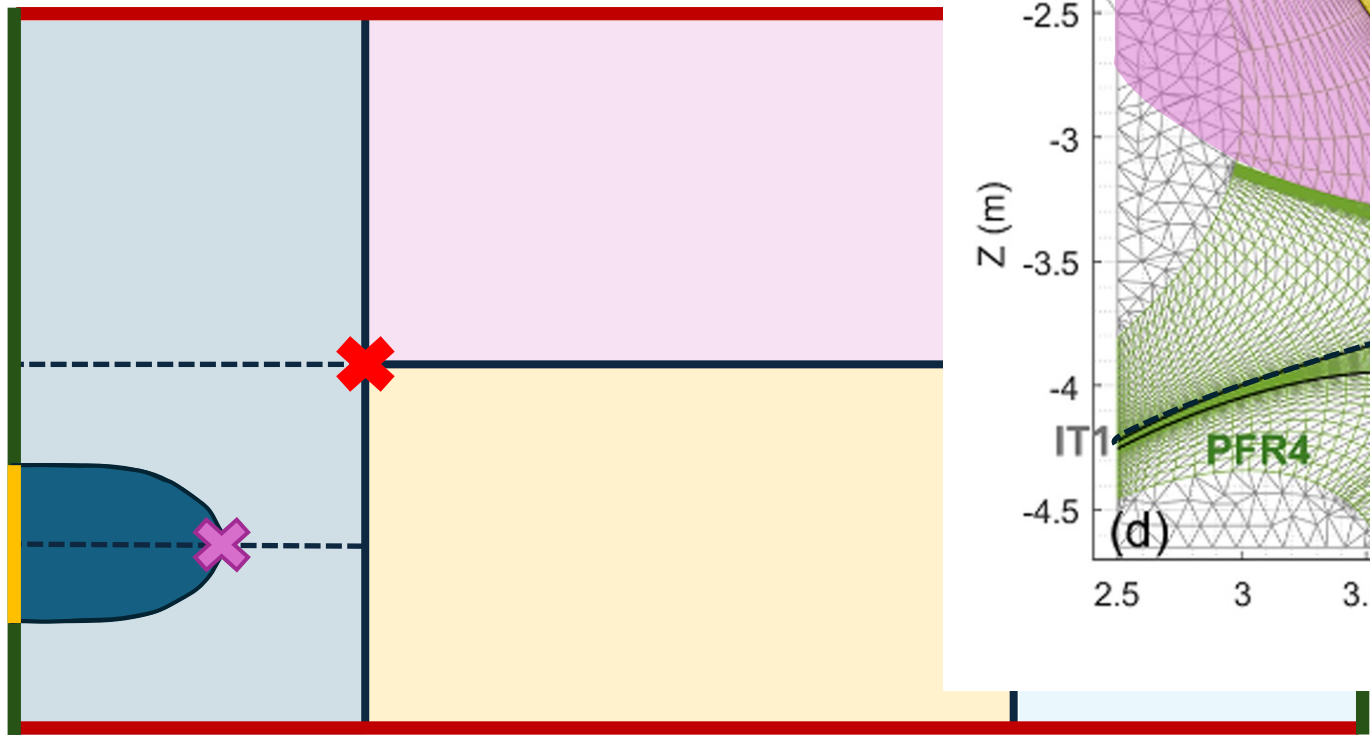

Core
SOL
PFR1
PFR2
PFR3
PFR4
OT1
OT2
IT1
IT2
(d)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

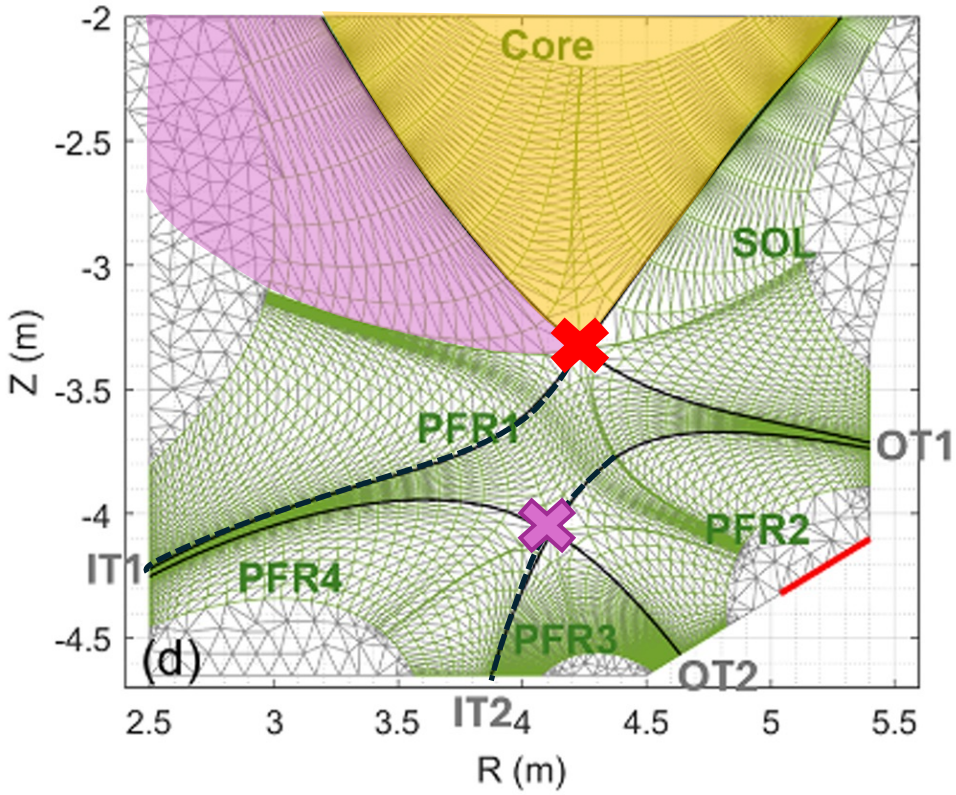
HFS SF+



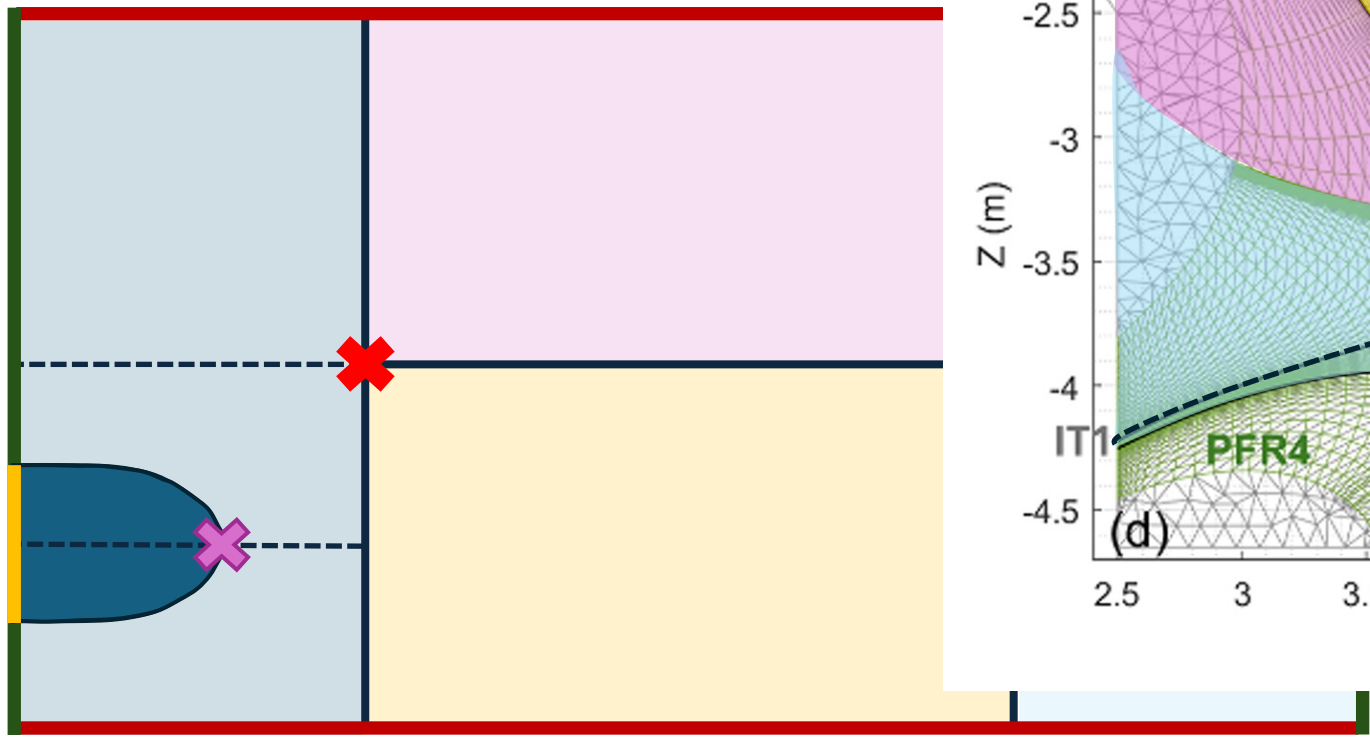

Core
SOL
PFR1
PFR2
PFR3
PFR4
OT1
OT2
IT1
IT2
(d)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

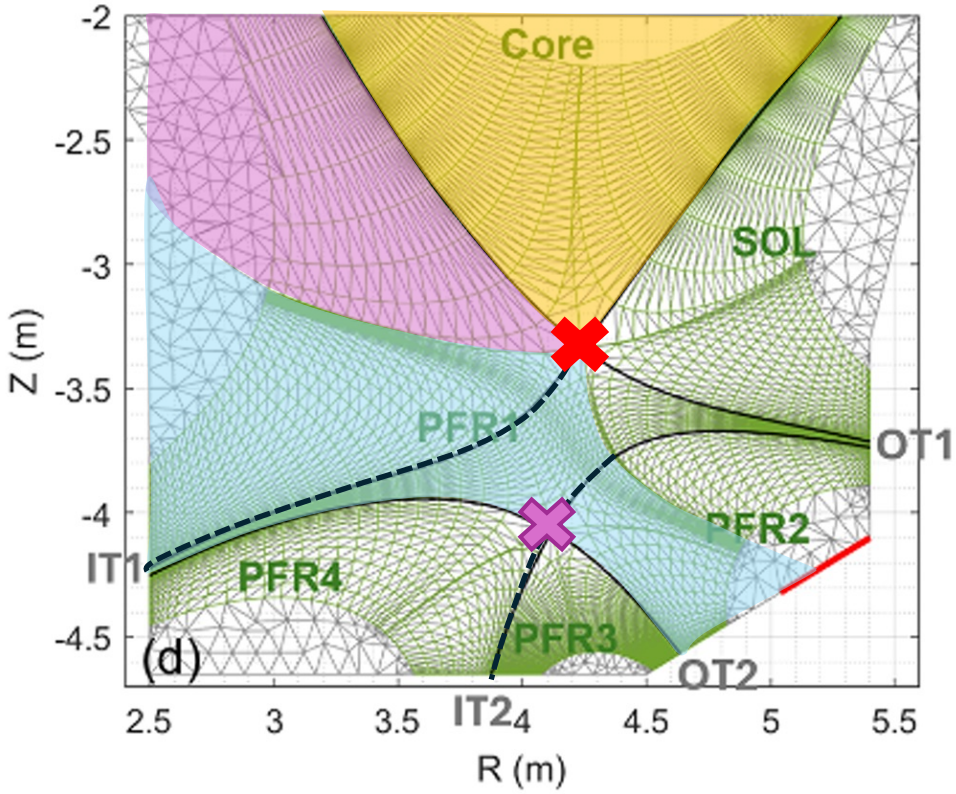
HFS SF+



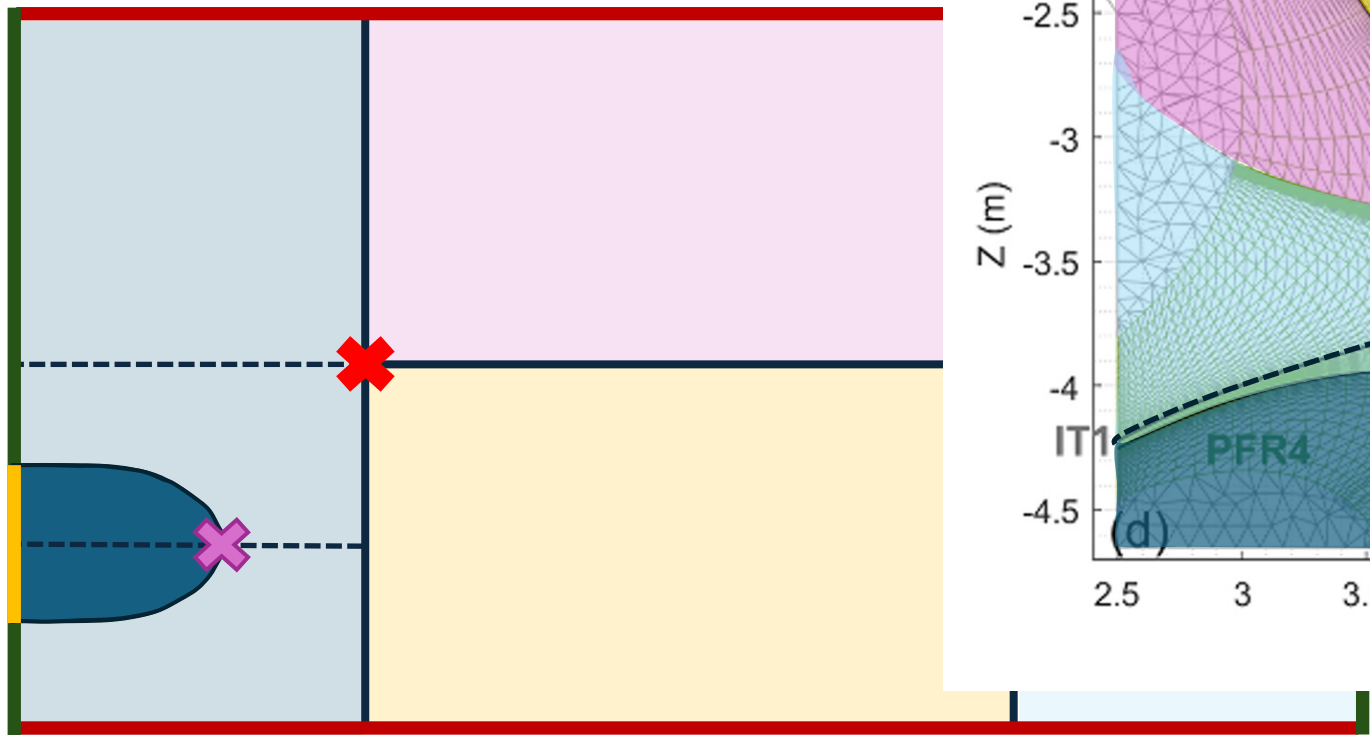

Core
SOL
PFR1
PFR2
PFR3
PFR4
OT1
OT2
IT1
IT2
(d)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

