LLNL-TR-838677

LAWRENCE LIVERMORE NATIONAL LABORATORY

INGRID: An interactive grid generator for 2D edge plasma modeling

B. M. Garcia, M. V. Umansky, J. Watkins, J. Guterl, O. Izacard

August 10, 2022

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

INGRID: an interactive grid generator for 2D edge plasma modeling

B. M. Garcia,[1, a] M. V. Umansky,[1, b] J. Watkins,[2, c] J. Guterl,[3, d] and O. Izacard[1, e]

[1]*Lawrence Livermore National Laboratory, Livermore, California, USA*

[2]*Brigham Young University, Idaho, USA*

[3]*General Atomics, San Diego, California, USA*

(Dated: February 6, 2022)

A fusion boundary-plasma domain is defined by axisymmetric magnetic surfaces where the geometry is often complicated by the presence of one or more X-points; and modeling boundary plasmas usually relies on computational grids that account for the magnetic field geometry. The new grid generator INGRID (Interactive Grid Generator) presented here is a Python-based code for calculating grids for fusion boundary plasma modeling, for a variety of configurations with one or two X-points in the domain. INGRID first performs partitioning over the domain consisting of a small number of patches conforming to the magnetic field and wall geometry; then it generates a subgrid on each of the patches and joins them into a global grid. This domain partitioning strategy makes possible a uniform treatment of various configurations with one or two X-points in the domain. This includes single-null, double-null, and other configurations with two X-points in the domain. The INGRID design allows generating grids either interactively, via a parameter-file driven GUI, or using a non-interactive script-controlled workflow. Results of testing demonstrate that INGRID is a flexible, robust, and user-friendly grid-generation tool for fusion boundary-plasma modeling.

[a]Electronic mail: garcia299@llnl.gov

[b]Electronic mail: umansky1@llnl.gov

[c]Electronic mail: joeymwatkins@gmail.com

[d]Electronic mail: guterlj@fusion.gat.com

[e]Electronic mail: izacard1@llnl.gov

Program Summary

Program Title: INGRID: Interactive Grid Generator for Tokamak Boundary Region

CPC Library link to program files: (to be added by Technical Editor)

Developer's repository link: https://github.com/LLNL/INGRID

Code Ocean capsule: (to be added by Technical Editor)

Licensing provisions: MIT

Programming language: Python

Supplementary material:

Nature of problem: Modeling of tokamak boundary edge plasma physics is essential for the design and optimization of future fusion reactors. A crucial component of certain edge transport modeling codes is the computational grid representing the magnetic topology and plasma facing components within a tokamak. Generation of grids can be a tedious process that requires frequent intervention from the user of the grid generator. Despite their importance, robust simple-to-use grid-generation tools are nonexistent. The creation of a grid generator capable of both modeling and automatically identifying advanced divertor configurations is of great importance.

Solution method: INGRID utilizes user-provided MHD equilibrium data and the target plates and limiter geometry to automatically identify magnetic topology, and generate the appropriate computational grid in block structured manner. First, the magnetic topology identification algorithm decides how the computational domain is partitioned. Within each partition, a local subgrid is generated which is used to construct the global grid ready for export.

I. INTRODUCTION

A. Tokamak boundary and magnetic divertors

Research in tokamak edge plasma physics is critical for realizing practical fusion energy and designing future fusion reactors. One of the greatest challenges that tokamak edge plasma researchers face today is determining effective methods for controlling particle and heat fluxes on tokamak plasma-facing components (PFC) while maintaining good core-plasma performance.

Early on in the tokamak research, controlling plasma interaction with the material wall was carried out with a toroidal or poloidal limiter, i.e., a material surface projected a short distance from the wall into the boundary plasma and thus limiting its extent. However, since the 1980s it was recognized that using the so-called magnetic divertor configuration is superior for controlling interaction of plasma with material wall, and since then a magnetic divertor has become an essential feature of virtually all high-performance tokamaks. The magnetic divertor is a device that creates a certain configuration of the magnetic field that allows redirecting plasma fluxes to a remote location in the tokamak vacuum chamber.

Due to severity of plasma-material interactions expected in next-generation tokamaks, in the last 10-20 years the tokamak boundary plasma community developed strong interest in advanced divertor configurations[1]. The traditional divertor configuration uses a first-order null point (X-point) for the poloidal magnetic field. This null point is usually placed at either the bottom or the top of the core plasma. The traditional double-null configuration uses two first-order X-points with one at the bottom and one at the top of the core plasma. In contrast, several advanced divertor configurations have been proposed where a secondary X-point is included in the divertor region. These include snowflake-like configurations and the X-point Target configuration. Snowflake-like configurations approximate a configuration with an exact second-order null of the poloidal field dubbed "snowflake"[2]. In practice, instead of an exact second-order null, a configuration is used where two regular X-points are brought close together, which leads to snowflake-plus and snowflake-minus configurations[3]. On the other hand, for an X-Point Target configuration[4], a secondary X-point is introduced in the divertor far away from the primary X-point in the divertor leg near the target plate. Each of these divertor configurations is characterized by the locations of the primary and secondary X-points in the domain. The primary X-point is the most significant as it separates the plasma into the hot core region and colder scrape-off layer (SOL) region. However, a secondary X-point helps redirect and distribute the flux of plasma particles and energy over multiple locations (i.e. strike points) on the material surface. Furthermore, a secondary X-point may also help increase plasma radiation in the divertor, potentially causing other interesting and important effects in divertor plasma.

B. Tokamak edge plasma modeling

Due to complexity of physics and geometry, modeling of tokamak edge plasmas relies heavily on numerical simulations. An earlier approach to tokamak boundary modeling, pioneered in[5] is based on a 2D axisymmetric collisional fluid plasma model with ad-hoc perpendicular transport coefficients. A more recent approach started, among others, by[6–8] attempts to reproduce edge plasma turbulent transport by direct simulation, solving 3D time-evolution equations for collisional fluid plasma. These days modeling of tokamak boundary plasma turbulence is getting increasingly sophisticated and mature, with several major boundary plasma codes making great strides in fluid turbulence modeling, e.g.,[9–12]; and in kinetic turbulence modeling, e.g.,[13–15].

Still, the traditional axisymmetric 2D fluid modeling remains an important area of research, with several major codes in use, e.g.,[16–18]. Clearly it has its limitations: using ad-hoc anomalous transport coefficients instead of calculating the anomalous transport from a first principles based theory or simulation, using toroidal symmetry assumption while some non-axisymmetric perturbations are certainly present in the plasma, and using collisional fluid closure while the collisional mean-free-path is not necessarily much smaller than the system spatial scales. On the other hand, a comprehensive first principles based calculation of the anomalous transport is not yet available for tokamak boundary plasma, and the toroidal symmetry and high collisionality assumptions are not unreasonable for the edge region in a typical tokamak. So, with all its limitations, the traditional 2D modeling approach has its place, and it produces many useful insights that help solidify interpretation of tokamak boundary plasma experimental data. In particular, it remains the primary modeling approach used in the area of tokamak divertor modeling. The interest to advanced divertors raises new demands to computational tools used for axisymmetric tokamak edge plasma transport modeling. In particular, snowflake-like configurations are producing new data on plasma transport and radiation, which needs modeling for interpreting experimental data. To help modeling advanced divertors, tokamak edge plasma transport codes are extended to include more than a single X-point in the divertor volume[19,20]. Computational grid generation for such configurations has its subtleties, and in this manuscript we are presenting a new grid generation tool INGRID for dealing with configurations with either one or two X-points in the domain.

C. Grid generation for tokamak axisymmetric edge plasma simulations

Tokamak boundary and divertor plasma modeling relies heavily on edge transport modeling codes such as UEDGE[16], SOLPS[17], EDGE2D[18], just to mention a few major ones. These codes are similar in many ways; they all solve the time-evolution fluid equations for toroidally-symmetric, collisional plasma based on the Braginskii equations, using ad-hoc radial transport coefficients. The simulations are typically carried out in the actual geometry of a modeled tokamak in order to account for details pertaining to magnetic field geometry and plasma-facing components.

Computational grids for tokamak boundary plasma transport modeling are usually chosen to follow the magnetic flux surfaces; a choice made to avoid numerical pollution caused by the extreme anisotropy of plasma transport along and across the magnetic field[21]. As a result of this, the grid topology can become highly nontrivial with one or several X-points present in the simulation domain.

There are several grid generators for tokamak edge plasma region currently in use. Among those, the UEDGE code uses a grid generator that is a part of the UEDGE package, SOLPS normally uses grid generator CARRE[22], and EDGE2D usually relies on grid generator GRID2D[23]. These are sufficient in most cases for modeling single-null and double-null configurations. However, modeling of advanced divertors may require incorporating secondary X-points in the divertor region that yield nontrivial snowflake-like grid topologies. Additionally, grid generators currently in use for major edge transport codes are not inherently designed to produce computational grids for general configurations containing more than one X-point at *arbitrary* locations in the domain.

To increment grid generation capabilities for tokamak boundary plasma transport modeling, in particular for advanced divertors, a new grid generator INGRID has been developed, as described in the present report. INGRID (Interactive Grid Generator) is a Python based, interactive, grid generator for edge plasma transport modeling that is capable of modeling configurations with up to two X-points anywhere in the computational domain. The INGRID package provides a robust set of tools and an easy to use GUI. Internally, INGRID automatically identifies the appropriate divertor configuration to model, and thus removes challenges that typically arise when generating grids. The result is tool that can be easily incorporated and benefit a user's workflow for edge plasma modeling. The INGRID algorithm's inspiration was drawn from the older IDL-based project Gingred[24] where a "divide and conquer", domain partitioning approach for grid generation was first attempted. Although Gingred provided grid generation capabilities, an important motivation for implementing INGRID was to leverage an open-source, modern language such as Python.

D. Paper outline

The aim of this section is to provide a short introduction to tokamak edge plasma modeling, the state of grid generation for tokamak edge plasma simulations, and motivate the development of INGRID. Section II discusses the classification of equilibria found within a plasma device and defines the magnetic topologies that INGRID has been designed to model. Following our discussion on classification, section III discusses the computational methodology INGRID adopts. In particular, the interpolation scheme, topology analysis algorithm, Patch Map construction algorithm, and grid construction methodology is discussed. In section IV, results of INGRID testing are presented. From there, section V discusses our software design choices made for the INGRID package. Finally, section VI summarize our results and point the reader to where INGRID can be obtained.

II. CLASSIFICATION OF EQUILIBRIA

The geometry of a tokamak (or a similar toroidally axisymmetric plasma device, such as a spheromak or a reversed-field pinch) is defined by the flux function $\Psi(R,Z)$, where R,Z are the cylindrical coordinates, such that $\nabla\Psi\cdot\vec{B}=0$[25]. The poloidal magnetic field components B_R, B_Z can be expressed as

$$B_R = -\frac{1}{R}\frac{\partial\Psi}{\partial Z} \tag{1}$$

$$B_Z = \frac{1}{R}\frac{\partial\Psi}{\partial R} \tag{2}$$

Surfaces $\Psi(R,Z)$=const form a set of nested magnetic flux surfaces confining the plasma, where the magnetic field is tangential to the flux surfaces. The nulls of the poloidal magnetic field, $B_R = B_Z = 0$ correspond to extrema or saddle points of the flux function $\Psi(R,Z)$. The "magnetic axis" corresponds to an extremum of Ψ at the innermost flux surface. Furthermore, we consider first-order saddle points of Ψ where the first derivatives vanish; these points are "X-points".

To understand the range of geometric possibilities with the presence of one and two X-points in the domain, consider the diagrams in Fig.(1). If there is a single X-point in the region it defines a separatrix which is a flux surface containing the X-point (Fig. 1a). The X-point is a self-intersection point of the separatrix that divides the plane into three topologically distinct regions: the "core", "private-flux", and "scrape-off-layer". First, the core is the region containing the magnetic axis. Next, the region lying opposite to the core plasma region across the X-point is the private-flux (PF) region. Finally, the remaining region is the "common flux region", or "scrape-off-layer" (SOL) region.

Further we will be using normalized flux and "compass" notation. The normalized poloidal flux is defined in the standard way,

$$\hat{\Psi} = \frac{\Psi - \Psi_{magx}}{\Psi_{sepx} - \Psi_{magx}}, \tag{3}$$

where Ψ_{magx} and Ψ_{sepx} are the values of the poloidal flux at the magnetic axis and at the primary separatrix.

Now we define the compass directions North-South-East-West that are associated with an X-point, as illustrated in Fig. 1a. The North direction (N) is defined to point into the core plasma along the $\nabla\hat{\Psi}$ direction, whereas the South direction (S) is defined to point into the private-flux along the $\nabla\hat{\Psi}$ direction. East (E) and West (W) are then defined orthogonal to the N and S directions. Then, as shown in Fig. the midplane points are labeled M_E and M_W according to the compass direction at the primary X-point. The advantage of the compass notation is that it is independent of the location of the primary X-point; even if it is near the inner or outer midplane.

As shown in Fig. 1b and 1c, a second X-point can be added outside of the core plasma region. Ignoring the degenerate case when the secondary X-point is on the primary-separatrix, there are two topologically distinct possibilities with respect to the primary separatrix: either the second X-point can be in the common flux region, (Fig. 1b), or the second X-point can be in the private flux region, (Fig. 1c). The former case is called "snowflake-minus" and the latter case is called "snowflake-plus"[2].

From the grid generation perspective, there are further variations of "snowflake-like" configurations. In original analysis of near-snowflake configurations by Ryutov et al.[26], based on local expansion of the flux function, the angle θ pointing from the primary X-point to the secondary one was used to distinguish topologically different configurations[26]. Further, the notation based on the θ angle was adopted to describe the mapping of orthogonal flux coordinates to the grid index space, for different snowflake-like configurations labeled by θ=15°, 45°, 75°, 105°, 135°, and 165° in[27],

In INGRID, for a given $\Psi(R,Z)$, the first step of the calculation is producing a Patch-map by partitioning the domain by Ψ=const lines and orthogonal to them Θ=const lines. That is similar to orthogonal grids considered in[27], and constructing those Patch-maps we recover the six topologically distinct cases considered in[27]. Therefore, here we also adopt the θ notation for snowflake-like configurations, e.g., using SF15 for the case labeled θ=15° in[27].