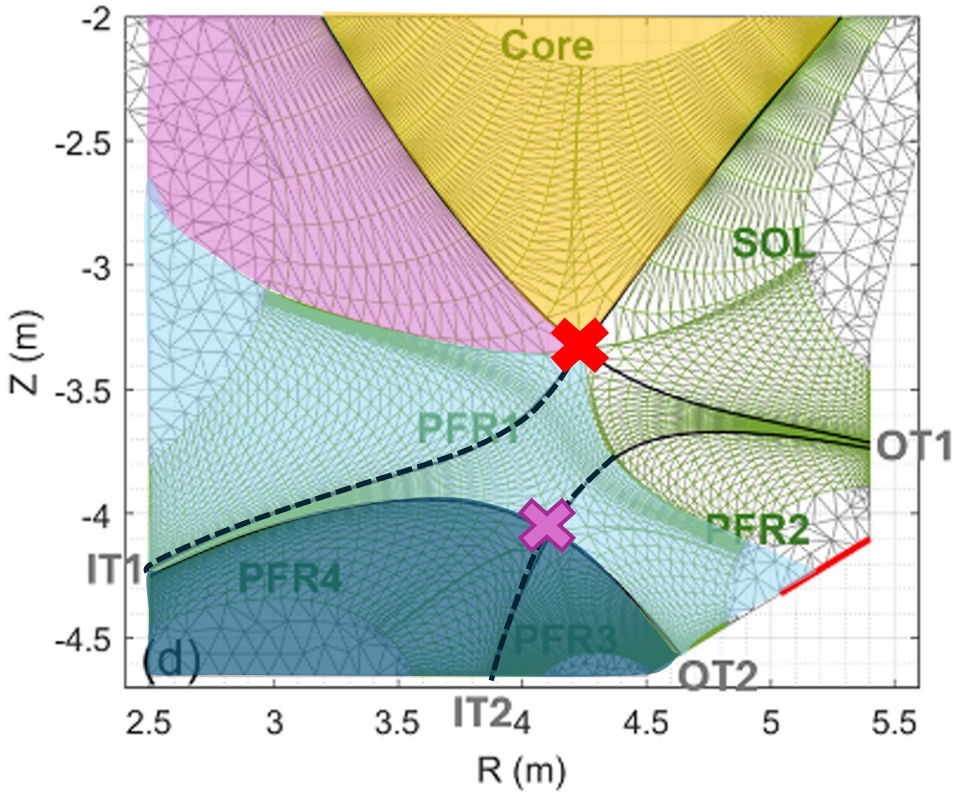
HFS SF+



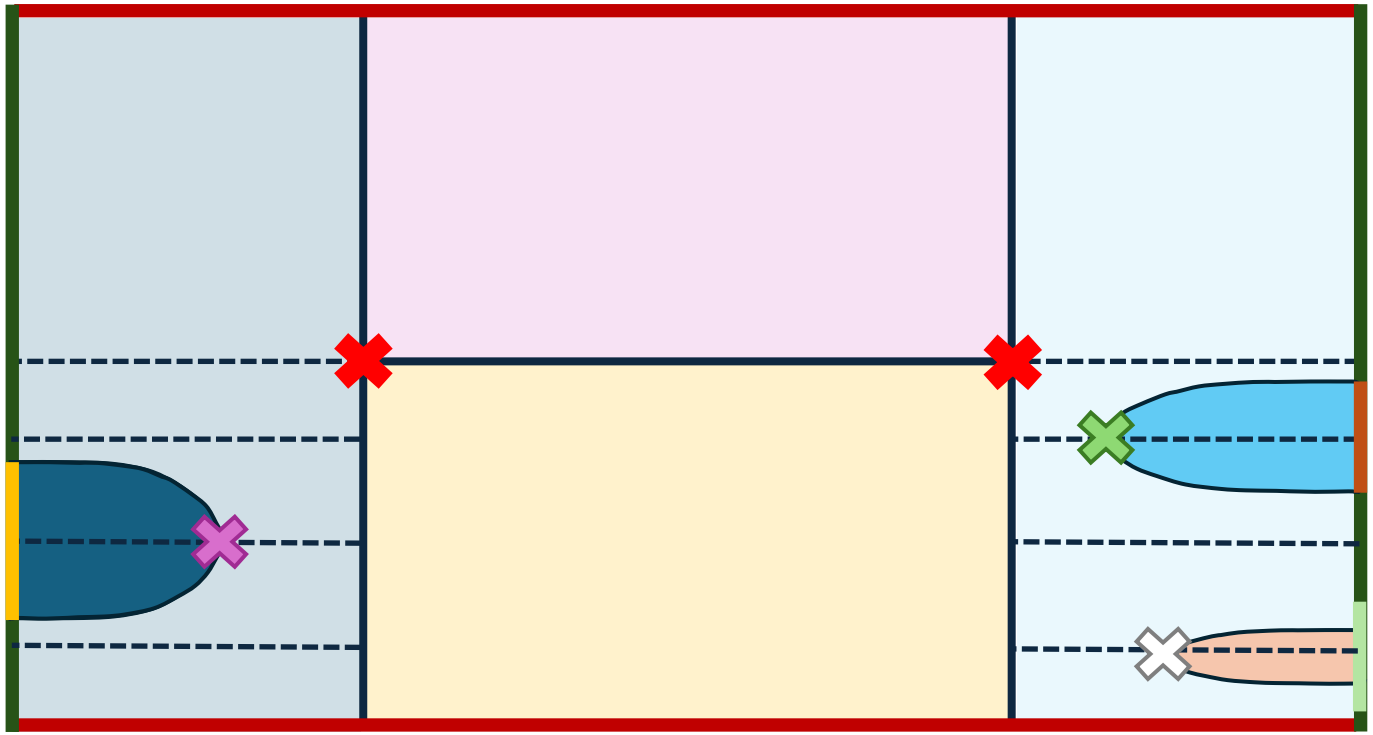

Core
SOL
PFR1
PFR2
PFR3
PFR4
OT1
OT2
IT1
IT2
(d)
Z (m)
R (m)
-2
-2.5
-3
-3.5
-4
-4.5
2.5
3
3.5
4
4.5
5
5.5

HFS SF+

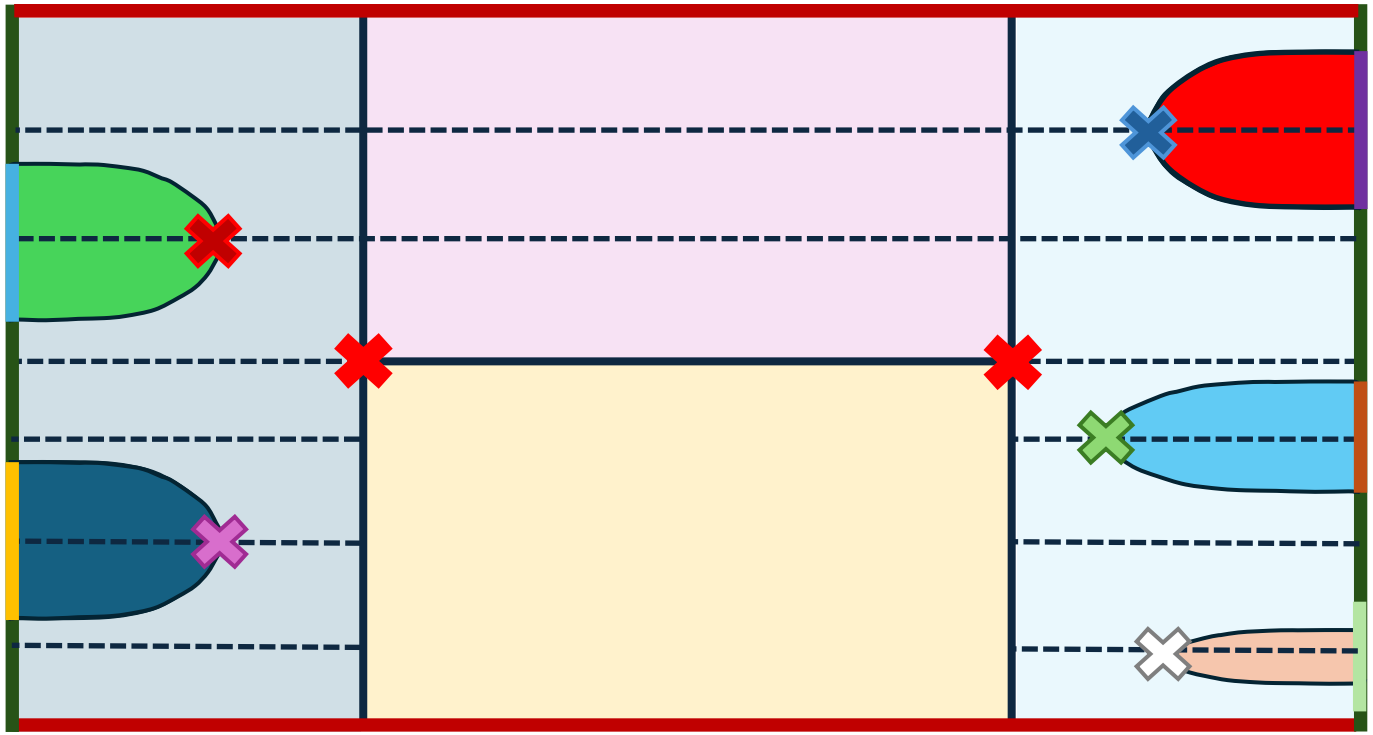



Core
SOL
PFR1
PFR2
PFR3
PFR4
OT1
OT2
IT1
IT2
(d)
-2
-2.5
-3
-3.5
-4
-4.5
Z (m)
2.5
3
3.5
4
4.5
5
5.5
R (m)

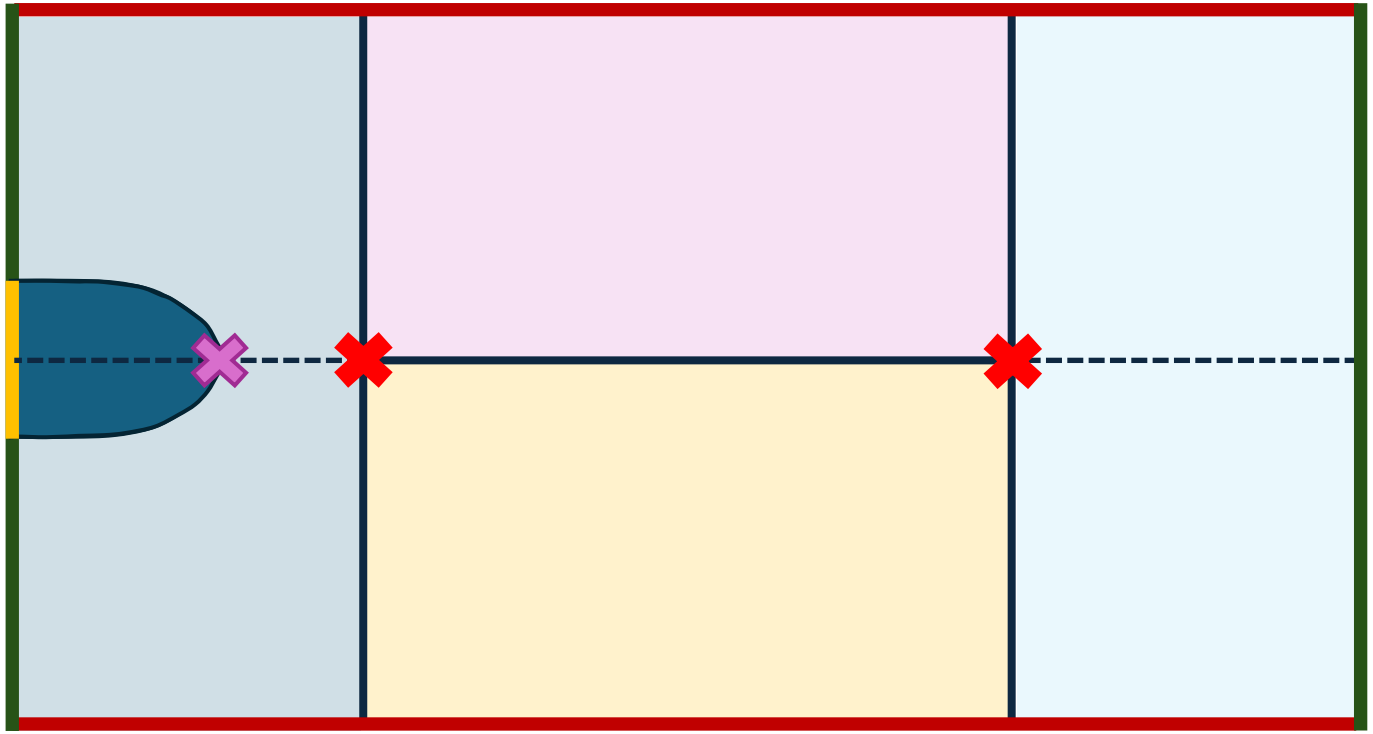
Here as well all possible combinations...



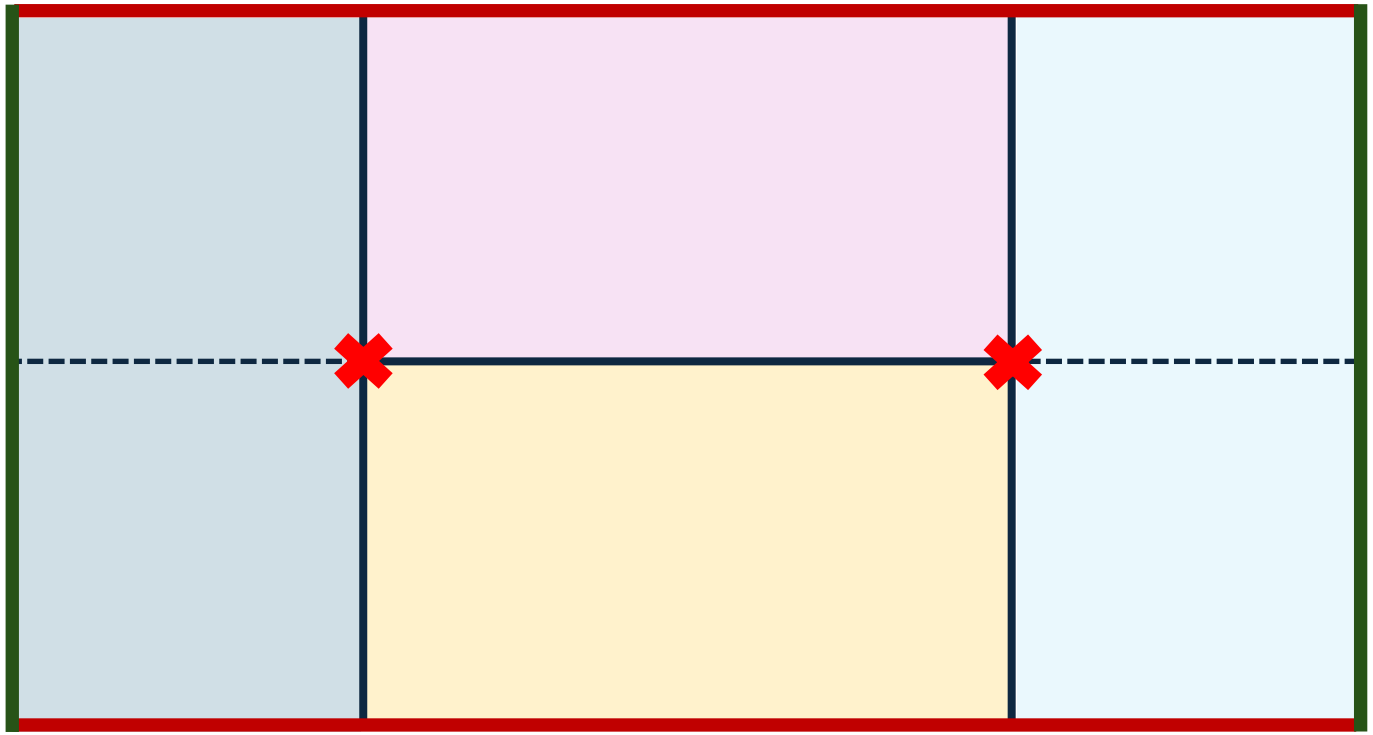
… of course also with SF-!



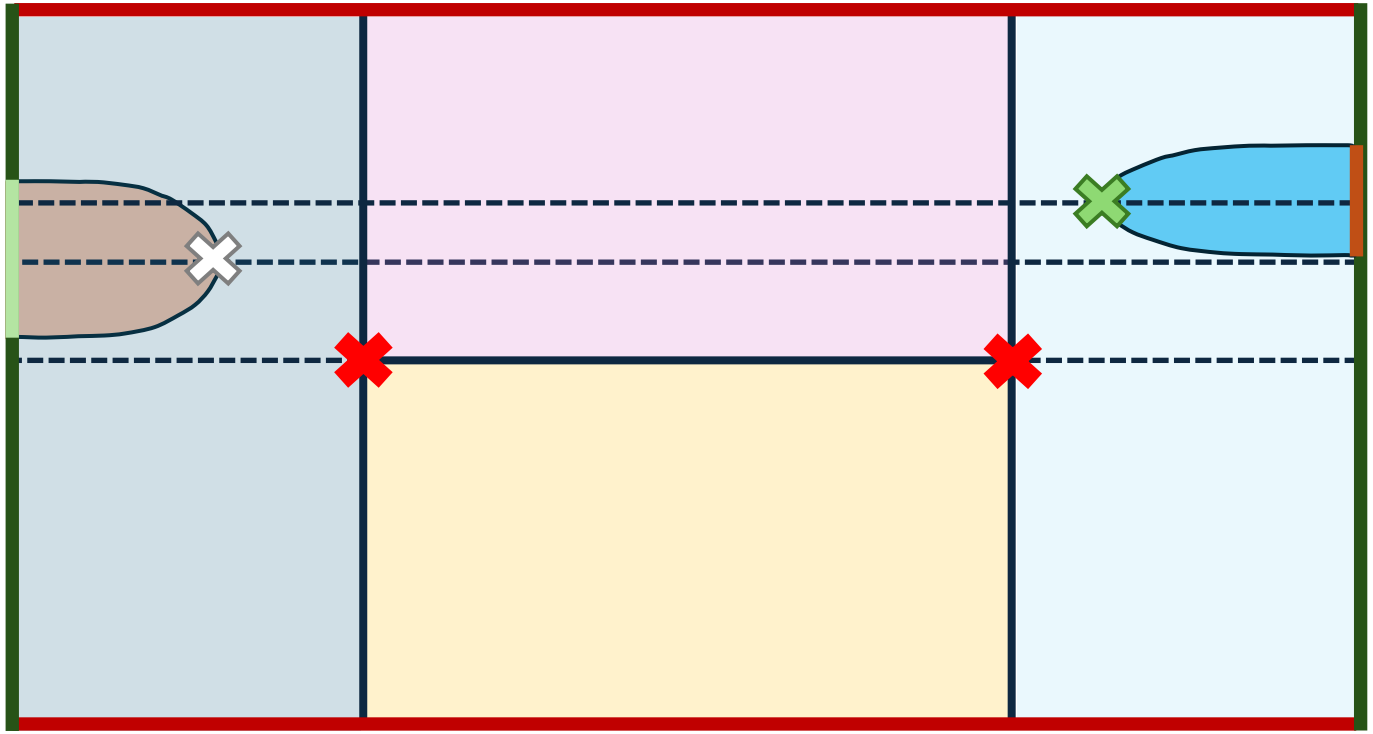
Special case: SF0 (additional X-point on same flux tube of primary separatrix, on divertor leg (max one per divertor)



An arbitrary number of additional regions can be detected in such a way in post-processing phase of Goat, depending on where X-point are located

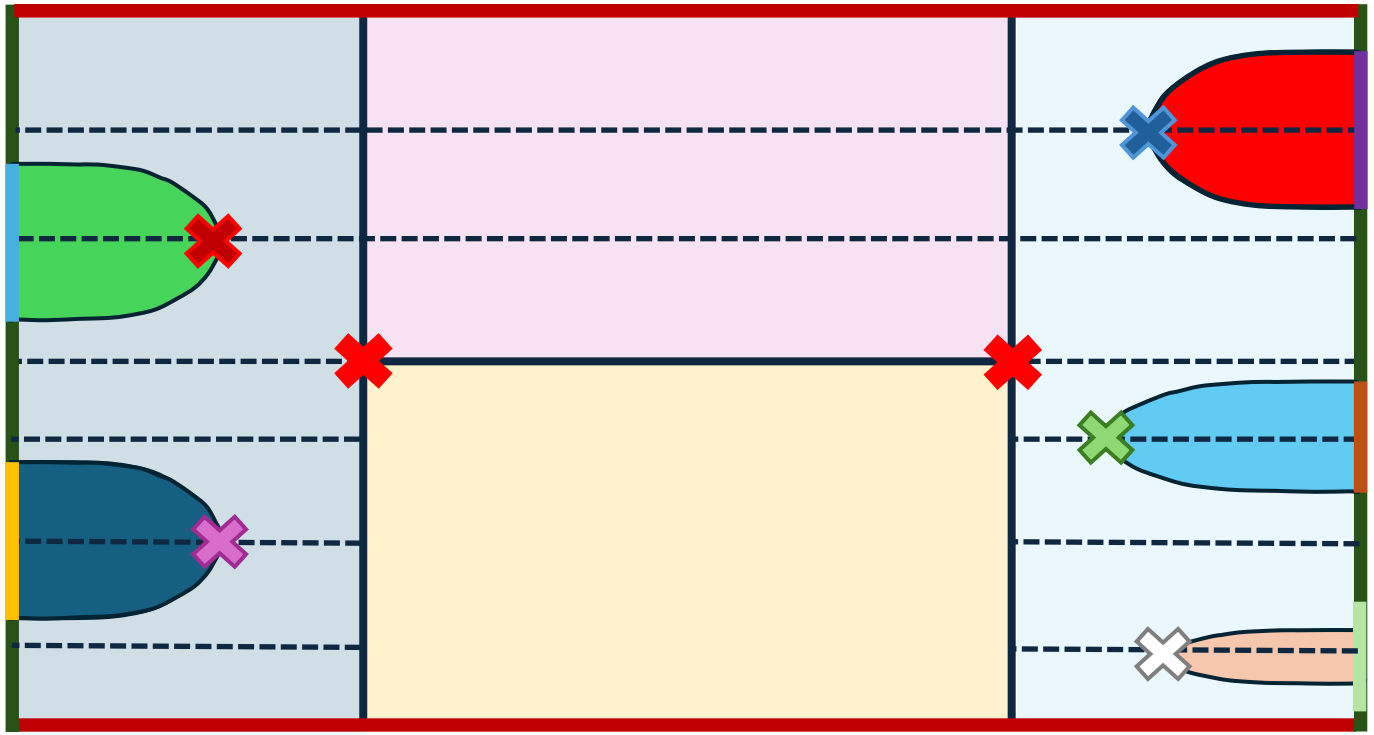


The detection algorithm would be robust, because this naturally **cannot** happen!
Otherwise another X-point would have been there, and Goat would have detected it!