With the compass notation used here, the six cases introduced in[27] are defined by the locations of the secondary X-point X_2 and its orthogonal projection on the primary separatrix X_2',

- UDN: $SF-$, $X_2' \in [M_E, M_W]$
- SF15: $SF-$, $X_2' \in [M_E, X_1]$
- SF165: $SF-$, $X_2' \in [M_W, X_1]$
- SF45: $SF-$, $X_2' \in [S_E, X_1]$
- SF135: $SF-$, $X_2' \in [S_W, X_1]$
- SF75: $SF+$, $X_2' \in [S_E, X_1]$
- SF105: $SF+$, $X_2' \in [S_W, X_1]$

The orthogonal projection of the secondary point on the primary separatrix is illustrated in Fig. (1), and the range of magnetic configurations is illustrated in Figs. (3-10).

All in all, with the single-null (SNL) configuration included, for either one and two X-points in the domain, there are eight possible configurations. We do not consider here degenerate cases where the secondary X-point is exactly on the primary separatrix, the projected point X_2' is exactly on the primary X-point X_1, or when X_2' is exactly on a midplane point M_E or M_W; the assumption is that a practical, experiment-relevant configuration would always have some finite degree of asymmetry to fall into one of the eight considered categories.

III. COMPUTATIONAL ALGORITHM METHODOLOGY

A. Domain partitioning strategy

The INGRID workflow consists of two main steps: (i) partitioning the domain into a collection of zones/patches that we call a "Patch Map", and (ii) generating a subgrid for each of the patches in the Patch Map. This idea illustrated in Fig. (2) where a Patch Map is shown with one of the patches covered with a subgrid.

An INGRID Patch Map is a template of the grid to be generated since the geometry of magnetic flux surfaces, in particular the X-points and the magnetic axis, and the geometry of plasma facing material surfaces all together are used to define the Patch Map. For calculating a subgrid, the code divides each four-sided Patch object into a number of radial and poloidal zones according to user specification. Finally, a post-processing step is performed in which all subgrids are joined together to finalize the global grid.

B. Magnetic field interpolation

Input data for grid generation are expected in the form of the poloidal magnetic flux Ψ sampled on a rectilinear grid in R, Z, from an MHD reconstruction code such as EFIT[28]. INGRID utilizes a bicubic interpolation implementation[29] for obtaining the values of the poloidal magnetic flux Ψ and its derivatives between data points of the provided magnetic equilibrium. This functionality is provided by class `RectBivariateSpline` belonging to the `scipy.interpolate`[30,31] package.

Poloidal flux surfaces are calculated in INGRID by integrating the ODEs,

$$\dot{R} = -\frac{1}{R}\frac{\partial \Psi}{\partial Z} \tag{4}$$

$$\dot{Z} = \frac{1}{R}\frac{\partial \Psi}{\partial R} \tag{5}$$

whereas surfaces orthogonal to poloidal flux surfaces are calculated by integrating the ODEs,

$$\dot{R} = \frac{1}{R}\frac{\partial \Psi}{\partial R} \tag{6}$$

$$\dot{Z} = \frac{1}{R}\frac{\partial \Psi}{\partial Z} \tag{7}$$

The bicubic interpolation guarantees continuity of first derivatives of Ψ at the edges of cells of the original R,Z grid, so the resulting flux surfaces are smooth.

C. Calculation of reference points

The topology of magnetic flux surfaces in tokamak edge plasmas is defined by relative positions of X-points and the magnetic axis. These key reference points are calculated in INGRID by finding nulls of the poloidal field, solving the equation

$$\left(\frac{\partial\Psi}{\partial R}\right)^2 + \left(\frac{\partial\Psi}{\partial Z}\right)^2 = 0 \tag{8}$$

INGRID utilizes method `scipy.optimize.root`[30,32] for this purpose. For finding the correct root of this equation the solver needs an accurate initial guess; the initial guesses are provided by the user upon inspection of the MHD data. These guesses are the approximate coordinates of the magnetic axis and for up to two X-points that are expected to be included in the computational domain.

D. Topology analysis

INGRID conducts an analysis and classification of the magnetic topology based on the user specification of one or two X-points in the domain. We first consider the case of a single X-point. The tokamak edge plasma community often classifies single X-point configurations as “upper single null” or “lower single null." However, for INGRID there is no distinction. If one X-point is specified, then INGRID considers the aforementioned configurations as a general single-null (SNL) configuration. Instead of using “lower” and “upper” for the divertor, and “inner” and “outer” for target plates, INGRID defines compass directions North-South-East-West associated with the primary X-point. The North direction is defined to point into the core plasma along the $\nabla\Psi$ direction, whereas the South direction is defined to point into the private-flux along the $\nabla\Psi$ direction. East and West are then defined orthogonal to the North and South directions. For example, the inner target plate of an LSN configuration is in the south-west direction of an SNL configuration, as illustrated in Fig. (3). Similarly, the inner target plate of a USN configuration is in the south-east direction. Determining the North-South-East-West directions associated with the primary X-point is the extent of the analysis performed in the SNL case.

If there are two X-points in the domain, INGRID first executes the compass analysis from the SNL case on the primary X-point. Next, INGRID determines whether the secondary X-point is in the private-flux (PF) region or in the common-flux region (SOL) with respect to the primary separatrix. If the secondary X-point resides in the private-flux, the configuration is of type SF+. If not, the configuration is of type SF-. This determination is conducted by representing the PF as a generalized polygon and determining whether a 2D point representing the X-point is contained within the polygon or not. The vertices of the polygon correspond to the points obtained from the portion of the limiter contained between points S_W and S_E (denoted as $[S_W, S_E]$) and line tracing (detailed in section E) the two arcs $[X_1, S_W]$, $[X_1, S_E]$. INGRID implements the point-in-polygon test with the help of class `matplotlib.path`. Representing the polygon boundary as a `matplotlib.path` object, the method `matplotlib.path.contains_point` can then determine whether the area enclosed by the path contains the given point or not. With the point-in-polygon test completed, an orthogonal projection is constructed from the secondary X-point to the primary separatrix. This is done by line tracing equations (3) and (4) where the termination criteria is intersection with the primary separatrix. Finally, the specific magnetic topology is determined by the criteria outlined in Section II for classification of equilibria.

E. Patch Map construction

After the magnetic geometry analysis for a given magnetic field is completed, INGRID creates a Patch Map corresponding to the identified topology. A Patch Map is a collection of Patch objects that are defined by four vertices and generally non-straight edges. All together, the Patch Map provides a template of the finalized global grid. Internal to the code, the Patch Map is represented as a 2D array of Patch objects with the dimensions of the array corresponding to the radial and poloidal directions respectively. The 2D layout of a Patch Map establishes the underlying block structure of the mesh, and allows for the identification of Patch neighbors in index space.