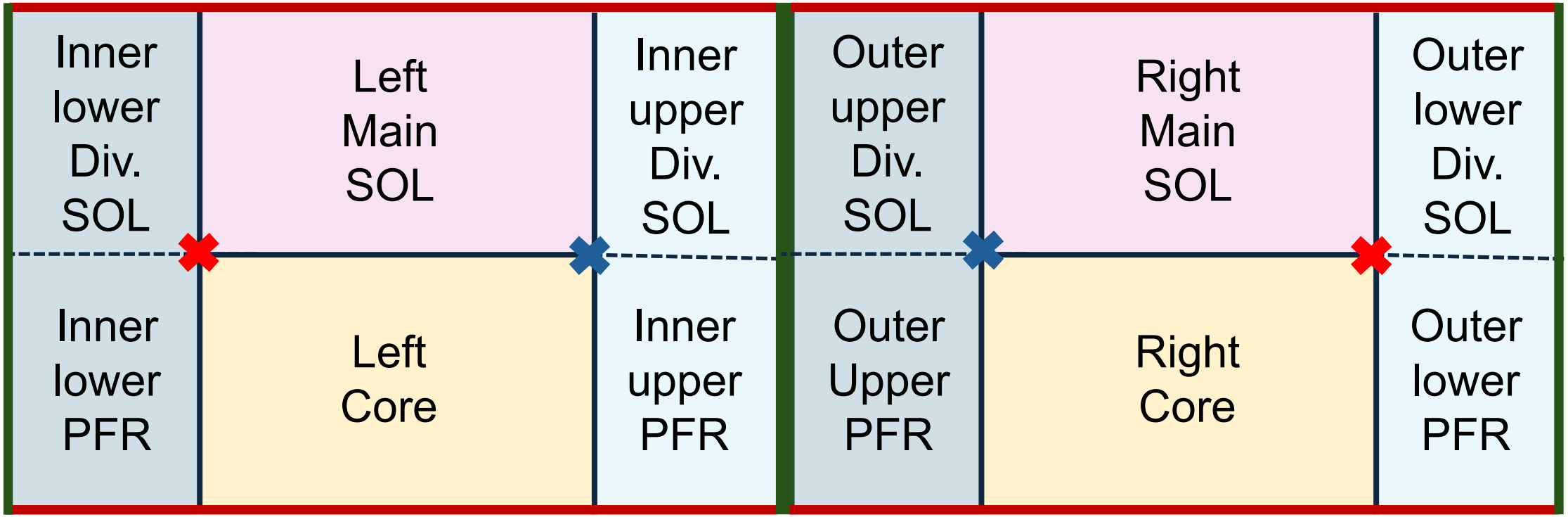


Additional X-points can be connected or disconnected from each other, but for a complete characterization is only sufficient to know:

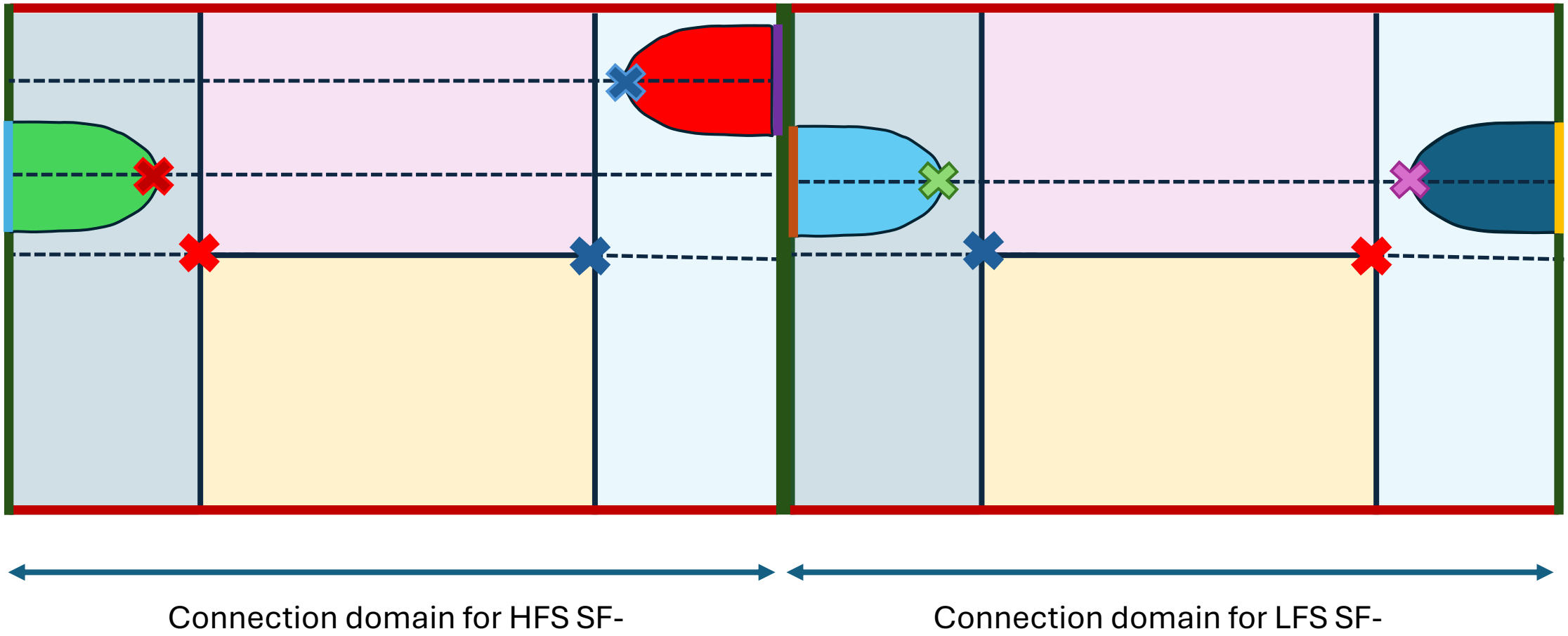
- How many additional X-points/PFR are there **on each divertor legs**
- **On which flux tube**(s) the relative X-point(s) are located



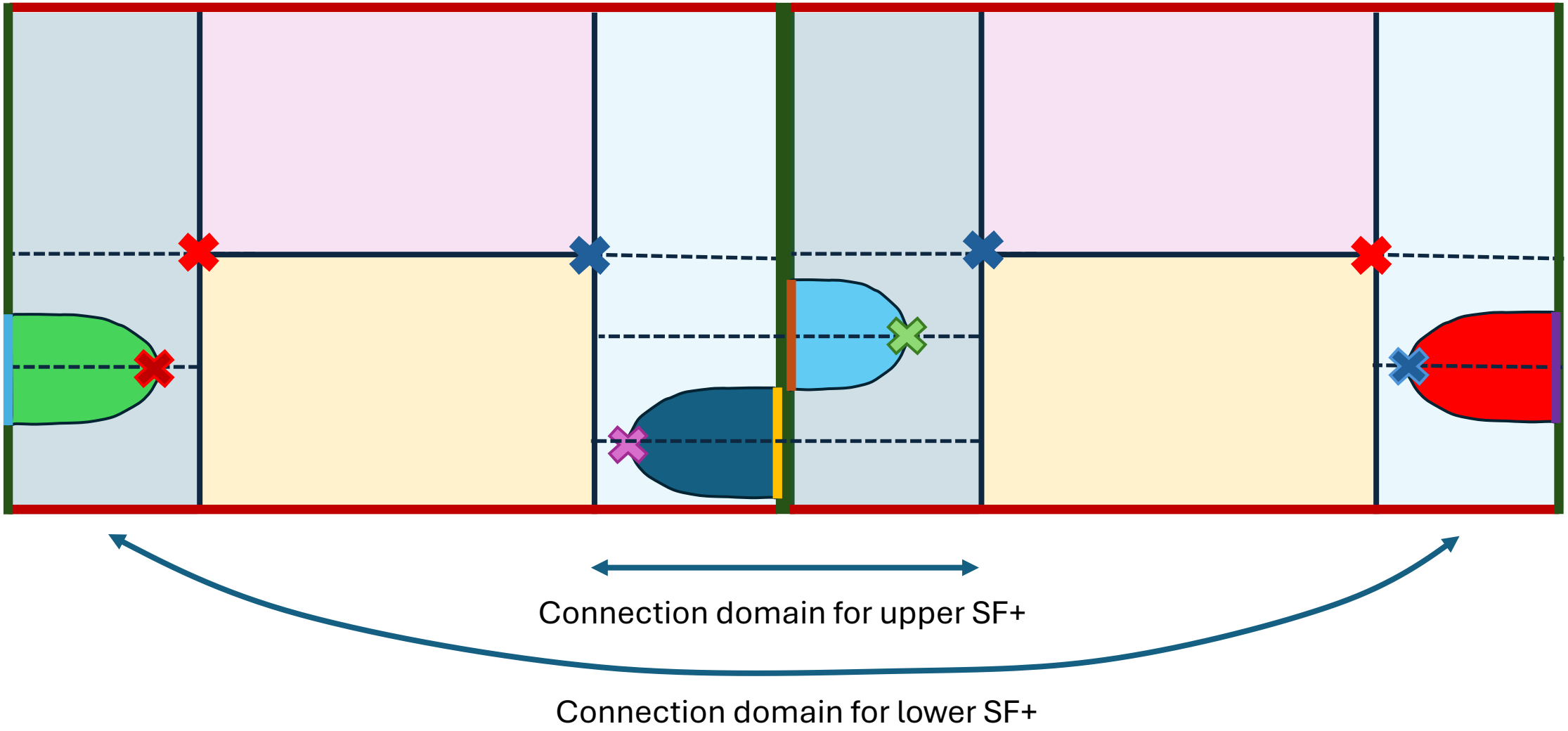
→ Natural generalization to connected double nulls…



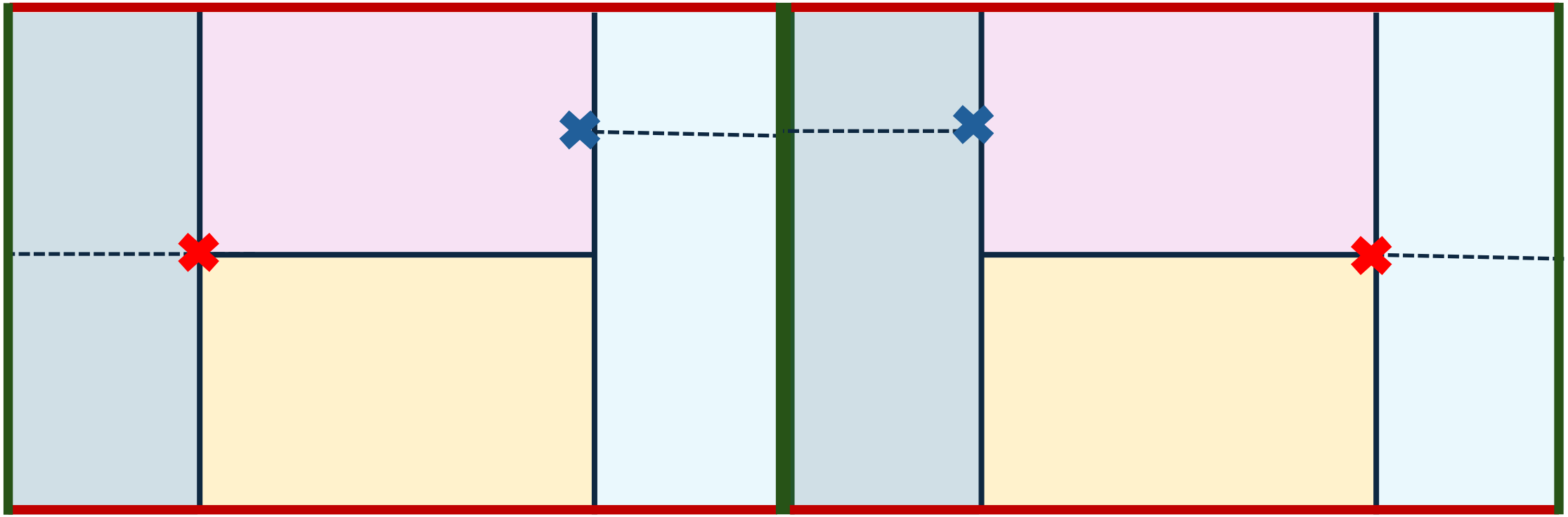
→ Natural generalization to connected double nulls…



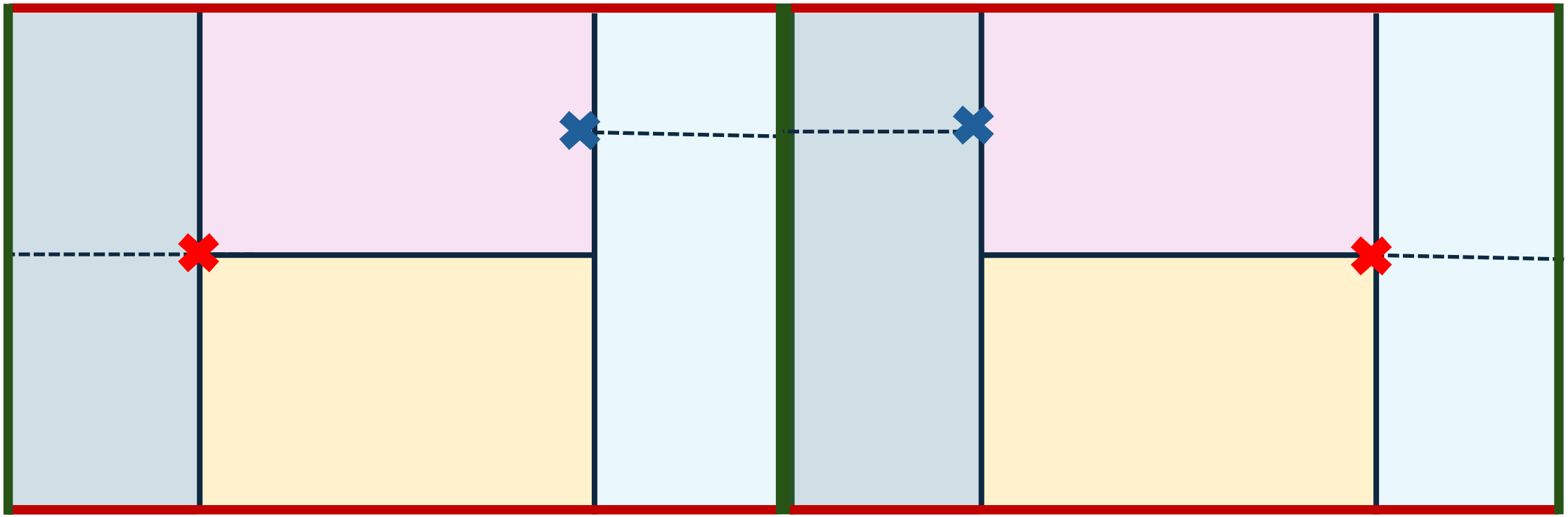
→ Natural generalization to connected double nulls…



… and disconnected double nulls



… and disconnected double nulls

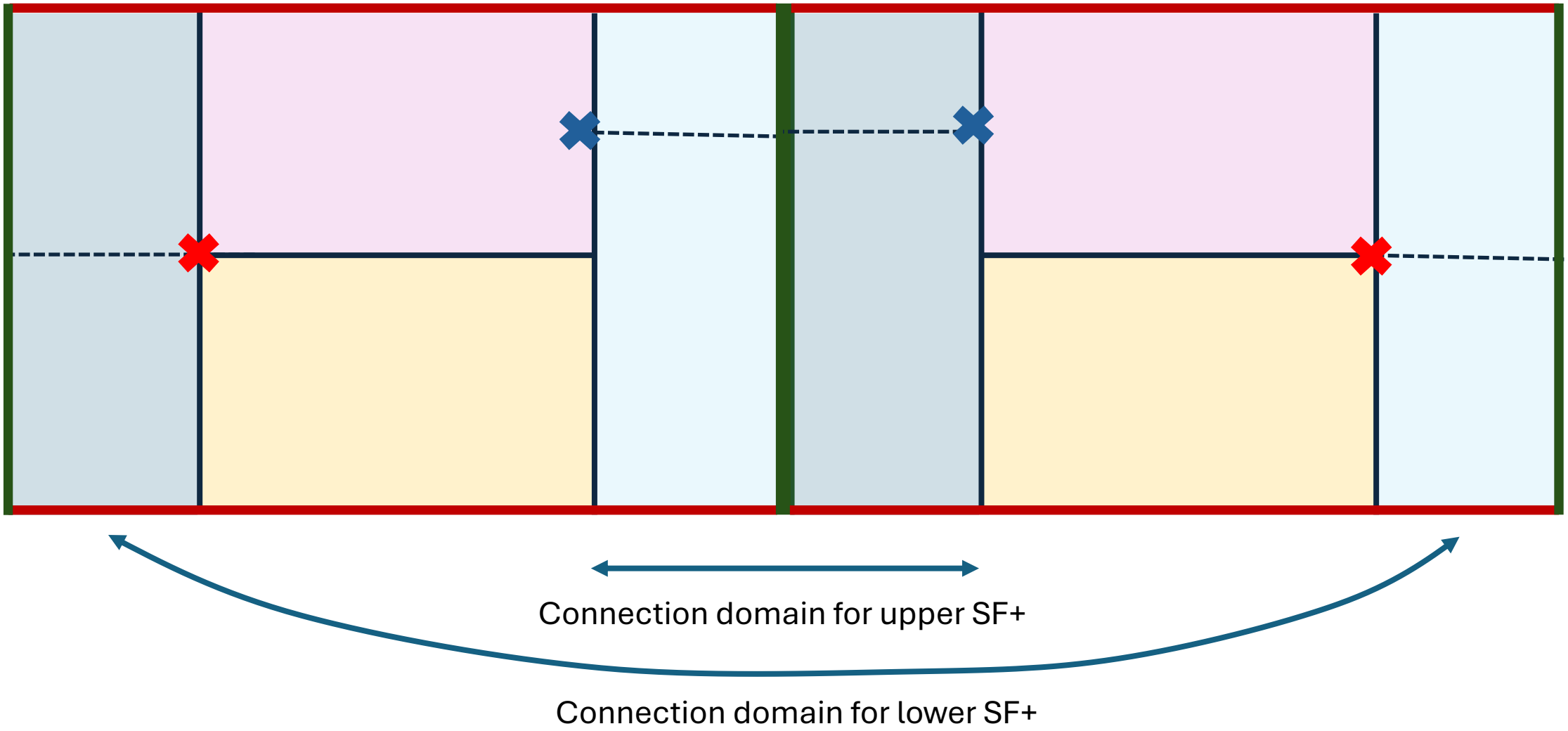


Connection domain for HFS/LFS SF- with X-point between primary and secondary separatrices

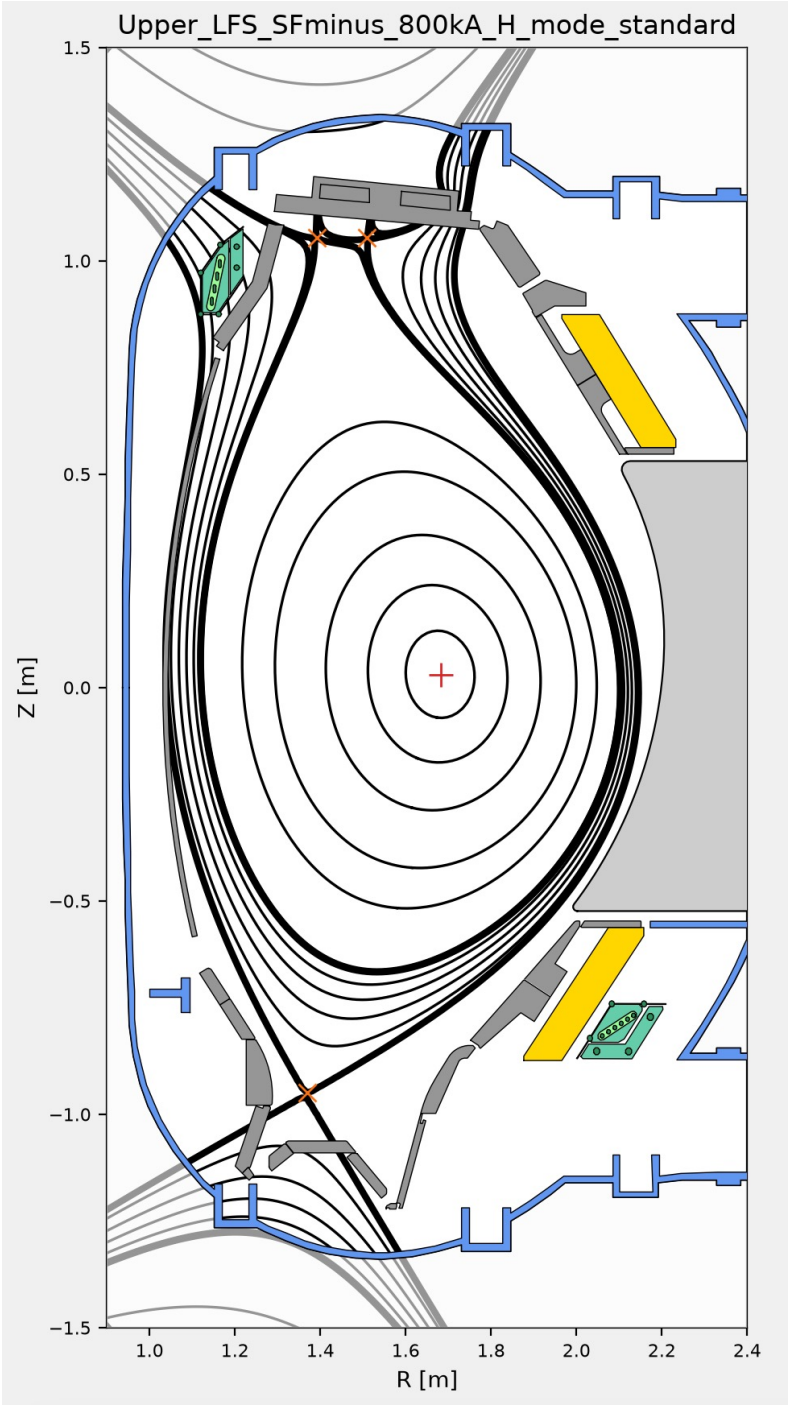
Connection domain for HFS SF- with X-point beyond the secondary separatrix