A Patch Map is constructed via numerical integration along $\Psi =$const and $\nabla\Psi$ directions through the X-points and other reference points in order to represent the user specified radial/poloidal surfaces used to define the domain boundaries. INGRID utilizes the `solve_ivp` method from the `scipy.integrate`[30] package for all numerical integration purposes. In particular, LSODA[33] was selected as the `solve_ivp` method of integration. Equations 4 provide INGRID the capability to trace out poloidal flux surfaces, whereas equations 6 provide INGRID the capability to trace out surfaces orthogonal to poloidal flux surfaces.

In a Patch Map, radial domain boundaries are flux surfaces corresponding to maximum/minimum values of the poloidal flux function Ψ that are specified by the user in the parameter file. The poloidal domain boundaries are defined by user with their provided geometry files (e.g target plates). For each of the divertor configurations that INGRID can use - which includes a single-null, unbalanced double-null, and six snowflake-like configurations - there is a specific type of Patch Map that defines the topology of this configuration. For example, for the SNL configuration shown in Fig. (3), its Patch Map includes two radial zones, two poloidal zones defining divertor legs, and four poloidal zones defining the edge plasma domain around the last closed flux surface. Such a Patch Map that contains these twelve Patches is sufficient to represent a general single-null geometry, albeit in a basic and crude way. Note that one could use only ten Patches to represent the single-null topology, but using twelve Patches allows matching the general shape of a single-null configuration. more accurately. Furthermore, a finer grid representing a single-null geometry can be always represented as this Patch Map with local refinement applied to one or several of these twelve patches. For the more complicated unbalanced double null (UDN) geometry shown in Fig. (4), the Patch Map must include three radial zones because there are two separatrices. Here there are also four poloidal zones to represent four divertor legs. With four poloidal zones covering the core domain, there is a total of eight poloidal zones. All in all, there are twenty four patches in the UDN case. For each of the snowflake-like configurations there are twenty seven patches, as illustrated in Figs. (5-10). INGRID enforces the consistency of radial boundaries between adjacent Patches by defining the interface between Patch boundaries in terms of each other. For Patches adjacent to target plates (no neighboring Patch object), INGRID will copy and paste a segment of the target plate in order to create a radial Patch boundary that conforms exactly to the provided geometry. Poloidal consistency of Patch boundaries is attained by simply ensuring line tracing continues from the end point of the adjacent Patch. It should be noted that the continuation process cannot guarantee the consistency of Patch boundaries along the upper core. The size of this boundary mismatch is proportional to the line tracing tolerance, and can be controlled by the `tol` attribute found within the integrator settings of the parameter file.

F. Grid construction

For the magnetic configuration in question, a Patch Map is created in order to serve as the template of the final grid. Consider an arbitrary Patch within the Patch Map. The radial sides of this Patch are defined by two flux surfaces, $\Psi(R,Z) = \Psi_1$ and $\Psi(R,Z) = \Psi_2$. The poloidal sides of a Patch are usually constructed to be aligned with $\nabla\Psi$, thus making the Patch Map locally orthogonal. The poloidal sides of a Patch, however, can deviate from the $\nabla\Psi$ direction. For example, for those Patches that contain the poloidal boundaries of the domain, given by the target plates, one of the sides is defined by the target plate shape. The curve describing the target plate can be arbitrary, as long as it does not form "shadow regions", i.e., Ψ is a monotonic function of the length along the plate.

A Patch can be divided into a number of radial and poloidal zones, thus forming a subgrid local to this Patch. The radial zones are constructed to be aligned with flux surfaces, so the global grid remains aligned with the poloidal magnetic field. For the poloidal zones, the main algorithm is based on dividing the Patch poloidally into uniform length line segments. That said, there are user options in the code for controlling the radial and poloidal distribution of the subgrid within a Patch.

The radial/poloidal dimensions and distributions of a subgrid can be dependent on subgrids of other Patches in the Patch Map. This results from the global grid being Cartesian in the index space. Thus, the poloidal dimension of a grid has to be consistent for Patches that are stacked on top of each other radially, and the radial dimension of a grid has to be consistent for those patches stacked on top of each other poloidally.

G. Grid customization

INGRID provides users a number of tools which can be controlled via the parameter file for refinement of the soon to be generated grid. These controls allow users to modify the default behavior that takes place in two stages of the grid generation process: the Patch Map creation stage, and the subgrid generation stage.

First we describe some controls that take effect during the Patch Map generation stage. Internal to the Patch Map generation scripts, the default algorithm for constructing a Patch Map relies on the horizontal plane through the magnetic axis (commonly referred to as the midplane) in order to define the radial boundaries for certain Patches. To modify the resulting Patch Map, INGRID allows shifting the reference location representing the “effective magnetic axis” for the grid generation purposes vertically and horizontally. This shifting can be used to compress Patch objects in order to increase the subgrid density without increasing the number of grid cells within a Patch subgrid. In addition to simple translations of the effective magnetic axis, the user can set two angles defining a “generalized midplane” rather than using the default horizontal directions for the midplane. Since the midplane can be thought of as two rays emanating from the magnetic axis with angles 0 and π radians, a “generalized midplane" is defined similarly but with user-defined “effective magnetic axis” location, and with user-specified angles for both rays. Also, for those Patches that include an X-point as one of their vertices, the default radial boundaries use the East and West directions from the X-point along the $\nabla\Psi$ direction. However, the user has the capability to replace the curves with straight lines that are oriented in a desired direction.

Next, the subgrid generation within a Patch can be adjusted as well. By default, during Patch refinement, grid seed-points are distributed uniformly along the length of radial boundaries and uniformly along the length of poloidal boundaries in locally-normalized Ψ. This default behavior can be changed so that grid seed-point placement obeys a user specified distribution function. Another powerful customization feature that can be enabled during the subgrid generation stage is the “`skewness_correction`" tool for mitigating grid shearing. This tool allows the user to bound the angles found within the interior of a grid cell in order to generate nearly orthogonal subgrids in patches. The implementation is as follows. Let a quadrilateral cell of a subgrid be defined by four nodes A, B, C, and D where AB is the top-face, BC is the right-face, CD is the bottom-face, and DA is the left-face of the cell. The angle $\measuredangle DAB := \alpha$ is measured. If α is not within a user-defined range $[\alpha_{\min}, \alpha_{\max}]$, then the node D is translated along the poloidal flux surface until α is within the range. The direction of translation is determined by whether $\alpha \geq \alpha_{\max}$ or $\alpha \leq \alpha_{\min}$. During the translation of D, we must also avoid collision with a neighboring node on the poloidal flux surface. To ensure collision does not occur, the translation stops when the moving node D comes within ε of a neighboring node. Upon termination, the current value of α is used to define the transformed cell. An example of the `skewness_correction` feature applied to a grid can be seen in Fig.(11). The cells of the local mesh in a Patch are modified one at a time in row-major order when `skewness_correction` is applied. This sequential nature of application means that after the cells of row i have been modified, the nodes of the adjacent cells in row $i+1$ must be updated. This updating operation is applied to neighboring Patches when row $i+1$ is outside the local mesh within a Patch. Thus, the order in which `skewness_correction` is applied to cells matters and is taken into account by the algorithm itself.

IV. PERFORMANCE

A. Scaling of calculation time

The results of INGRID timing tests are shown in Table (I) and in Fig. (12). These tests were run on an Apple Mac Mini (2020 model) housing an Apple M1 chip (3.2 GHz processor). The scaling appears sublinear for small grids; for larger grids it asymptotes to linear. Note that the cost of grid generation is not significant in a typical edge plasma modeling workflow; running the simulation takes orders of magnitude more computing time. This scaling makes INGRID practical for constructing large grids.

SNL			SF75		
Cells Per Patch	Total Cells	Time (s)	Cells Per Patch	Total Cells	Time (s)
9	108	8.02	9	243	21.32
16	192	11.44	16	432	27.69
25	300	14.79	25	675	34.45
36	432	18.29	36	972	41.10
49	588	21.58	49	1323	48.31
64	768	25.02	64	1728	54.82
81	972	28.49	81	2187	62.24
100	1200	32.27	100	2700	68.69
121	1452	35.32	121	3267	75.44
144	1728	38.61	144	3888	82.63
169	2028	41.98	169	4563	89.97
196	2352	45.46	196	5292	96.21
225	2700	49.08	225	6075	103.71

Table I: Results of a performance test for grid generation for both an SNL and SF75 configuration. Grids were generated with $n \times n$ many cells per Patch with $n = \{3, 4, 5, \dots, 15\}$. With $n \times n$ subgrid dimensions, SNL configurations contain $12n^2$ many cells, whereas SF75 configurations contain $27n^2$ many cells.

B. Grid testing

To assess the performance of grids calculated by INGRID, several tests have been performed with the UEDGE code.

In these tests, UEDGE solutions were compared, using grids from INGRID and grids produced with the UEDGE internal grid generator; for the same physics problem statement, the same boundary conditions, using the same (or very close) domain geometry.

In one of these test problems, a snowflake-like SF75 configuration was used, based on magnetic reconstruction data from the TCV tokamak. The UEDGE grid generator cannot deal with the SF75 configuration directly; but it can be set up to treat each X-point as a part of a separate SN configuration, and then joining two such SN grids together one can produce a grid for the full SF75 domain.

The UEDGE code was set up to solve to the steady state the time-evolution equations for plasma density, plasma parallel momentum, ion thermal energy, and electron thermal energy.

$$\frac{\partial}{\partial t} n_i + \nabla \cdot \left[n_i \vec{V}_i \right] = S_i \tag{9}$$

$$\frac{\partial}{\partial t} \left[M n_i V_{i,||} \right] + \nabla \cdot \left[M n_i \vec{V}_i V_{i,||} - \hat{\eta}_i \nabla V_{i,||} \right] = S_{m,||} \tag{10}$$

$$\frac{\partial}{\partial t} \left[\frac{3}{2} n T_i \right] + \nabla \cdot \left[\frac{5}{2} n_i T_i \vec{V}_i + \vec{q}_i \right] = S_{E,i} \tag{11}$$

$$\frac{\partial}{\partial t} \left[\frac{3}{2} n T_e \right] + \nabla \cdot \left[\frac{5}{2} n_e T_e \vec{V}_e + \vec{q}_e \right] = S_{E,e} \tag{12}$$

Here we use the standard notation: n_i is the plasma density, $\vec{V}_i$ is the plasma fluid velocity, $T_{e,i}$ is the electron and ion temperature, $\vec{q}_{e,i}$ is the electron and ion heat flux, S_i, $S_{m,||}$, $S_{E,e,i}$ are sources of plasma density, parallel momentum, and electron and ion thermal energy; full details of the model are given in[16].

The grids used for the calculation are shown in Fig. (13). The grid generated with the UEDGE grid generator is constructed to be strictly locally orthogonal; the grid from INGRID is not orthogonal. Also, there is slight difference in the domain shape; the grid from INGRID uses flat target plates while the grid from UEDGE uses curved target plates orthogonal to Ψ. Still, the steady state solutions exhibit essentially the same distributions of plasma density, temperature, and parallel flow velocity, as can be seen in Fig. (14). A quantitative comparison of radial plasma profiles at the outer midplane was carried out, indicating agreement within 2%; that level of agreement was considered convincing enough to indicate absence of any major issue in INGRID grids.

Note that grid convergence of a numerical solution may be complicated by the presence of X-points in the domain. Previously, convergence tests using grids similar to those discussed in this manuscript were conducted with the linear stability code 2DX, in the context of linear analysis of ideal-ballooning and resisitive-ballooning instabilities in X-point divertor geometry[34], and in a circular geometry[35]. It is important to note that in those tests the convergence rate in the X-point geometry was found to be close to the 1st order, while in the circular geometry the convergence rate was close to the 2nd order. This is consistent with the analysis in[36], and also with[37,38] pointing out that the X-point metric singularity for a flux-aligned coordinate system degrades grid convergence for a numerical solution. That general result should apply to INGRID generated grids that contain X-points, but the details will be problem specific and will depend on the numerical methods used.

V. SOFTWARE DESIGN AND USER INTERACTION

A. INGRID package

INGRID has been exclusively developed in the Python programming language due to the increasing popularity in major tokamak plasma modeling projects such as OMFIT[39,40] and PyUEDGE[41], as well as to take advantage of the free, community supported graphical and numerical libraries. The `Ingrid` class contained within the `ingrid` module provides the primary API for users. This `Ingrid` class is used to activate INGRID's GUI mode and also contains high-level methods for importing data, visualizing data, analyzing data, grid-generation, and exporting of data; all of which can be utilized noninteractively in Python scripts. Class `IngridUtils` is contained within the `utils` module and serves as the base class for `Ingrid`. `IngridUtils` class methods encapsulate much of the lower-level software details used to implement the methods in the `Ingrid` class. Because of this, `IngridUtils` is encouraged for use by advanced users and developers of INGRID. In addition to `IngridUtils`, class `TopologyUtils` can be found within the `utils` module. In a manner similar to `IngridUtils`, the `TopologyUtils` class serves as a base class for each magnetic-topology class within the `topologies` subpackage. `TopologyUtils` contains key methods for generating Patch Maps, visualizing data, generating grids, and exporting grids. Eight magnetic-topology classes are contained within their own modules within the `topologies` subpackage: `SNL`, `UDN`, `SF15`, `SF45`, `SF75`, `SF105`, `SF135`, and `SF165`. Each magnetic-topology class contains configuration specific line-tracing instructions for construction of Patch Maps, Patch Map layout information, and exported grid file formatting information. `Ingrid` and `IngridUtils` conduct analysis of MHD equilibrium data in order to decide which magnetic-topology class to instantiate from the `topologies` subpackage. The `IngridUtils` class always maintains a reference to the instantiated object in order to effectively manage grid-generation.

All GUI operation is managed by class `ingrid_gui` within the `guis` subpackage. INGRID's GUI front-end was developed with the Tkinter package; a Python interface to the Tk GUI toolkit that is available within the Python Standard Library. Class `ingrid_gui` is simply responsible for managing event handling, and managing an `Ingrid` object that is used to drive the GUI with direct calls to the available high-level methods.