Connection domain for LFS SF- with X-point beyond the secondary separatrix

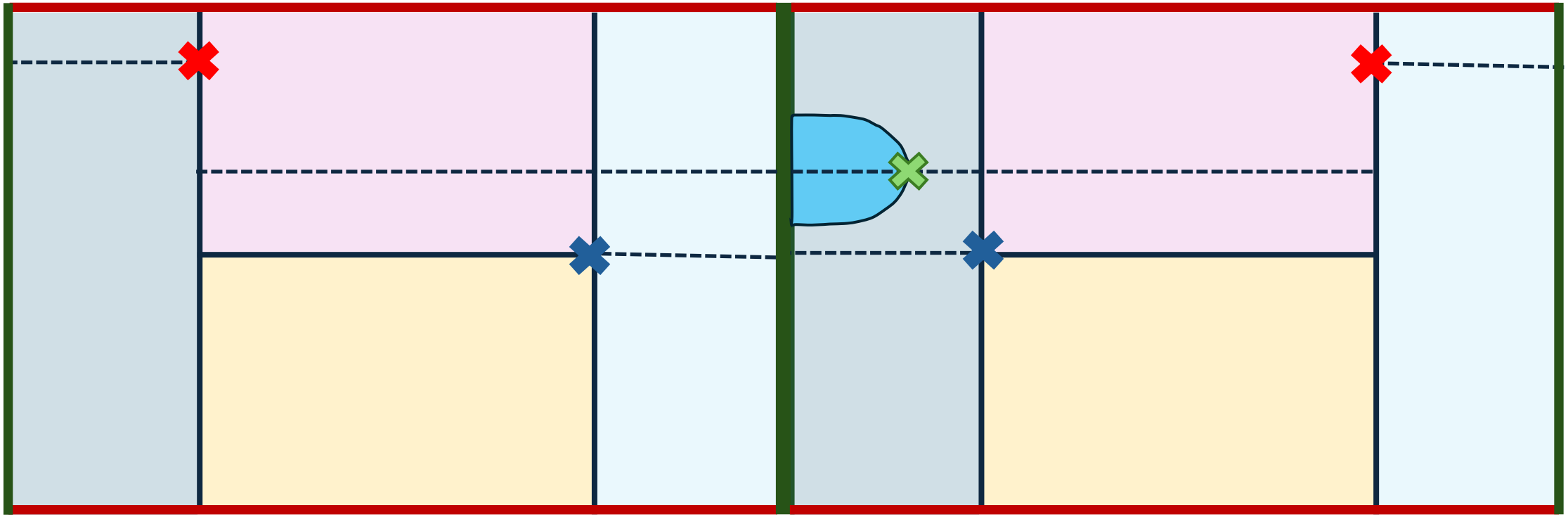
… and disconnected double nulls



Example: upper LFS SF- with additional X-point in lower divertor



How this would appear



Proposed algorithm:

- Check whether we have only 1 O-points

Proposed algorithm:

- Check whether we have only 1 O-points
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.

Proposed algorithm:

- Check whether we have only 1 O-points
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.
- For each side (lower and upper), check the flux surface of the innermost X-point.
  Based on the answer and on which one sits at the primary separatrix, we define the basic configuration:
  SN (lower/upper), CDN, DDN (bottom/top)

Proposed algorithm:

- Check whether we have only 1 O-points
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.
- For each side (lower and upper), check the flux surface of the innermost X-point.
  Based on the answer and on which one sits at the primary separatrix, we define the basic configuration:
  SN (lower/upper), CDN, DDN (bottom/top)
- At the beginning, use the same hard-coded numbering and ordering of volume and face regions as in the classic SOLPS convention, disregarding any additional X-point per side.

Proposed algorithm:

- Check whether we have only 1 O-points
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.
- For each side (lower and upper), check the flux surface of the innermost X-point.
  Based on the answer and on which one sits at the primary separatrix, we define the basic configuration:
  SN (lower/upper), CDN, DDN (bottom/top)
- At the beginning, use the same hard-coded numbering and ordering of volume and face regions as in the classic SOLPS convention, disregarding any additional X-point per side.
- If there is more than 1 X-point at either lower or upper sides:
  - Add an additional volume region for any additional PFR, starting from the end of max(NREG), and overwrite the previous volume region number in the volume cells where this PFR exists

Proposed algorithm:

- Check whether we have only 1 O-points.
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.
- For each side (lower and upper), check the flux surface of the innermost X-point.
  Based on the answer and on which one sits at the primary separatrix, we define the basic configuration:
  SN (lower/upper), CDN, DDN (bottom/top)
- At the beginning, use the same hard-coded numbering and ordering of volume and face regions as in the classic SOLPS convention, disregarding any additional X-point per side.
- If there is more than 1 X-point at either lower or upper sides:
  - Add an additional volume region for any additional PFR, starting from the end of max(NREG), and overwrite the previous volume region number in the volume cells where this PFR exists
- Follow a precise order: e.g. fill, from first to last PFR, new regions with:
  - If the basic topology was SN: first fill with all new PFRs in left leg, then fill with all new PFRs in right leg
  - If it was CDN/DDN: use bottom left leg, top left leg, top right leg, bottom right leg as order.
  And in any case with radial direction on increasing rhop.

Proposed algorithm:

- Check whether we have only 1 O-points
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.
- For each side (lower and upper), check the flux surface of the innermost X-point.
  Based on the answer and on which one sits at the primary separatrix, we define the basic configuration:
  SN (lower/upper), CDN, DDN (bottom/top)
- At the beginning, use the same hard-coded numbering and ordering of volume and face regions as in the classic SOLPS convention, disregarding any additional X-point per side.
- If there is more than 1 X-point at either lower or upper sides:
  - Add an additional volume region for any additional PFR, starting from the end of max(NREG), and overwrite the previous volume region number in the volume cells where this PFR exists
- Follow a precise order: e.g. fill, from first to last PFR, new regions with:
  - If the basic topology was SN: first fill with all new PFRs in left leg, then fill with all new PFRs in right leg
  - If it was CDN/DDN: use bottom left leg, top left leg, top right leg, bottom right leg as order.
  And in any case with radial direction on increasing rhop.
- Goat delivers to B2.5, via traduit, general topology information, which will contain how many additional X-points are there per side, on which flux tubes are they, and in which of the 2 or 4 divertor legs.

Proposed algorithm:

- Check whether we have only 1 O-points
- Check whether we have any X-point in the lower and in the upper side of the plasma. If not, then limiter.
- For each side (lower and upper), check the flux surface of the innermost X-point.
  Based on the answer and on which one sits at the primary separatrix, we define the basic configuration:
  SN (lower/upper), CDN, DDN (bottom/top).
- At the beginning, use the same hard-coded numbering and ordering of volume and face regions as in the classic SOLPS convention, disregarding any additional X-point per side.
- If there is more than 1 X-point at either lower or upper sides:
  - Add an additional volume region for any additional PFR, starting from the end of max(NREG), and overwrite the previous volume region number in the volume cells where this PFR exists
- Follow a precise order: e.g. fill, from first to last PFR, new regions with:
  - If the basic topology was SN: first fill with all new PFRs in left leg, then fill with all new PFRs in right leg
  - If it was CDN/DDN: use bottom left leg, top left leg, top right leg, bottom right leg as order.
  And in any case with radial direction on increasing rhop.
- Goat delivers to B2.5, via traduit, general topology information, which will contain how many additional X-points are there per side, on which flux tubes are they, and in which of the 2 or 4 divertor legs.
- B2.5 has all information to link a topology with arbitrary number of additional X-points with a deterministic numbering and ordering of volume regions

Basic requirement:

Goat needs to be able to correctly classify volume and face regions in a B2.5 compatible way at least for SN, CDN, DDN

Basic requirement:

Goat needs to be able to correctly classify volume and face regions in a B2.5 compatible way at least for SN, CDN, DDN

Strategy I am attempting (forked branch `antonellozito:azito` of Goat repo):

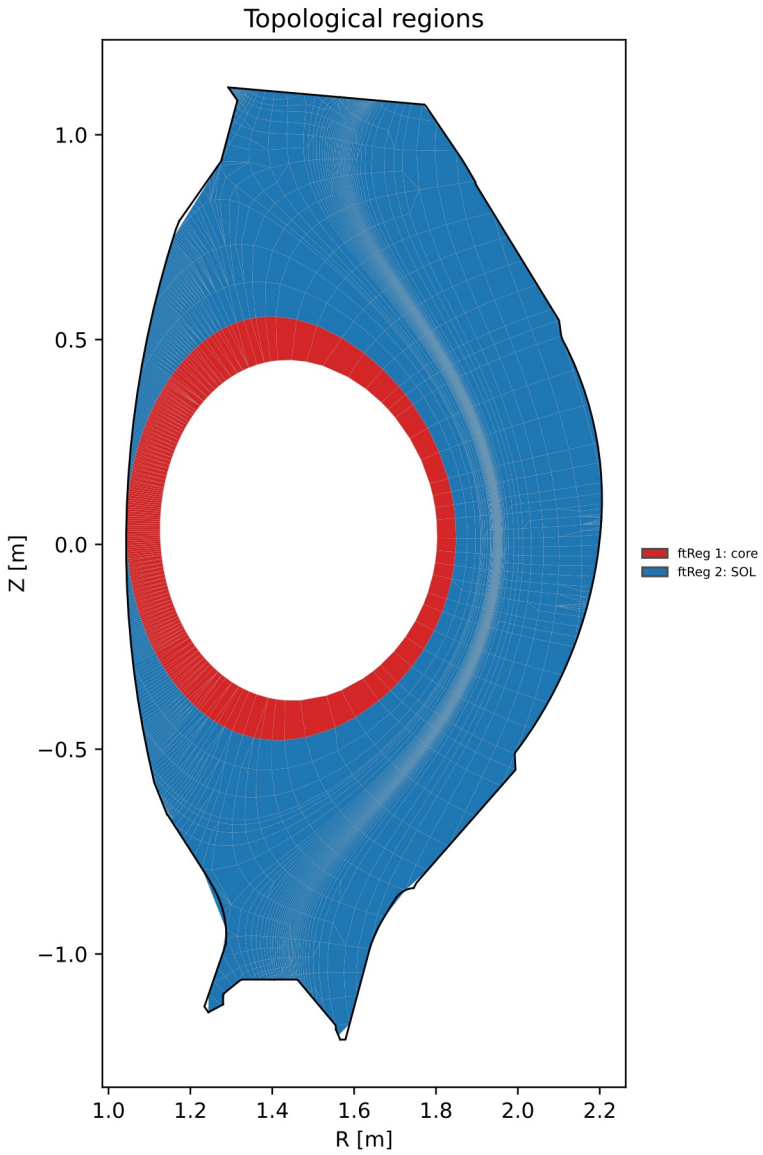
Basic requirement:
Goat needs to be able to correctly classify volume and face regions in a B2.5 compatible way at least for SN, CDN, DDN

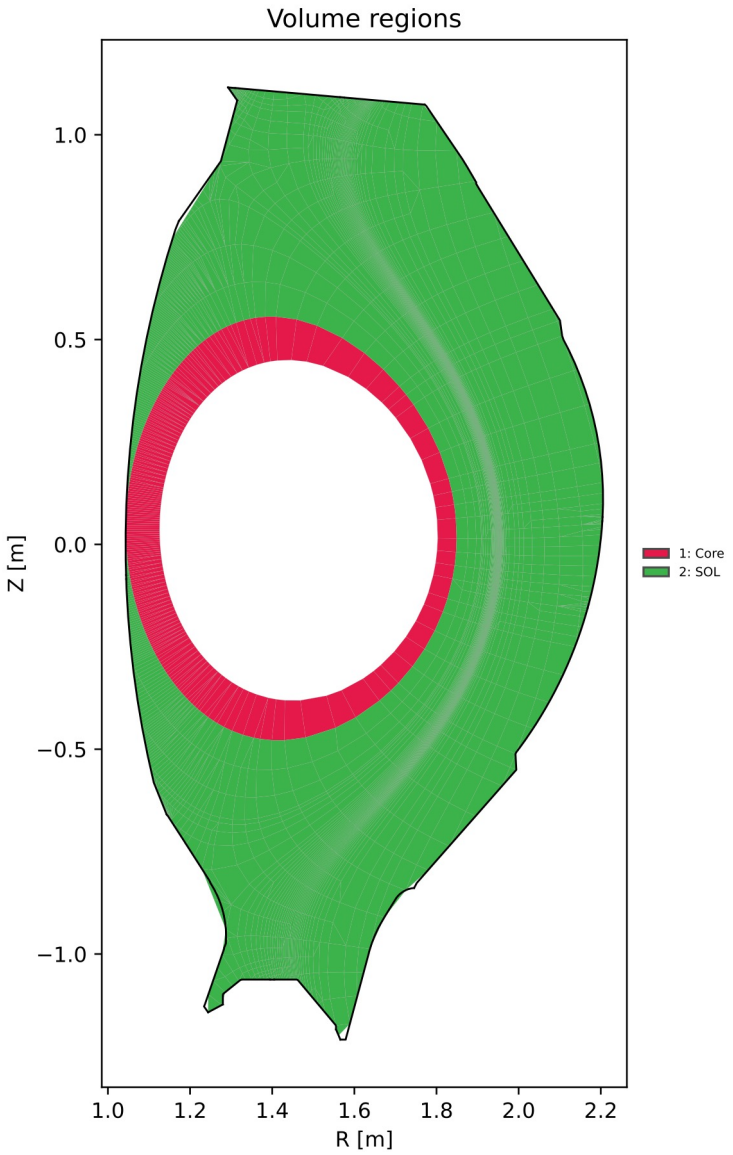
Strategy I am attempting (Goat fork `antonellozito: SOLPS_regions_compatibility`):

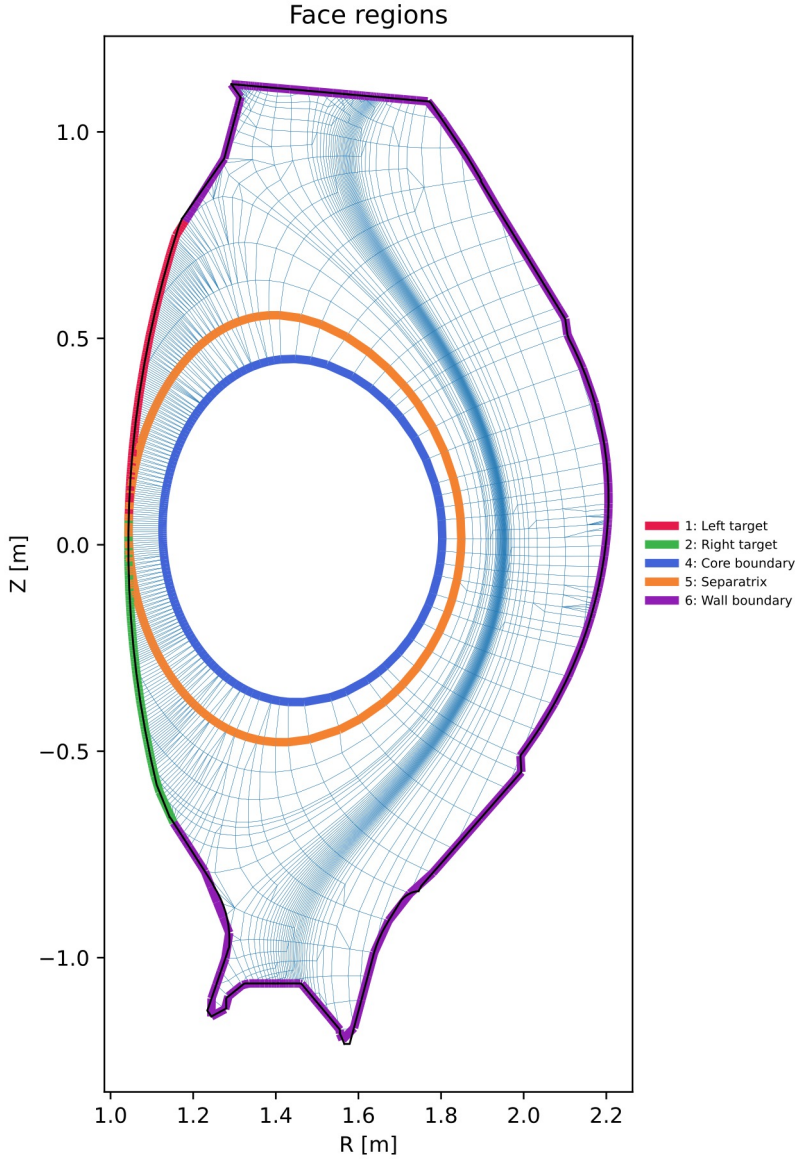
New switches:

- `goat.vessel.solps_limiter_target`
  → List of adjacent surface segments, from structure.dat, to be defined as target surface for a limiter geometry
  It is checked that, if the detected geometry is a limiter, then this list MUST be non empty, and the tangency point between limiter and wall contour MUST be on the surface formed in this way, otherwise error

Result (limiter)







Basic requirement:
Goat needs to be able to correctly classify volume and face regions in a B2.5 compatible way at least for SN, CDN, DDN

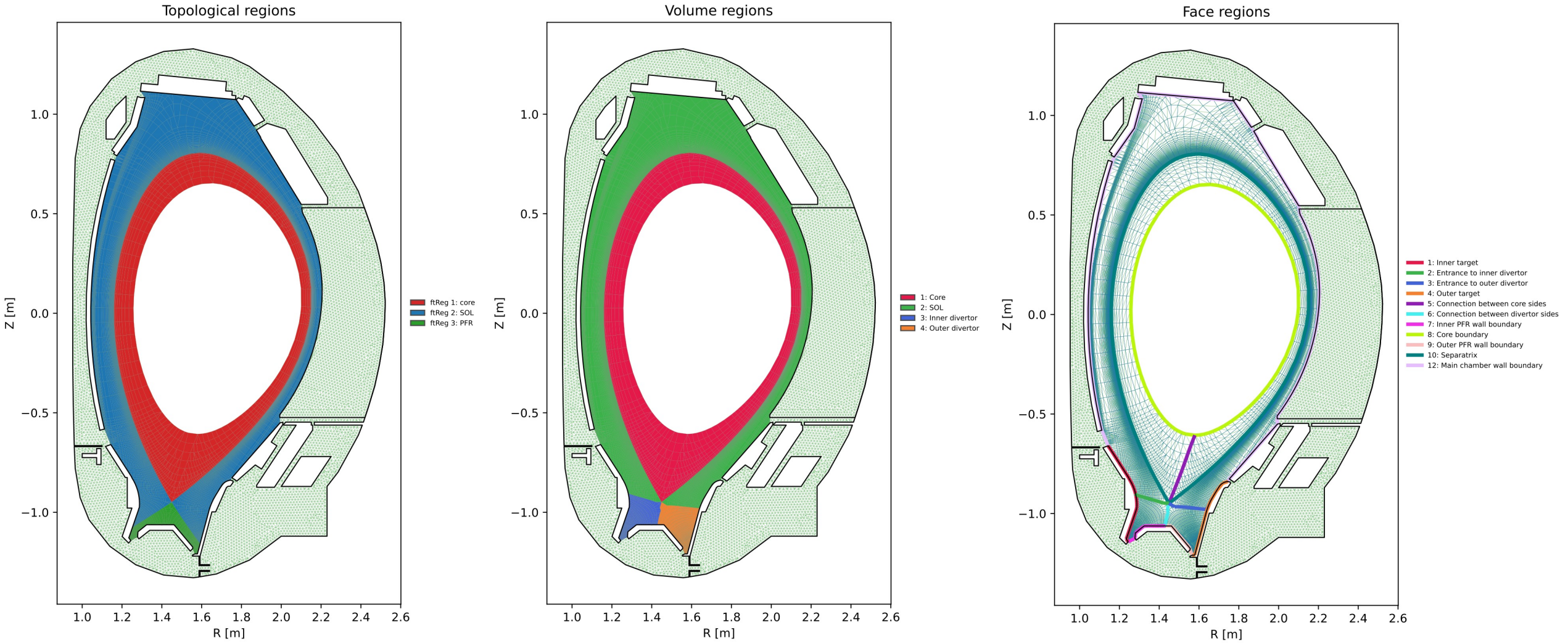
Strategy I am attempting (Goat fork `antonellozito: SOLPS_regions_compatibility`):