Beyond modules `ingrid` and `utils`, modules `geometry`, `interpol`, and `line_tracing` form the computation and modeling foundation of INGRID. Class `EfitData` can be found within the `interpol` module. Class `EfitData` is used to provide an interpolated representation of provided MHD equilibrium data. `EfitData` computes partial derivative information of MHD equilibrium data, provides interpolated Ψ function values by interfacing with class `scipy.interpolate.RectBivariateSpline`, and contains methods for visualization of interpolated MHD equilibrium data. Module `geometry` contains classes `Point`, `Line`, `Patch`, and `Cell`. These classes are the building-blocks for creation of Patch Maps and generation of grids.

B. INGRID geometry object hierarchy

We adopted an object-oriented approach to grid generation, which calls for the development of a set of tools that can be utilized throughout our project. Here we discuss how INGRID defines a collection of geometric classes in order to make the grid generating process as simple as possible for all magnetic topologies of interest. To do so, INGRID defines the following collection of geometric abstractions: the Point class, the Line class, the Cell class, and the Patch class. All together, these classes arm INGRID with the ability to represent any magnetic topology of interest. We provide a very brief description of the classes here. Figure 2 illustrates the geometry collection described below. A Point object simply represents an arbitrary (R,Z) spatial coordinate in the computational domain. Along with (R,Z), coordinates, Ψ values can be returned from the Point object. A Line object is defined by a list of two or more Point objects. This Line object definition allows for the representation of any arbitrary curve we may encounter (e.g. constant Ψ surface, target plate, limiter geometry). This can be done since the collection of Point objects correspond to segments that are the discretization of the curve of interest. A Patch object represents a closed region of the domain, and a block of the block-structured mesh INGRID computes. Patches are defined by four Line objects. The “North" Line (top), “East" Line (right), “South" Line (bottom), and “West" Line (left) define the Patch borders which form a closed, clockwise-oriented loop. Methods of the Patch class assist with visualization and computation. For example, Patch method `make_subgrid` directly handles grid generation for the local region of interest. A Cell object resides within a Patch and represents a quadrilateral grid cell. Cells are defined by five Point objects: four corners (NW, NE, SW, SE) and a center. These Cell objects provide the spatial and experimental data that are written to exported grid files.

From these definitions, we have the building blocks for modeling any of the magnetic topologies of interest we mentioned in the previous section. In particular, we aim to construct a collection of Patch objects representing the divertor configuration of interest. We call this collection of Patch objects a Patch Map. This Patch Map allows us to create a grid of Cell objects within each Patch, thus providing the final grid. Patch Map creation and grid generation is managed by a magnetic topology class of modeling interest. As of the current INGRID release, we have defined magnetic topology classes SNL, UDN, SF15, SF45, SF75, SF105, SF135, and SF165. These are contained within a dedicated topologies subpackage within the INGRID code.

C. INGRID parameter file

We have selected YAML[42] formatted files for our parameter file structure. A YAML file is similar to the familiar Fortran namelist files due to the key-value structure it employs, and is in an easy to read format that has extensive support within Python. Internal to each INGRID instance, a Python dictionary is used to store grid generation settings and data dependencies that are needed for all aspects of the algorithm. The YAML parameter file acts as an interface to this core dictionary since YAML entries are used to override the default INGRID settings. This YAML parameter file interface provides users the flexibility to model cases with the INGRID GUI and later reuse the same parameter file in scripts after loading the data as a dictionary. Some key controls within the parameter file include: EFIT file specification, specification of number of X-points, approximate coordinates of X-point(s) of interest, approximate magnetic-axis coordinates, Ψ level values, and target plate settings (files, transformations). Other controls in the parameter file include: path specification for data files, grid cell np/nr values, poloidal and radial grid transformation settings, limiter specific settings, saving/loading of Patch Maps, exported grid file settings, and debug settings. This is not an exhaustive list. Further details can be found in INGRID's Read The Docs online documentation.

D. INGRID target plate file

INGRID users must specify the geometry of the limiter (i.e., the entire first-wall contour surrounding the plasma) and/or target plates, to represent the shape of material walls in a modeled device. The limiter and target plates are represented in INGRID by a piecewise-linear model defined by a set of nodes; the (R,Z) coordinates of those nodes are expected to be provided in separate data files. There is one data file for the limiter and one for each target plate, either in the text format or as a NumPy binary. The names of those data files are set in the INGRID parameter file. In the case that the limiter and target plate data are provided in text format, the user must specify the (R,Z) coordinates for each point defining the surface sequentially on a separate line in the corresponding data file; and Python-formatted single- line comments can be included, as shown in the Appendix.

For use of NumPy binary files, users must also adhere to a particular internal file structure. Given two NumPy arrays of shape $(n,)$ that represent R and Z coordinate values respectively, one can define a NumPy array of shape $(2,n)$ representing the n-many points required to model the piecewise-linear model of interest. This NumPy array of shape $(2,n)$ can be saved into a NumPy binary file in order to be loaded into INGRID. In addition to the requirements above, INGRID asserts that strike-point geometry files used for Patch Maps are monotonic in Ψ along the length of the target plates (i.e. no shadow regions). The requirement of target plates to be monotonic in Ψ allows INGRID to parameterize the (R,Z) coordinates of the geometry with the Ψ values. With a unique mapping of Ψ to (R,Z) target plate coordinates, INGRID can generate Patch objects that conform exactly to the plate or limiter geometry. While operating INGRID in GUI mode, users will be warned if the loaded geometry file is not monotonic in Ψ along the target plate length.

E. INGRID workflow

INGRID can be operated via GUI or utilizing the INGRID library directly in Python scripts. The GUI workflow highlights the interactive nature of INGRID by allowing users to visually inspect MHD equilibrium data, configure geometry, and adjust parameter file values on the fly. Figure 15 shows the simple GUI with both MHD equilibrium data, and the corresponding grid plotted (text editor not pictured, see the Appendix for parameter file). For both GUI operation and scripting with INGRID, the high-level INGRID workflow is: (i) Parameter file visualization and editing, (ii) Analysis of MHD equilibrium data and creation of Patch Map, (iii) Patch Map refinement and grid export.

INGRID internally handles step (ii) and leaves the user to with steps (i) and (iii). These steps are where the user is able to customize the Patch Map and grid to meet their modeling needs.

Step (i) in the INGRID workflow allows users to visually inspect MHD equilibrium data, target-plates and limiter geometry, and Ψ-level contours that are specified within a loaded parameter file. Since creation of grids is tied directly to MHD equilibrium analysis and Patch Map creation, step (i) is crucial for successful grid generation. To simplify this step, the INGRID GUI provides an easy to use environment for preparation of a parameter file for the subsequent analysis of MHD equilibrium data and Patch Map creation. Examples of common operations at this step include modifications to strike-point geometry and Ψ-level boundaries for subsequent Patch Maps. Once a user is satisfied with parameter file settings, step (ii) can be immediately executed with no further user intervention. Should any errors in Patch Map creation occur (e.g. misplaced target-plates, Ψ-boundaries that do not conform to configuration specific requirements), INGRID will prompt the user and allow for appropriate edits to be made. Upon completion of step (ii), the created Patch Map will be provided to users as a new matplotlib figure. From here the user can decide to proceed with Patch Map refinement or start over at step (i) to make edits to the Patch Map.