New switches:

- `goat.vessel.solps_lower_divertor_inner_target`
- `goat.vessel.solps_lower_divertor_outer_target`

  → List of adjacent surface segments, from structure.dat, to be defined as inner and outer target surfaces, respectively, if there is AT LEAST ONE X-point in the lower divertor (below the O-point).
  It is checked that, if the detected geometry has at least one lower X-point, then both these lists MUST be non empty, and the strikepoints of each separatrix leg (even from additional X-points from each divertor leg) MUST be on the corresponding surfaces formed in this way, otherwise error

Result (lower single null)



Basic requirement:
Goat needs to be able to correctly classify volume and face regions in a B2.5 compatible way at least for SN, CDN, DDN

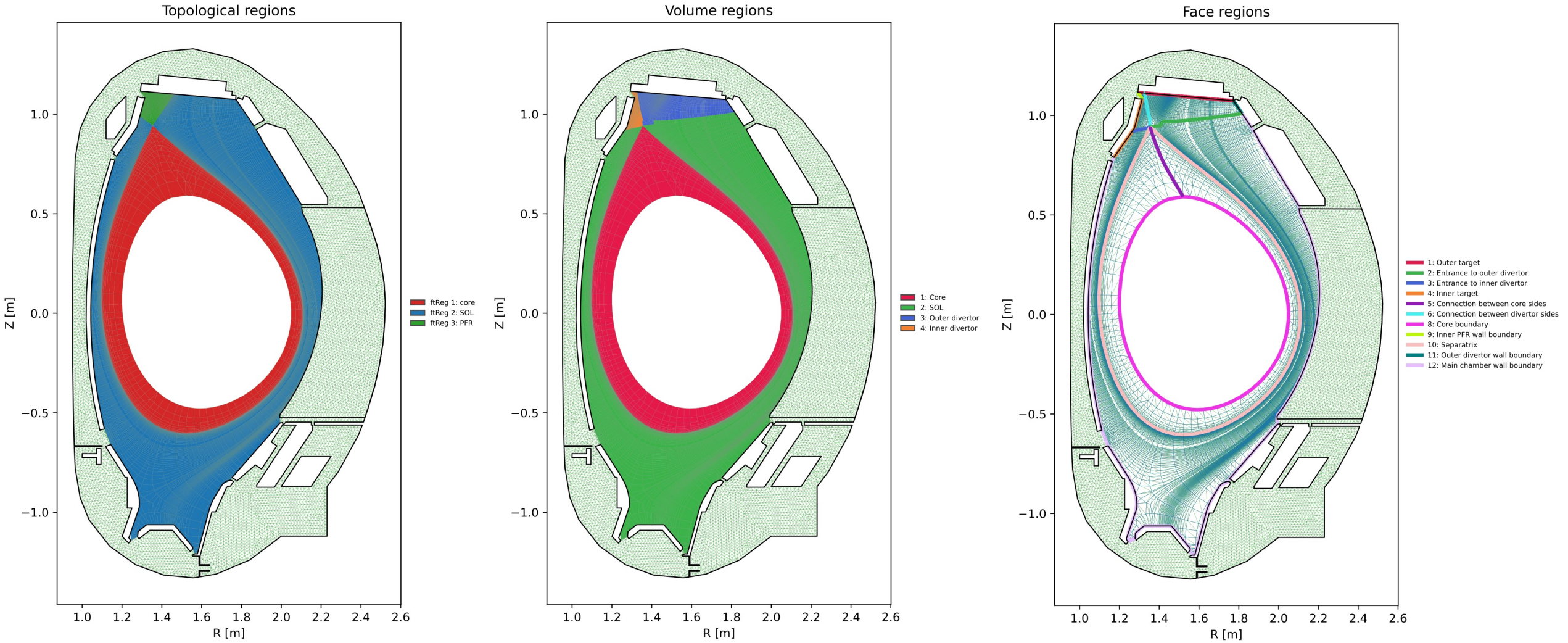
Strategy I am attempting (Goat fork `antonellozito: SOLPS_regions_compatibility`):

New switches:

- `goat.vessel.solps_upper_divertor_inner_target`
- `goat.vessel.solps_upper_divertor_outer_target`

  → List of adjacent surface segments, from structure.dat, to be defined as inner and outer target surfaces, respectively, if there is AT LEAST ONE X-point in the lower divertor (below the O-point).
  It is checked that, if the detected geometry has at least one lower X-point, then both these lists MUST be non empty, and the strikepoints of each separatrix leg (even from additional X-points from each divertor leg) MUST be on the corresponding surfaces formed in this way, otherwise error

Result (upper single null)



Basic requirement:
Goat needs to be able to correctly classify volume and face regions in a B2.5 compatible way at least for SN, CDN, DDN

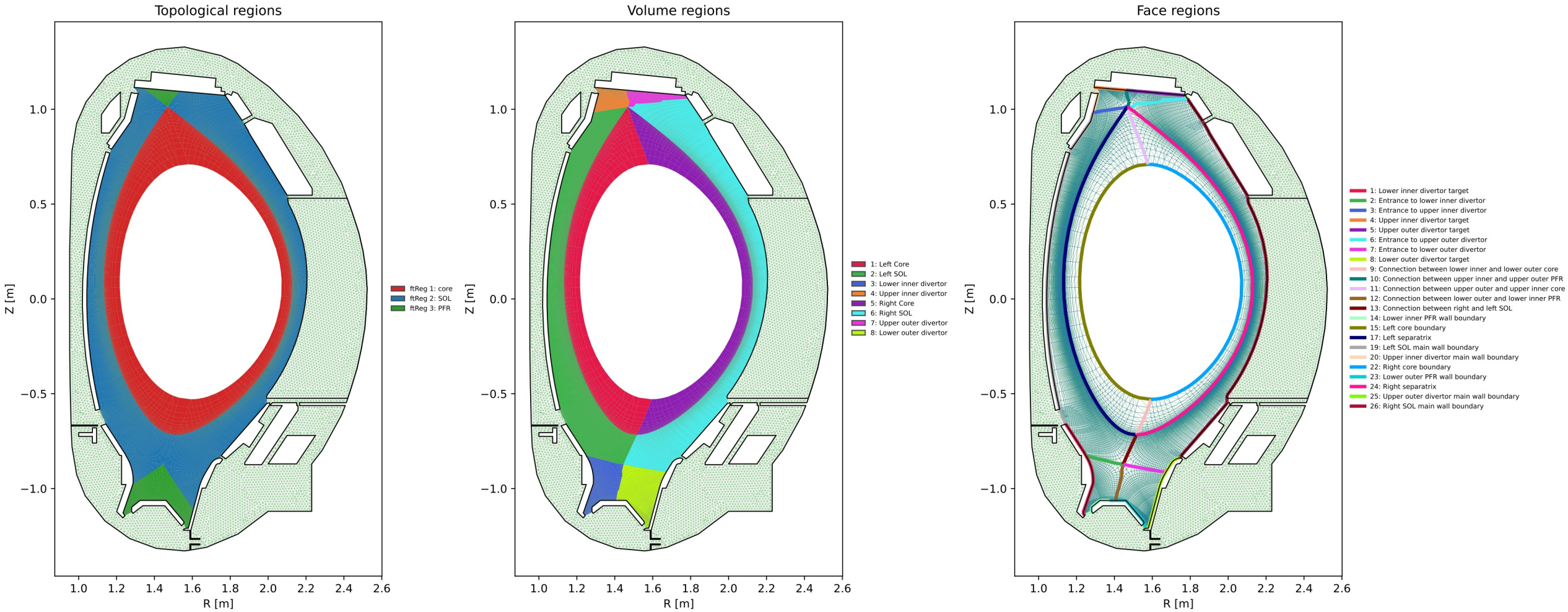
Strategy I am attempting (Goat fork `antonellozito: SOLPS_regions_compatibility`):

If there is at least one X-point at both lower and upper side of the O-point, then

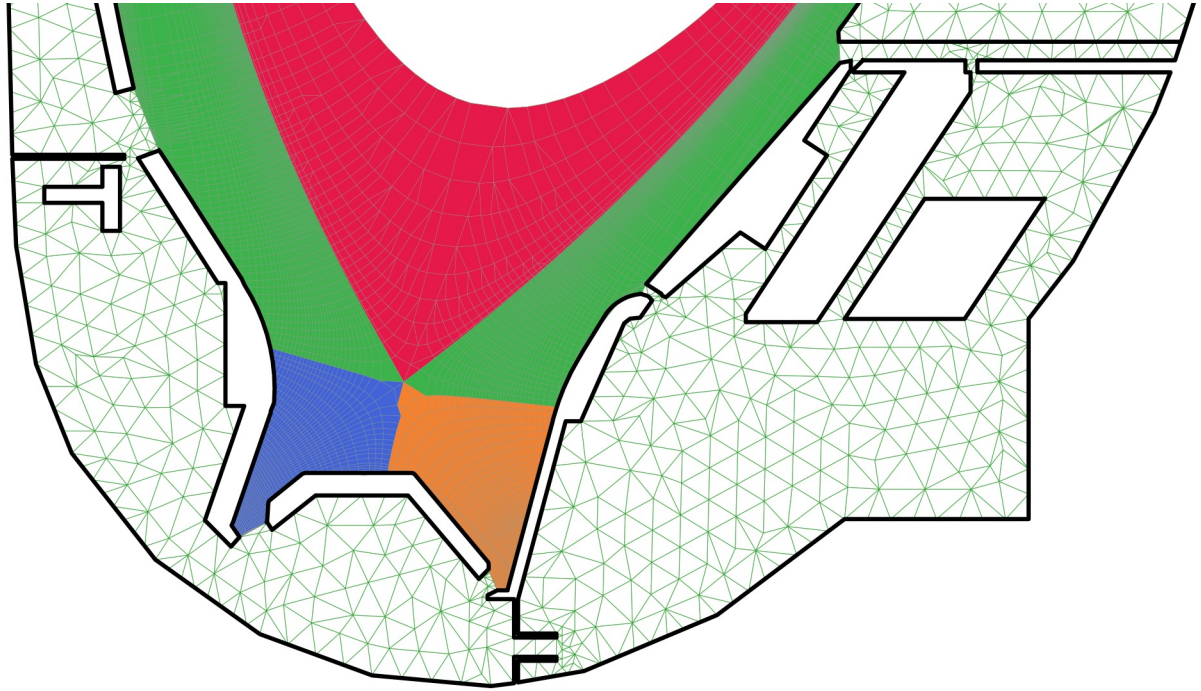
- `goat.vessel.solps_lower_divertor_inner_target`
- `goat.vessel.solps_lower_divertor_outer_target`
- `goat.vessel.solps_upper_divertor_inner_target`
- `goat.vessel.solps_upper_divertor_outer_target`

  → Must all be non empty

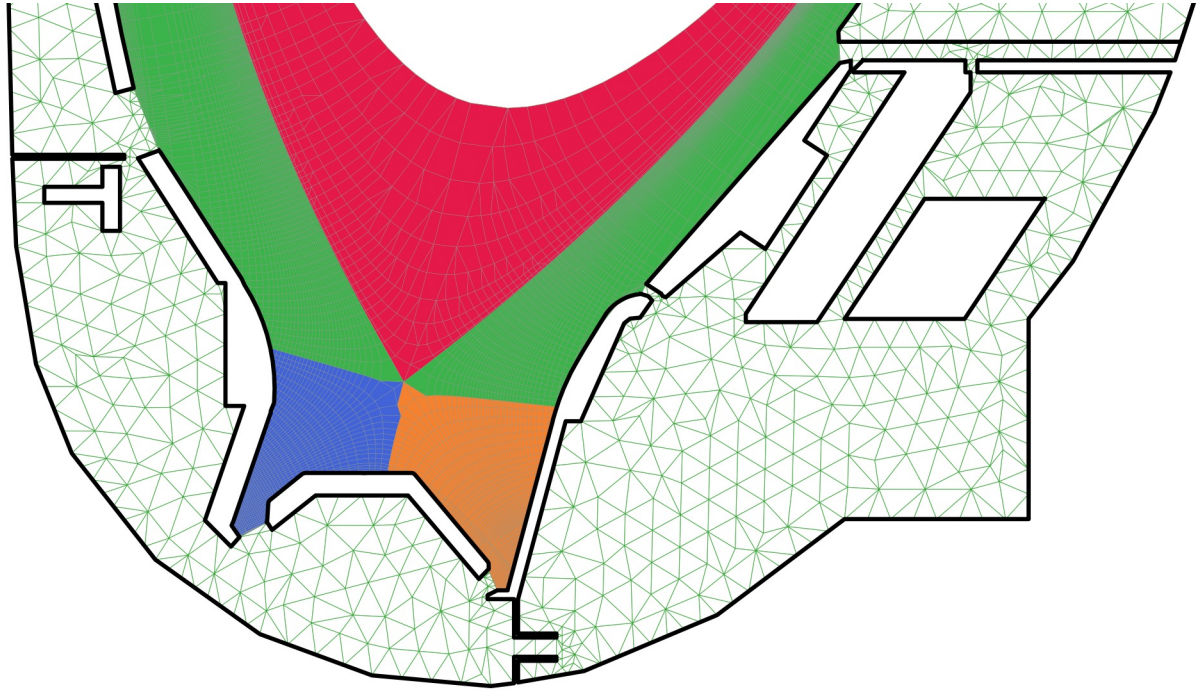
Result (double null)



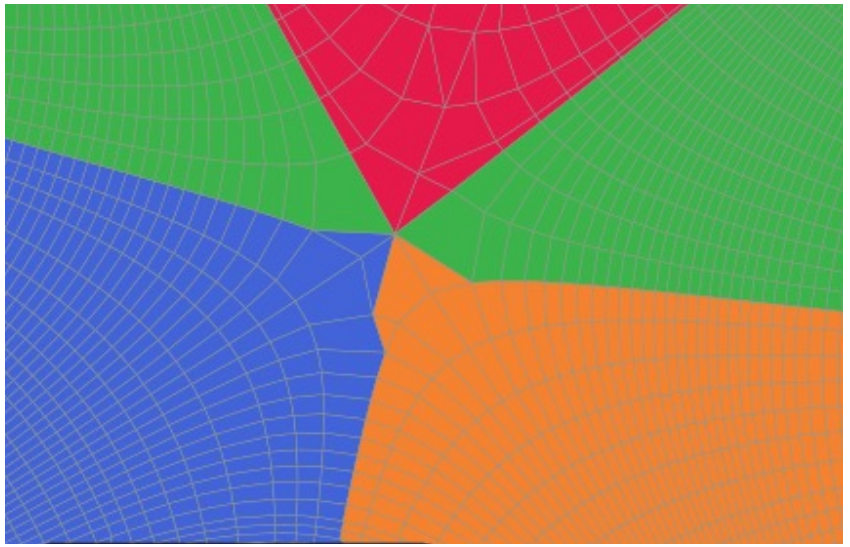
Boundary between main SOL and divertor SOL and between inner/outer PFR branches



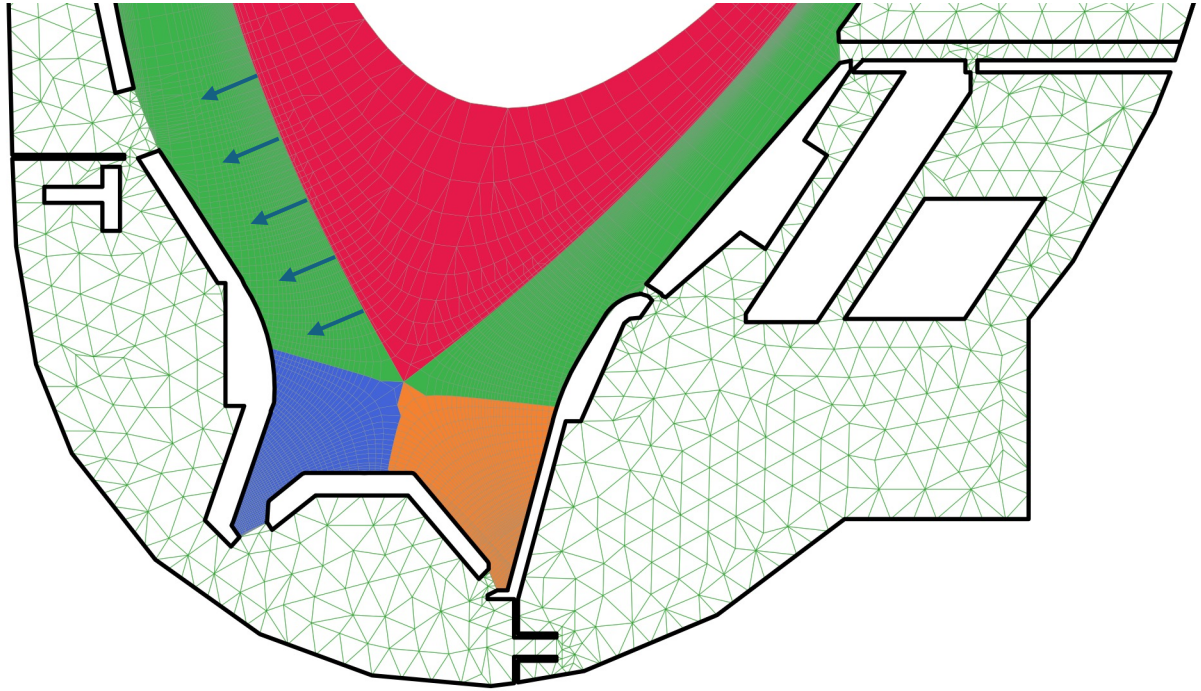
Boundary between main SOL and divertor SOL and between inner/outer PFR branches