In order to streamline grid generation and skip directly to step (iii), INGRID supports Patch Map reconstruction. This feature allows users to bypass line-tracing routines by reloading a saved Patch Map from a previous INGRID session. To do so, INGRID encodes essential geometry and topology analysis data in a specially formatted dictionary that is then saved as a NumPy binary file. Class `IngridUtils` handles the encoding and reconstruction of Patch Maps. These reconstruction features can be configured by the user within the INGRID parameter file.

After a Patch Map has been generated or reconstructed, users can configure grid generation specific settings that will be utilized during Patch Map refinement. Similar to Patch Map generation, once all local subgrids have been created within Patch objects, a new matplotlib figure is presented with the generated grid. From here, users can make grid generation setting edits in the INGRID parameter file, or proceed to exporting a grid file.

VI. SUMMARY

INGRID is a new grid generator for a tokamak boundary region, it is capable of producing grids for single-null (SNL), unbalanced double-null (UDN), and snowflake-like (SF) configurations. Currently, exported grids are in the format of the UEDGE code, as detailed in Ref. ([27]); future development may include addition of grid formats used by other codes if INGRID is adopted in the broader edge-plasma community, beyond UEDGE. INGRID can be utilized via the INGRID Python package, or through a parameter file driven GUI mode. Source code as well as documentation is publicly available on GitHub[43] and Read the Docs[44] respectively. The internal equilibrium topology analysis algorithm provides the ability to automatically identify the divertor configuration embedded within experimental data with minimal user interaction. The geometry class hierarchy approach to domain partitioning and Patch Map abstraction is an essential component of INGRID and results in a modular approach to grid generation. These localized grids are combined into a global grid that is then ready for export. Current computational scaling of grid generation algorithm follows a sublinear trend independent of the magnetic topology modeled. Comparison of INGRID against the internal grid generator in UEDGE is demonstrated for an SF75 snowflake-like configuration. These tests illustrate INGRID's ability to produce practical grids for tokamak edge modeling, for complex magnetic flux function with one or two X-points in the domain, and for nontrivial target plate geometry.

VII. ACKNOWLEDGMENTS

The authors would like to thank M.E.Rensink for his help with grid generation in UEDGE, and L.L.LoDestro for many critical comments on the manuscript. This work was performed for U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and General Atomics under Contract DE-FG02-95ER54309.

VIII. APPENDIX: PARAMETER FILE

An example INGRID parameter file is shown below. In the file, the user is supposed to provide settings relevant to grid and Patch generation: 1) For the magnetic field geometry, the name (with path) of data file, in the commonly used neqdsk format; 2) For radial boundaries, the values of the normalized poloidal flux Ψ for each of the radial boundaries; 3) For poloidal boundaries, the code has options to use one of these: a) limiter data embedded in the eqdsk file b) limiter data provided in a separate file (the file name and path must be included) c) target plates geometry in separate files, one per each plate (the file names and paths must be included); 4) How many X-points to include in the domain, 1 or 2; 5) Approximate R,Z coordinates for each of the included X-points and for the magnetic axis, to provide initial guess for the solver; 6) Dimensions of sub-grids; 7) Options related to grid customization; 8) Options related to integrator settings.

```
# ---------------------------------------------------------------
# User data directories
# ---------------------------------------------------------------
dir_settings:
  eqdsk: ../data/SNL/DIII-D/  # dir containing eqdsk
  limiter: .  # dir containing limiter
  patch_data: ../data/SNL/DIII-D/  # dir containing patch data
  target_plates: ../data/SNL/DIII-D/ # dir containing target plates
# ---------------------------------------------------------------
# eqdsk file name
# ---------------------------------------------------------------
eqdsk: neqdsk
# ---------------------------------------------------------------
# General grid settings
# ---------------------------------------------------------------
grid_settings:
  # -------------------------------------------------------------
  # Settings for grid generation
  # (num cells, transforms, skewness_correction)
  # -------------------------------------------------------------
  grid_generation:
    skewness_correction:
      all:
        active: True # true, 1 also valid.
        resolution: 1000
        theta_max: 120.0
        theta_min: 80.0
    np_default: 3
    nr_default: 3
    poloidal_f_default: x, x
    radial_f_default: x, x
  # ---------------------------------------------------------------
  # guard cell size
  # ---------------------------------------------------------------
  guard_cell_eps: 0.001
  # ---------------------------------------------------------------
  # num levels in efit plot
  # ---------------------------------------------------------------
  nlevs: 30
  # ---------------------------------------------------------------
  # num xpts
  # ---------------------------------------------------------------
  num_xpt: 1
  patch_generation:
    strike_pt_loc: target_plates # 'limiter' or 'target_plates'
    rmagx_shift: 0.0
    zmagx_shift: 0.0
  # ---------------------------------------------------------------
  # Psi levels
  # ---------------------------------------------------------------
  psi_1: 1.066
  psi_core: 0.95
  psi_pf_1: 0.975
  # ---------------------------------------------------------------
```

```
  # magx coordinates
  # ------------------------------------------------------------
  rmagx: 1.75785604
  zmagx: -0.0292478683
  # ------------------------------------------------------------
  # xpt coordinates
  # ------------------------------------------------------------
  rxpt: 1.300094032687
  zxpt: -1.133159375302
  # ------------------------------------------------------------
  # Filled contours vs contour lines
  # ------------------------------------------------------------
  view_mode: filled
# --------------------------------------------------------------
# Saved patch settings
# --------------------------------------------------------------
patch_data:
  file: LSN_patches_1597099640.npy
  preferences:
    new_file: true
    new_fname: LSN_patches_1597099640.npy
  use_file: false
# --------------------------------------------------------------
# Integrator
# --------------------------------------------------------------
integrator_settings:
  dt: 0.01
  eps: 5.0e-06
  first_step: 5.0e-05
  max_step: 0.064
  step_ratio: 0.02
  tol: 0.005
# ---------------------------------------------------------------
# Limiter settings
# ---------------------------------------------------------------
limiter:
  file: ''
  use_efit_bounds: false
# ---------------------------------------------------------------
# target plate settings
# ---------------------------------------------------------------
target_plates:
  plate_E1:
    file: d3d_otp.txt
    zshift: -1.6
  plate_W1:
    file: d3d_itp.txt
    zshift: -1.6
```

Listing 1: The YAML formatted configuration file. YAML files utilize Python formatted comments, keyword-value mappings, and nesting of structures via indentation.

REFERENCES

[1]M. Kotschenreuther, Physics of Plasmas **14**, 072502 (2007).

[2]D. D. Ryutov, Physics of Plasmas **14**, 064502 (2007).

[3]D. D. Ryutov, Physics of Plasmas **15**, 092501 (2008).

[4]B. LaBombard, Nuclear Fusion **55**, 053020 (2015).

[5]B. J. Braams, Ph.D. thesis, Rijksuniversitet Utrech (1986).

[6]P. N. Guzdar, J. F. Drake, D. McCarthy, A. B. Hassam, and C. S. Liu, “Three-dimensional fluid simulations of the nonlinear drift-resistive ballooning modes in tokamak edge plasmas,” Physics of Fluids B: Plasma Physics **5**, 3712–3727 (1993), https://doi.org/10.1063/1.860842.

[7]B. Scott, “Three-dimensional computation of drift alfvén turbulence,” Plasma Physics and Controlled Fusion **39**, 1635–1668 (1997).