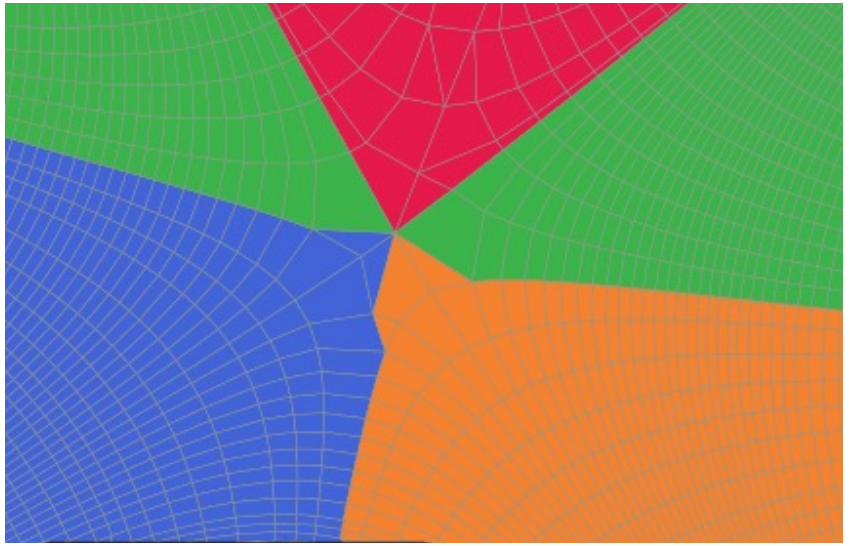
Identify ”rays” moving radially outwards from X-point until a wall is found



Boundary between main SOL and divertor SOL and between inner/outer PFR branches

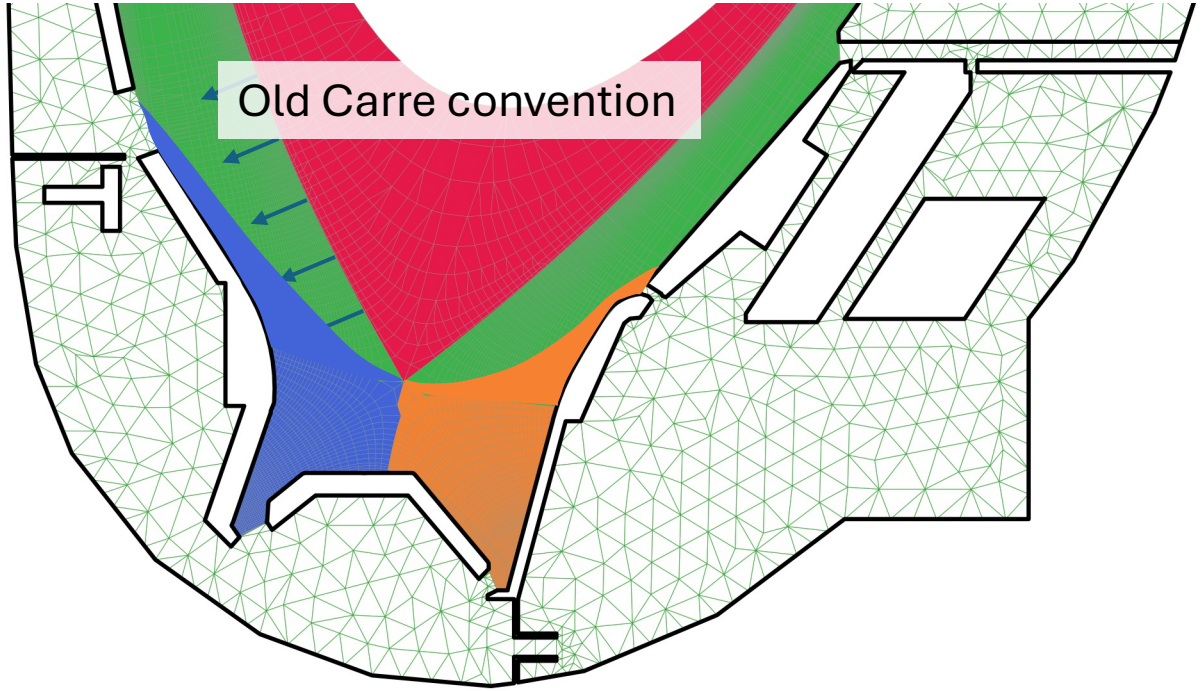


Identify ”rays” moving radially outwards from X-point until a wall is found

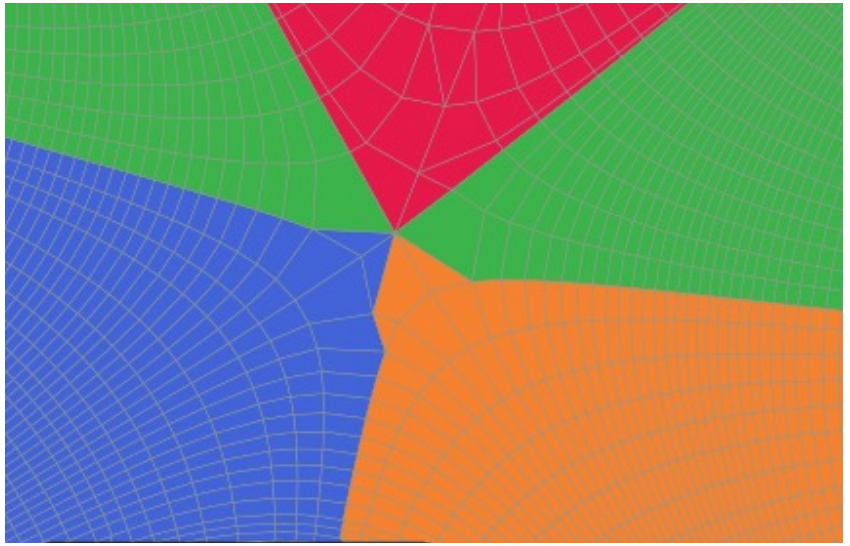


Definition of ”divertor volume” physically relevant, even more than with Carre!
Remaining ”main SOL” region is really a region which is radially exposed to the confined plasma, e.g. where anomalous transport coefficients should be applied

Boundary between main SOL and divertor SOL and between inner/outer PFR branches



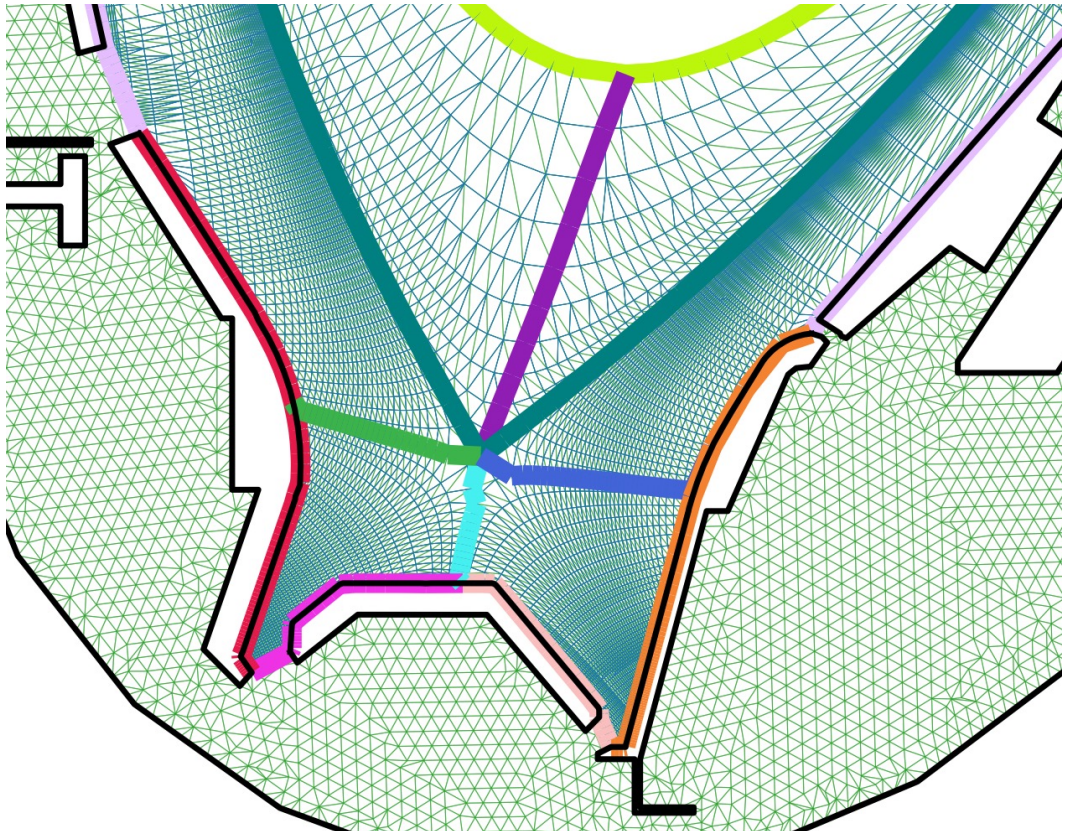
Identify ”rays” moving radially outwards from X-point until a wall is found



Definition of ”divertor volume” physically relevant, even more than with Carre!
Remaining ”main SOL” region is really a region which is radially exposed to the confined plasma, e.g. where anomalous transport coefficients should be applied

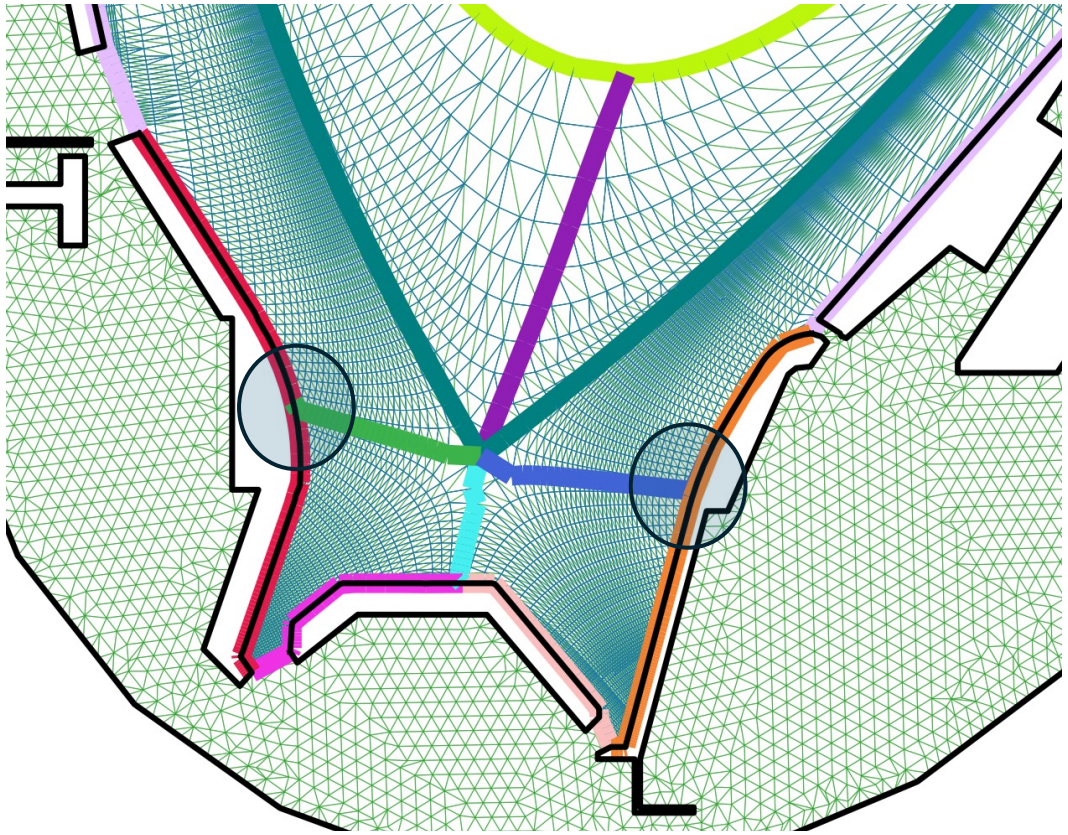
Several face regions may be missing, depending on the case

Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



Several face regions may be missing, depending on the case

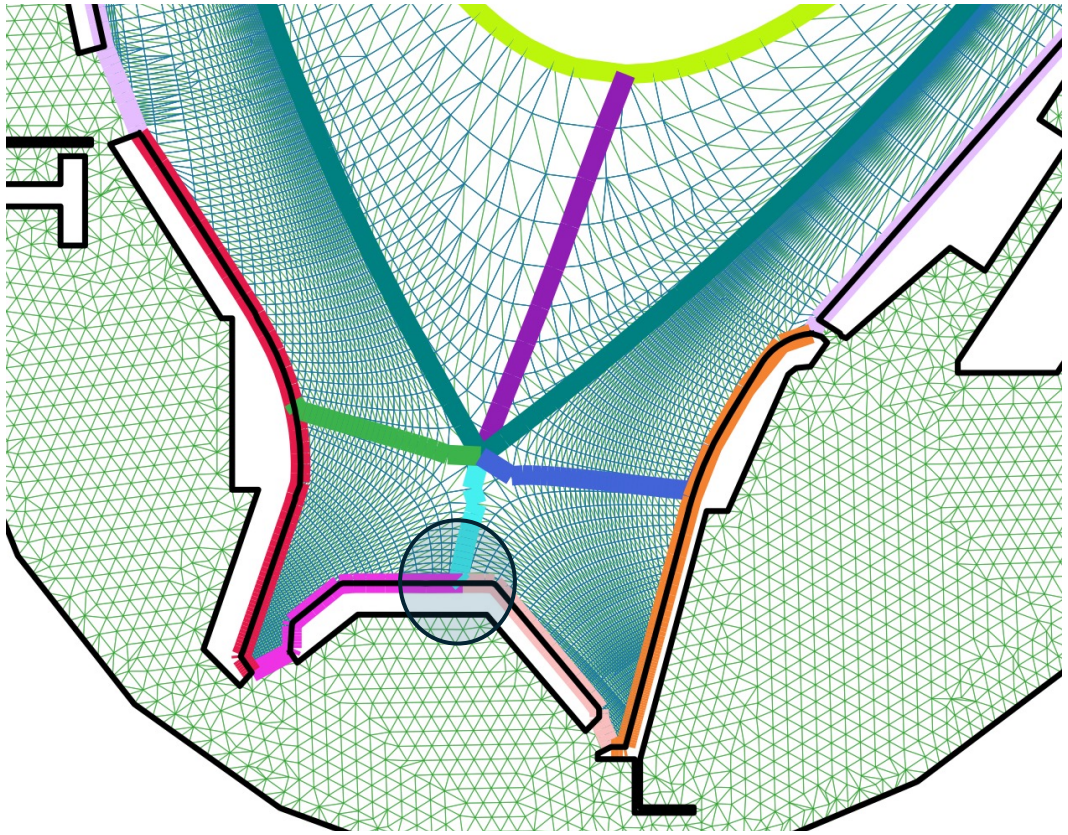
Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



- Segments defining inner/outer divertor entrance hit both the relative targets (defined as user input)
  → Divertor-to-wall face regions both empty

Several face regions may be missing, depending on the case

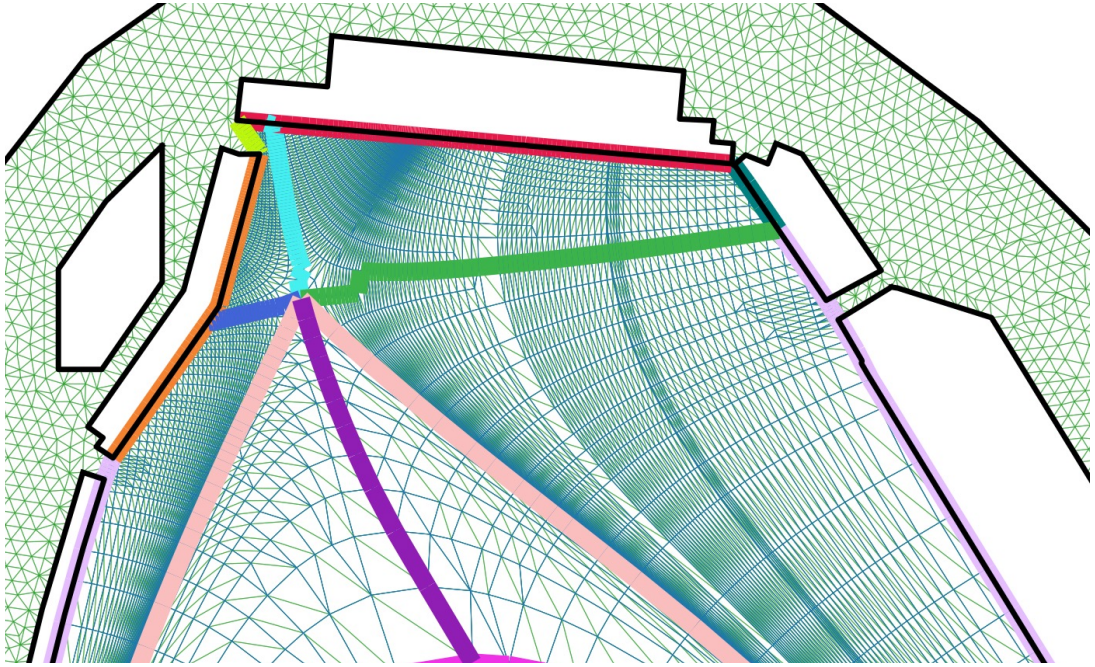
Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



- Segments defining inner/outer divertor entrance hit both the relative targets (defined as user input) → Divertor-to-wall face regions both empty
- There is a wall element between inner and outer target, and the segment defining the connection between the PFR sides intercepts it at half → PFR-to-wall face regions both full

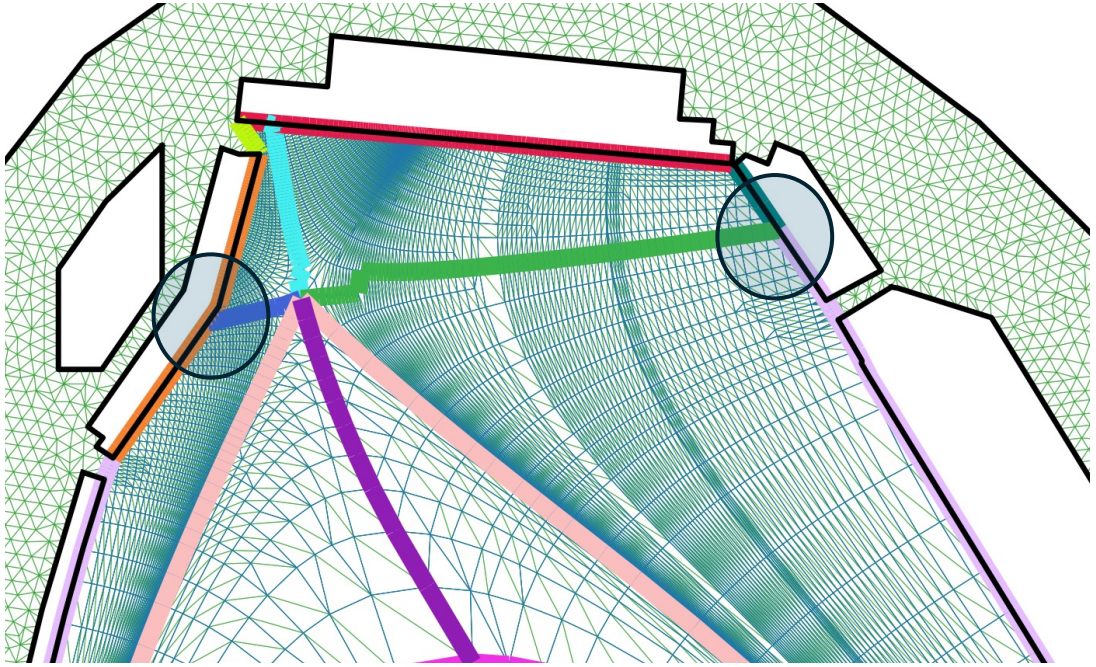
Several face regions may be missing, depending on the case

Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



Several face regions may be missing, depending on the case

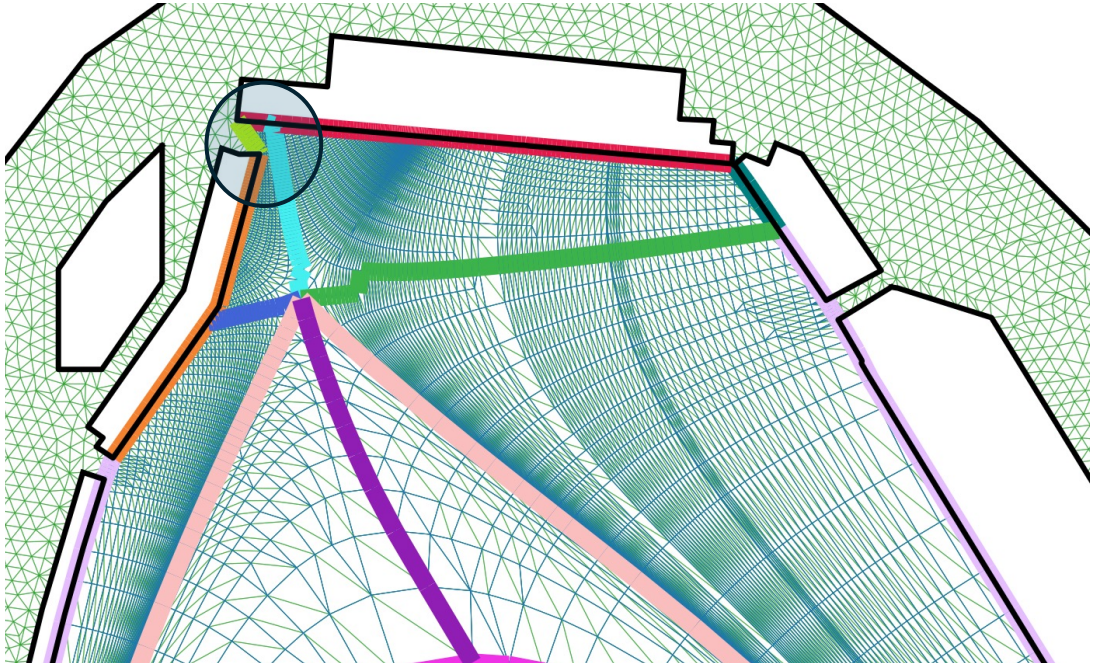
Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



- Segments defining inner divertor entrance hits inner target, but the one defining outer divertor entrance hits the wall above the start of the user-defined target
  → Outer divertor-to-wall face region full,
  inner divertor-to-wall face region empty

Several face regions may be missing, depending on the case

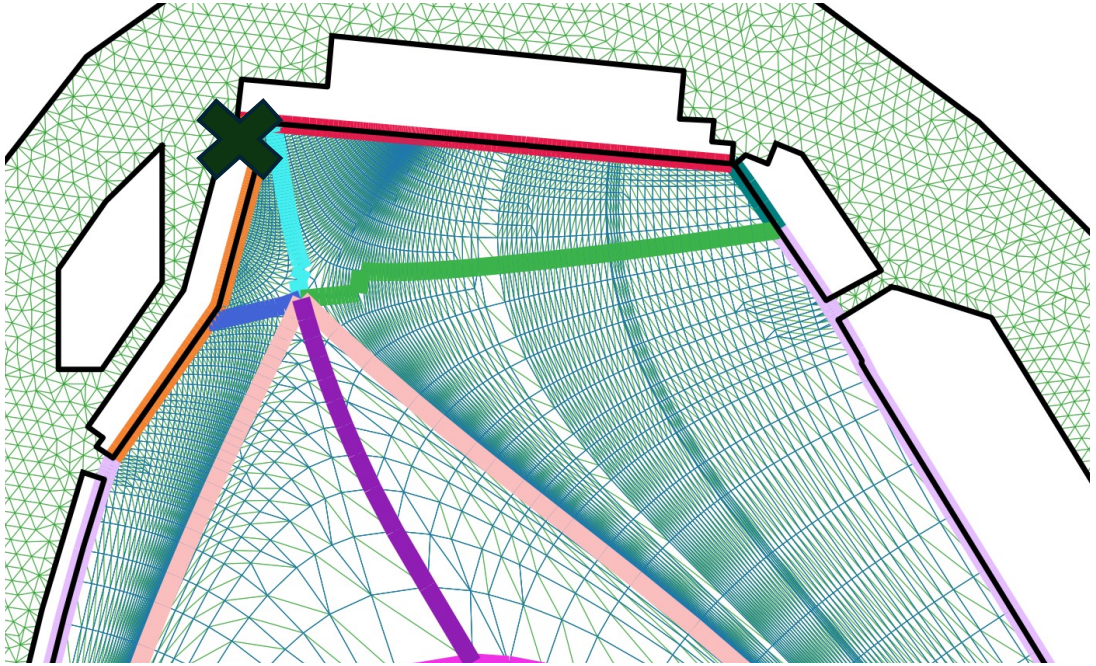
Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



- Segments defining inner divertor entrance hits inner target, but the one defining outer divertor entrance hits the wall above the start of the user-defined target
  → Outer divertor-to-wall face region full,
  inner divertor-to-wall face region empty
- There is a wall element between inner and outer target, but the segment defining the connection between the PFR sides intercepts the outer target
  → Inner PFR-to-wall face region full,
  outer PFR-to-wall face region empty

Several face regions may be missing, depending on the case

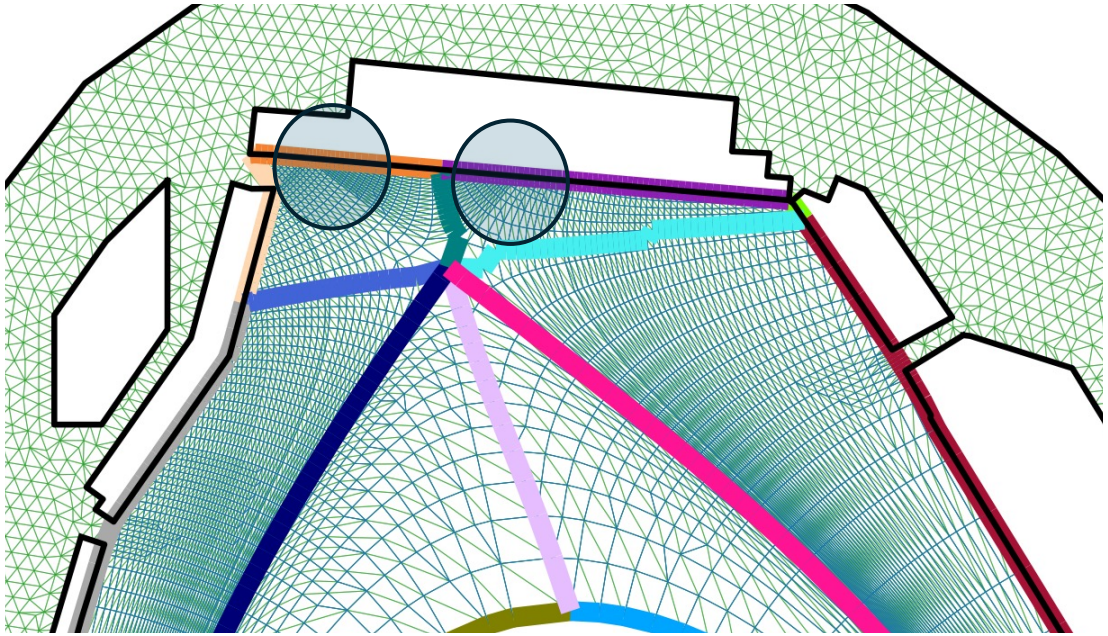
Divertor-to-main-wall boundary and PFR-to-wall boundary might be full or empty, depending on the particular case



- Segments defining inner divertor entrance hits inner target, but the one defining outer divertor entrance hits the wall above the start of the user-defined target
  → Outer divertor-to-wall face region full,
  inner divertor-to-wall face region empty
- There is a wall element between inner and outer target, but the segment defining the connection between the PFR sides intercepts the outer target
  → Inner PFR-to-wall face region full,
  outer PFR-to-wall face region empty
- If the two user-defined surfaces defining the inner and outer target were adjacent, then both PFR-to-wall face regions would be necessarily empty

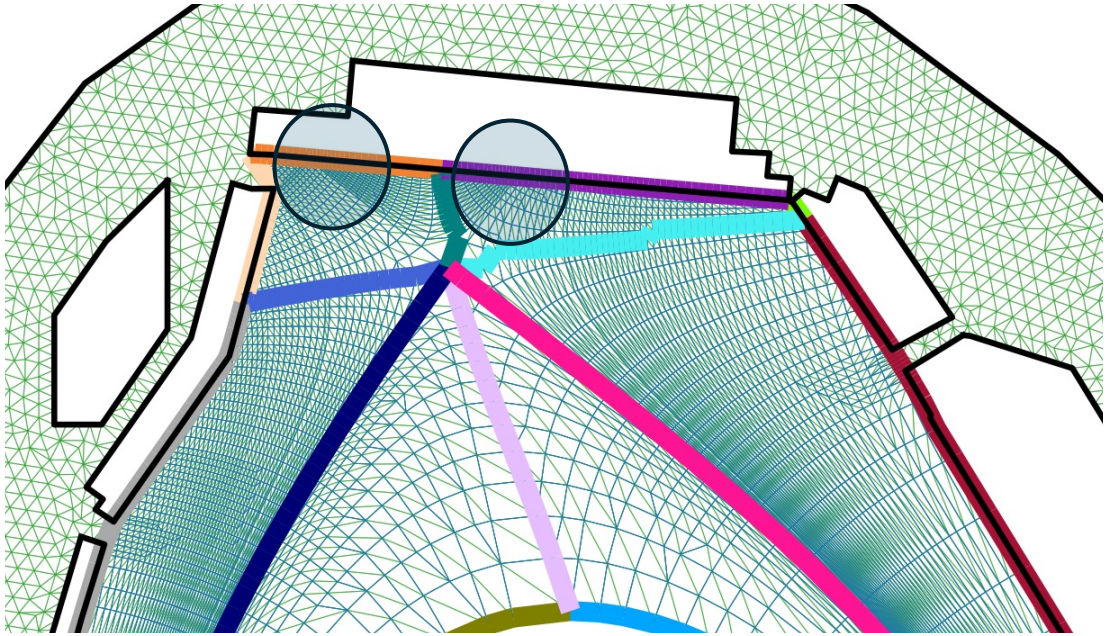
Several face regions may be missing, depending on the case

Divertor targets may also degenerate into one single input segment



Several face regions may be missing, depending on the case

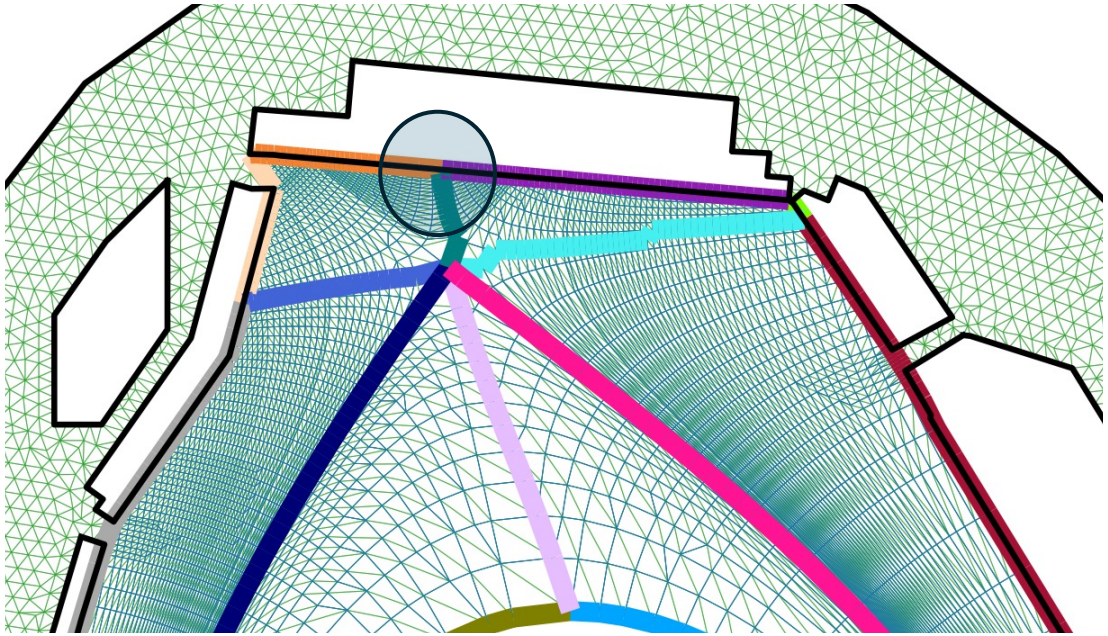
Divertor targets may also degenerate into one single input segment



- If both strikepoints lie on the same target segment, then one must impose same segment(s) number(s) for both inner and outer target as Goat input switches

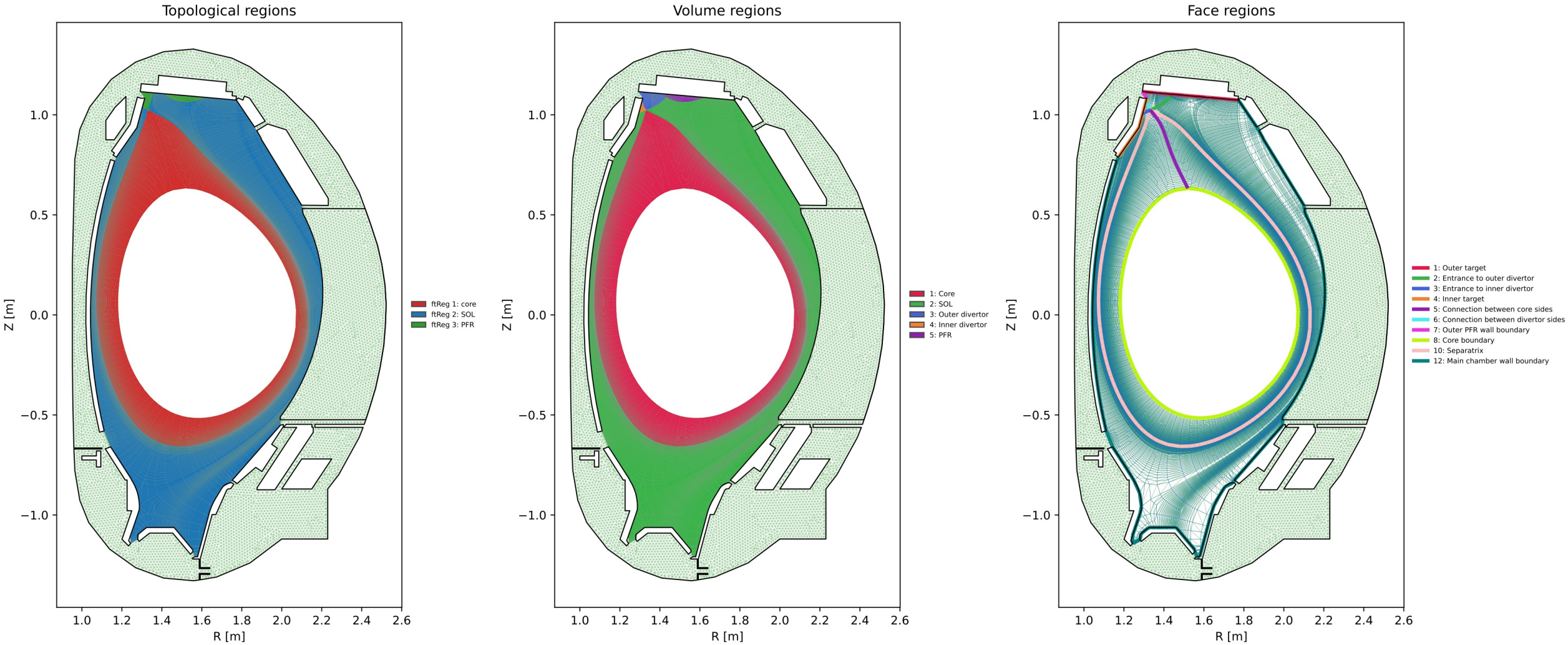
Several face regions may be missing, depending on the case

Divertor targets may also degenerate into one single input segment



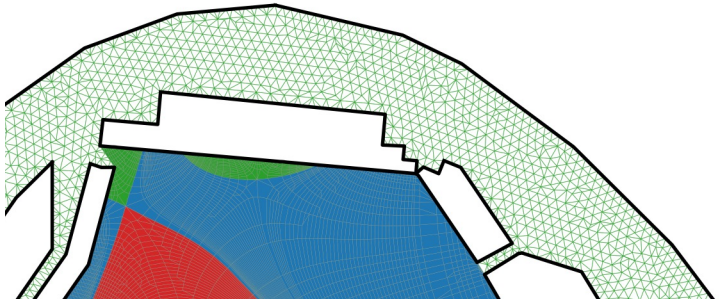
- If both strikepoints lie on the same target segment, then one must impose same segment(s) number(s) for both inner and outer target as Goat input switches
- Result: inner and outer target as face regions found by splitting the same original (single) target structure in two by the border which defined the connection between the two PFR sides (radially emanating from X-point)
- Consequently: both face regions defining PFR-to-wall boundaries are necessarily empty in this case

Result (USN → LFS SF-)

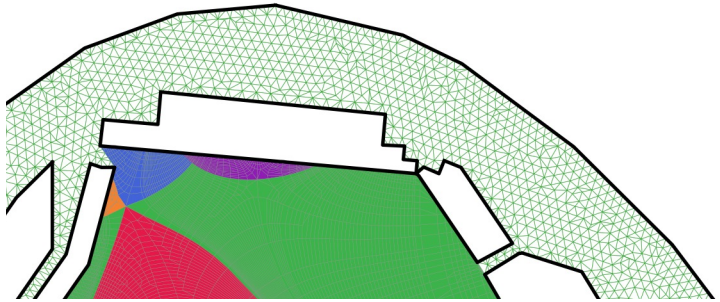


Result (USN → LFS SF-)