[8]X. Q. Xu and R. H. Cohen, “Scrap-off layer turbulence theory and simulations,” Contributions To Plasma Physics **38**, 158–170 (1997).

[9]B. Dudson, M. Umansky, X. Xu, P. Snyder, and H. Wilson, “Bout++: A framework for parallel plasma fluid simulations,” Computer Physics Communications **180**, 1467 – 1480 (2009).

[10]F. Halpern, P. Ricci, S. Jolliet, J. Loizu, J. Morales, A. Mosetto, F. Musil, F. Riva, T. Tran, and C. Wersal, “The gbs code for tokamak scrape-off layer simulations,” Journal of Computational Physics **315**, 388–408 (2016).

[11]P. Tamain, H. Bufferand, G. Ciraolo, C. Colin, D. Galassi, P. Ghendrih, F. Schwander, and E. Serre, “The tokam3x code for edge turbulence fluid simulations of tokamak plasmas in versatile magnetic geometries,” Journal of Computational Physics **321**, 606–623 (2016).

[12]A. Stegmeir, D. Coster, A. Ross, O. Maj, K. Lackner, and E. Poli, “GRILLIX: a 3d turbulence code based on the flux-coordinate independent approach,” Plasma Physics and Controlled Fusion **60**, 035005 (2018).

[13]S. Ku, R. Hager, C. Chang, J. Kwon, and S. Parker, “A new hybrid-lagrangian numerical scheme for gyrokinetic simulation of tokamak edge plasma,” Journal of Computational Physics **315**, 467–475 (2016).

[14]M. Dorf and M. Dorr, “Progress with the 5d full-f continuum gyrokinetic code cogent,” Contributions to Plasma Physics 10.1002/ctpp.201900113.

[15]A. H. Hakim, N. R. Mandell, T. N. Bernard, M. Francisquez, G. W. Hammett, and E. L. Shi, “Continuum electromagnetic gyrokinetic simulations of turbulence in the tokamak scrape-off layer and laboratory devices,” Physics of Plasmas **27**, 042304 (2020), https://doi.org/10.1063/1.5141157.

[16]T. D. Rognlien, D. D. Ryutov, N. Mattor, and G. D. Porter, Physics of Plasmas **6**, 1851 (1999).

[17]S. Wiesen, Journal of Nuclear Materials **463**, 480 (2015).

[18]R. Simonini, Journal of Nuclear Materials **34**, 368 (1994).

[19]O. Pan, T. Lunt, M. Wischmeier, D. Coster, and U. S. and, “SOLPS-ITER modeling with activated drifts for a snowflake divertor in ASDEX upgrade,” Plasma Physics and Controlled Fusion **62**, 045005 (2020).

[20]A. Khrabry, V. Soukhanovskii, T. Rognlien, M. Umansky, D. Moulton, and J. Harrison, “Modeling snowflake divertors in mast-u tokamak using uedge code,” Nuclear Materials and Energy

26, 100896 (2021).

[21]M. V. Umansky, M. S. Day, and T. D. Rognlien, Numerical Heat Transfer, Part B **47**, 533 (2005).

[22]R. Marchand and M. Dumberry, "CARRE: a quasi-orthogonal mesh generator for 2d edge plasma modelling," Computer Physics Communications **96**, 232 – 246 (1996).

[23]A. Taroni, "The multi-fluid codes Edgeid and Edge2D: Models and results," Contribution to Plasma Physics **34**, 448 (1992).

[24]O. Izacard and M. Umansky, "Gingred, a general grid generator for 2d edge plasma modeling," (2017), arXiv:1705.05717 [physics.plasm-ph].

[25]J. P. Freidberg, *Ideal MHD* (Cambridge University Press, New York, 2014).

[26]D. D. Ryutov, M. A. Makowski, and M. V. Umansky, "Local properties of the magnetic field in a snowflake divertor," Plasma Physics and Controlled Fusion **52**, 105001 (2010).

[27]M. E. Rensink and T. D. Rognlien, "Mapping of orthogonal 2d flux coordinates for two nearby magnetic x-points to logically rectangular domains," Tech. Rep. LLNL-TR-731515 (Lawrence Livermore National Laboratory, Livermore, California, 2017).

[28]L. Lao, H. S. John, R. Stambaugh, A. Kellman, and W. Pfeiffer, "Reconstruction of current profile parameters and plasma shapes in tokamaks," Nuclear Fusion **25**, 1611–1622 (1985).

[29]W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. (Cambridge University Press, Cambridge, USA, 1992).

[30]P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İlhan Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "Scipy 1.0–fundamental algorithms for scientific computing in python," (2019), arXiv:1907.10121 [cs.MS].

[31]I. Enthought, "SciPy Interpolate GitHub repository," `https://github.com/scipy/scipy/tree/master/scipy/interpolate`.

[32]"scipy.optimize.root documentation," `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html`.

[33]A. Hindmarsh, "Odepack. a collection of ode system solvers," (1992).

[34]D. Baver, J. Myra, and M. Umansky, "Linear eigenvalue code for edge plasma in full tokamak x-point geometry," Computer Physics Communications **182**, 1610–1620 (2011).

[35]D. A. Baver, J. R. Myra, and M. Umansky, "ELM benchmark of the 2DX code," Tech. Rep. (Lodestar Research Corporation, 2010).

[36]M. Wiesenberger, M. Held, L. Einkemmer, and A. Kendl, "Streamline integration as a method for structured grid generation in x-point geometry," Journal of Computational Physics **373**, 370–384 (2018).

[37]M. Dorf et al., Bulletin of the American Physical Society **59** (2014), TP8.00027.

[38]M. A. Dorf, M. R. Dorr, J. A. Hittinger, R. H. Cohen, and T. D. Rognlien, "Continuum kinetic modeling of the tokamak plasma edge," Physics of Plasmas **23**, 056102 (2016), https://doi.org/10.1063/1.4943106.

[39]O. Meneghini, S. Smith, L. Lao, O. Izacard, Q. Ren, J. Park, J. Candy, Z. Wang, C. Luna, V. Izzo, B. Grierson, P. Snyder, C. Holland, J. Penna, G. Lu, P. Raum, A. McCubbin, D. Orlov, E. Belli, N. Ferraro, R. Prater, T. Osborne, A. Turnbull, and G. Staebler, "Integrated modeling applications for tokamak experiments with OMFIT," Nuclear Fusion **55**, 083008 (2015).

[40]O. Meneghini and L. Lao, "Integrated modeling of tokamak experiments with omfit," Plasma and Fusion Research **8**, 2403009–2403009 (2013).

[41]"UEDGE code GitHub repository," https://github.com/LLNL/UEDGE, accessed: 2021-01-18.

[42]K. Simonov, "Pyyaml homepage," https://pyyaml.org/wiki/PyYAML (2006).

[43]B. M. Garcia, J. Watkins, J. Guterl, and M. V. Umansky, "INGRID code GitHub repository," `https://github.com/LLNL/INGRID` (2019).

[44]B. M. Garcia, J. Watkins, J. Guterl, and M. V. Umansky, "INGRID Read the Docs," `https://ingrid.readthedocs.io/en/latest/` (2020).