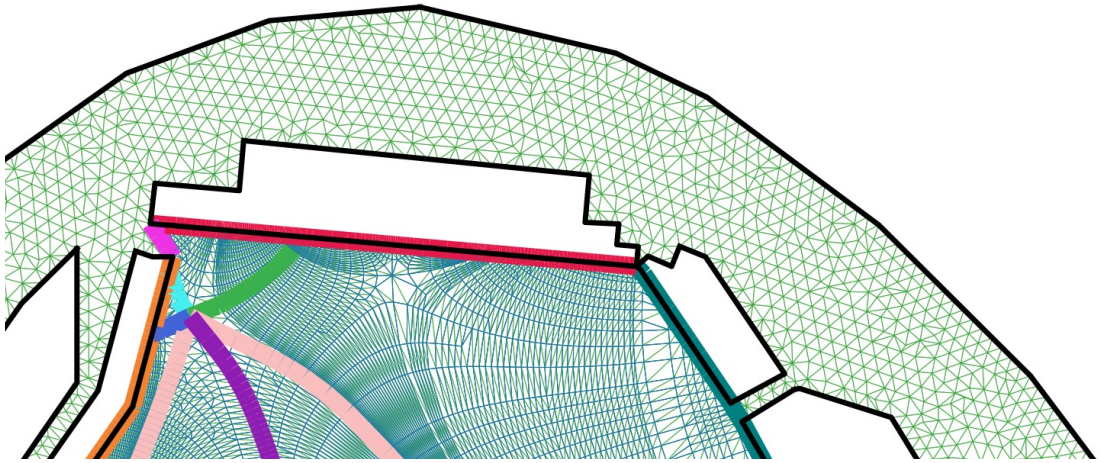
Topological regions



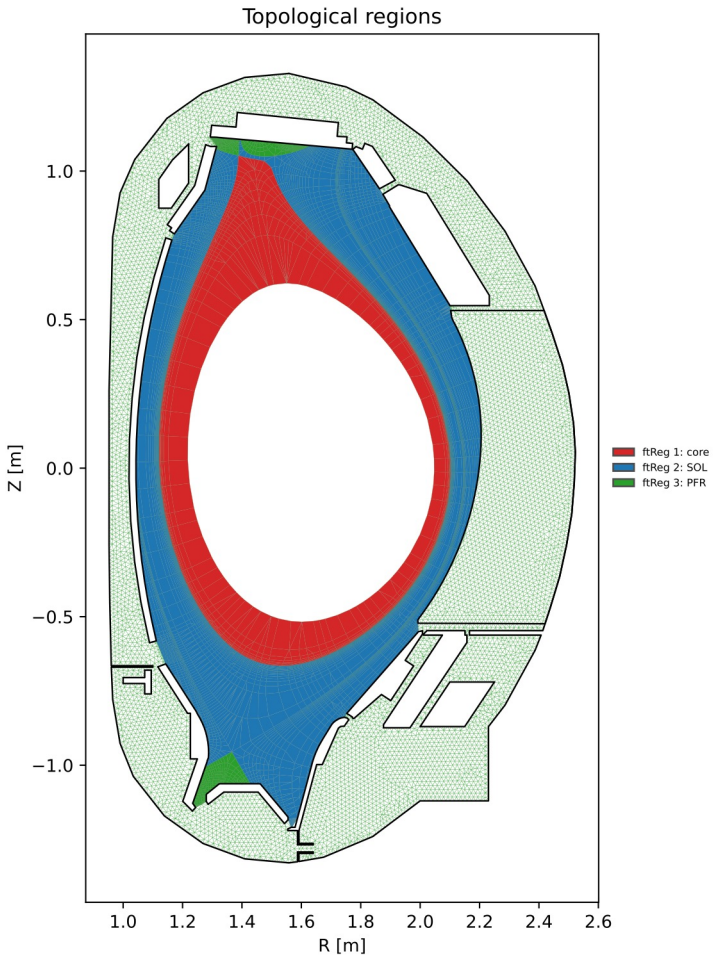
Volume regions

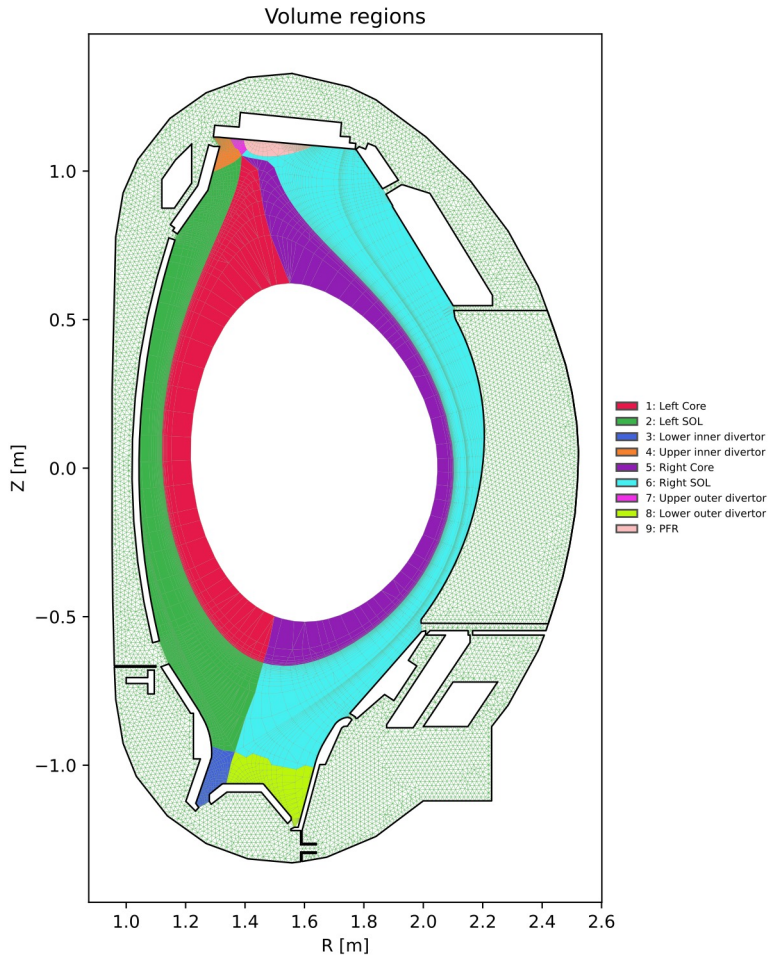


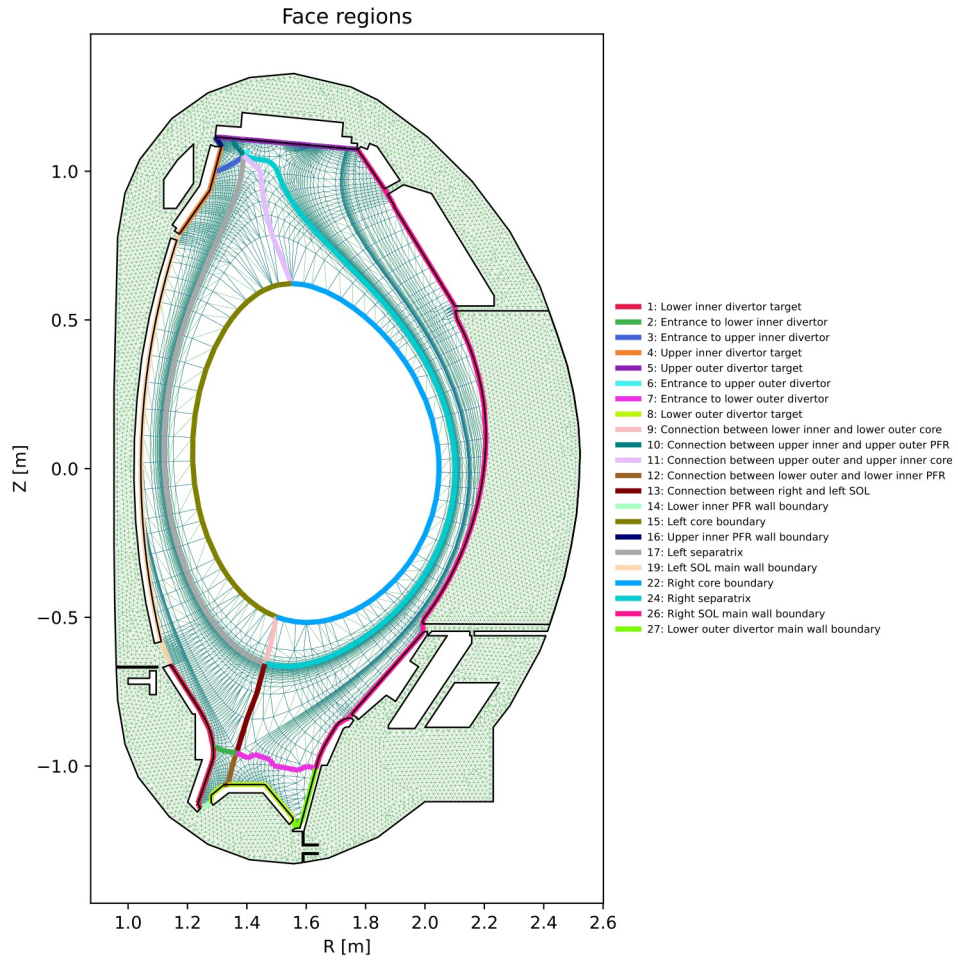
Face regions



Result (CDN → LFS SF-)

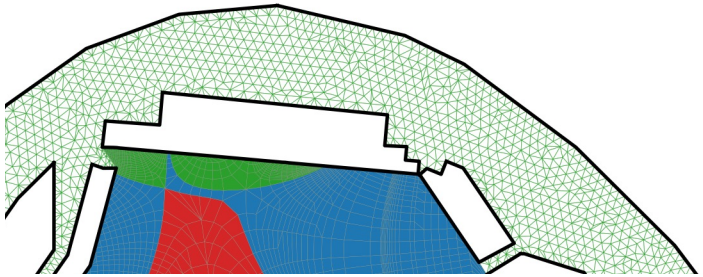




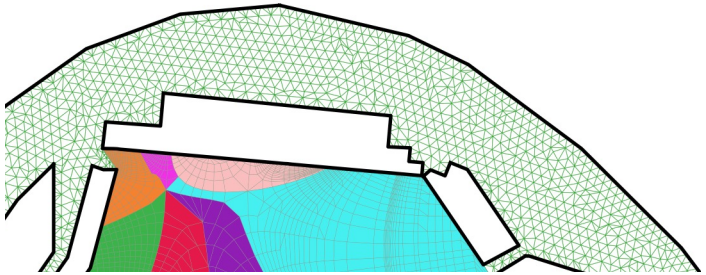


Result (CDN → LFS SF-)

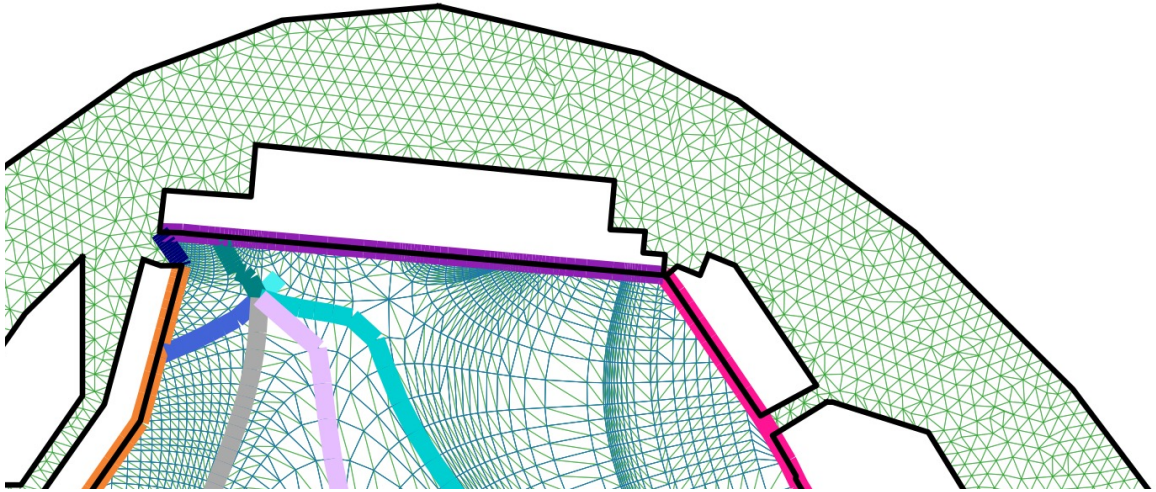
Topological regions



Volume regions



Face regions



My proposal, elaborated:

- Goat classifies internally a basic topology, if this falls into a historically SOLPS-relevant category (one O-point → limiter, SN, CDN, DDN → classification based ONLY on number and behavior of primary X-point per lower/upper divertor):
  - Two lower targets if there is at least 1 X-point below the O-point
  - Two upper targets if there is at least 1 X-point above the O-point

My proposal, elaborated:

- Goat classifies internally a basic topology, if this falls into a historically SOLPS-relevant category (one O-point → limiter, SN, CDN, DDN → classification based ONLY on number and behavior of primary X-point per lower/upper divertor):
  - Two lower targets if there is at least 1 X-point below the O-point
  - Two upper targets if there is at least 1 X-point above the O-point
- Configurations still with one O-point but with more than 1 X-point per divertor, are STILL classified as such, since only what the first X-point does matter for this classification (e.g. snowflake with a primary X-point and a secondary X-point, but both in the same divertor, still classified as SN).

My proposal, elaborated:

- Goat classifies internally a basic topology, if this falls into a historically SOLPS-relevant category (one O-point → limiter, SN, CDN, DDN → classification based ONLY on number and behavior of primary X-point per lower/upper divertor):
  - Two lower targets if there is at least 1 X-point below the O-point
  - Two upper targets if there is at least 1 X-point above the O-point
- Configurations still with one O-point but with more than 1 X-point per divertor, are STILL classified as such, since only what the first X-point does matter for this classification (e.g. snowflake with a primary X-point and a secondary X-point, but both in the same divertor, still classified as SN).
- Only for configurations with more than 1 O-point, the classification is “GENERAL”.

My proposal, elaborated:

- Goat classifies internally a basic topology, if this falls into a historically SOLPS-relevant category (one O-point → limiter, SN, CDN, DDN → classification based ONLY on number and behavior of primary X-point per lower/upper divertor):
  - Two lower targets if there is at least 1 X-point below the O-point
  - Two upper targets if there is at least 1 X-point above the O-point
- Configurations still with one O-point but with more than 1 X-point per divertor, are STILL classified as such, since only what the first X-point does matter for this classification (e.g. snowflake with a primary X-point and a secondary X-point, but both in the same divertor, still classified as SN).
- Only for configurations with more than 1 O-point, the classification is “GENERAL”.
- This classification (number and behavior of primary X-point(s) + number and behavior-of additional X-points) is directly passed to B2.5: Goat → traduit.out.b2us → b2fgmtry
  Therefore no need of doing any X-point classification inside of B2.5

My proposal, elaborated:

- Current goat_integration branch provides divertor-related fields (strDiv, divFcP, ...) to B2.5 as tied to number of X-points/strikepoints.
  My proposal: keep this definition in Goat (stays useful for really general topologies), but rework it in B2.5:

My proposal, elaborated:

- Current goat_integration branch provides divertor-related fields (strDiv, divFcP, ...) to B2.5 as tied to number of X-points/strikepoints.
  My proposal: keep this definition in Goat (stays useful for really general topologies), but rework it in B2.5:
  - If the topology is classified as one of the basic ones (limiter, SN, CDN, DDN, also including any additional X-points), these fields are reworked, in the init_mapping subroutine, to being tied to only the behavior of the primary X-points

My proposal, elaborated:

- Current goat_integration branch provides divertor-related fields (strDiv, divFcP, ...) to B2.5 as tied to number of X-points/strikepoints.
  My proposal: keep this definition in Goat (stays useful for really general topologies), but rework it in B2.5:
  - If the topology is classified as one of the basic ones (limiter, SN, CDN, DDN, also including any additional X-points), these fields are reworked, in the init_mapping subroutine, to being tied to only the behavior of the primary X-points
    → All downstream consumers of this variables (fields written in IMAS IDs, NetCDF outputs, tracing files...) automatically conserve the historical classification of targets, where max. 2 X-points, one per side (and max. 4 targets) were considered. With the additional advantage that now this includes also the cases with any additional X-points
    (e.g.: if we have a LFS SF- with 2 X-points, we still consider the LFS target as a single entity, even if it hosts multiple strikepoints now)

Consequences of this proposal:

- The geometry IDs already existing for LFS SF (GEOMETRY_LFS_SNOWFLAKE_MINUS, GEOMETRY_LFS_SNOWFLAKE_PLUS) become completely obsolete.

Consequences of this proposal:

- The geometry IDs already existing for LFS SF (GEOMETRY_LFS_SNOWFLAKE_MINUS, GEOMETRY_LFS_SNOWFLAKE_PLUS) become completely obsolete.
- For target-related output files (b2time.nc, tracing files) a SF does not split anymore a target into 3 separate targets, one for each strikepoint, but keeps only one (more natural IMHO).

Consequences of this proposal:

- The geometry IDs already existing for LFS SF (GEOMETRY_LFS_SNOWFLAKE_MINUS, GEOMETRY_LFS_SNOWFLAKE_PLUS) become completely obsolete.
- For target-related output files (b2time.nc, tracing files) a SF does not split anymore a target into 3 separate targets, one for each strikepoint, but keeps only one (more natural IMHO).
- B2.5 should stop using max(NREG) and max(FREG) since this value will be not relevant anymore (in case there are X-points additional to the primary one(s))

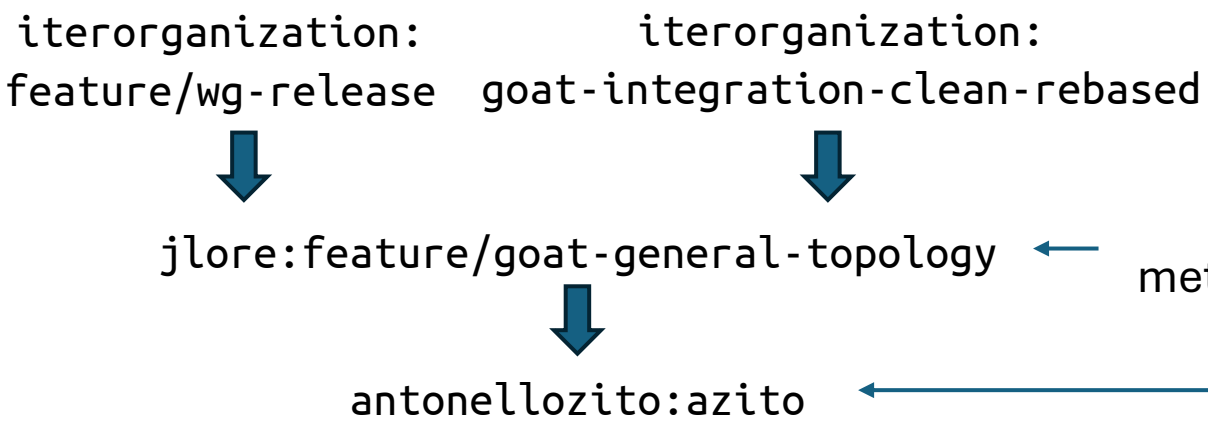
Consequences of this proposal:

- The geometry IDs already existing for LFS SF (GEOMETRY_LFS_SNOWFLAKE_MINUS, GEOMETRY_LFS_SNOWFLAKE_PLUS) become completely obsolete.
- For target-related output files (b2time.nc, tracing files) a SF does not split anymore a target into 3 separate targets, one for each strikepoint, but keeps only one (more natural IMHO).
- B2.5 should stop using max(NREG) and max(FREG) since this value will be not relevant anymore (in case there are X-points additional to the primary one(s))
- For REALLY general topologies (geometryID from Goat = GENERAL):
  - writing of b2tallies.nc stay the same (purely based on a variable number of volume and face regions)
  - b2time.nc should write for now only midplane data – divertor data are strictly tied to 2-or-4 targets classification, not adapt for general topologies
  - also ANK tracing files should not be written (same issue with b2time.nc)

All this is already implemented and tested

(based on the branches already worked on by Jeremy):

B2.5:



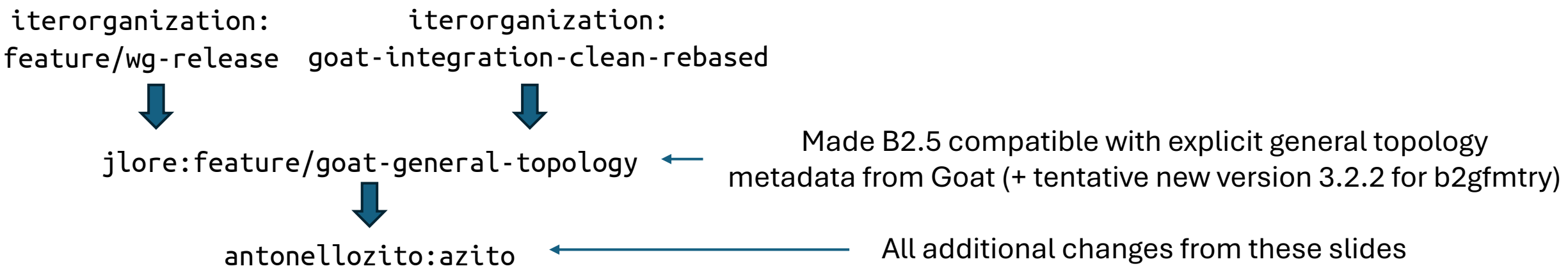
Made B2.5 compatible with explicit general topology metadata from Goat (+ tentative new version 3.2.2 for b2gfmtry)

All additional changes from these slides

All this is already implemented and tested

(based on the branches already worked on by Jeremy):

B2.5:



Goat:

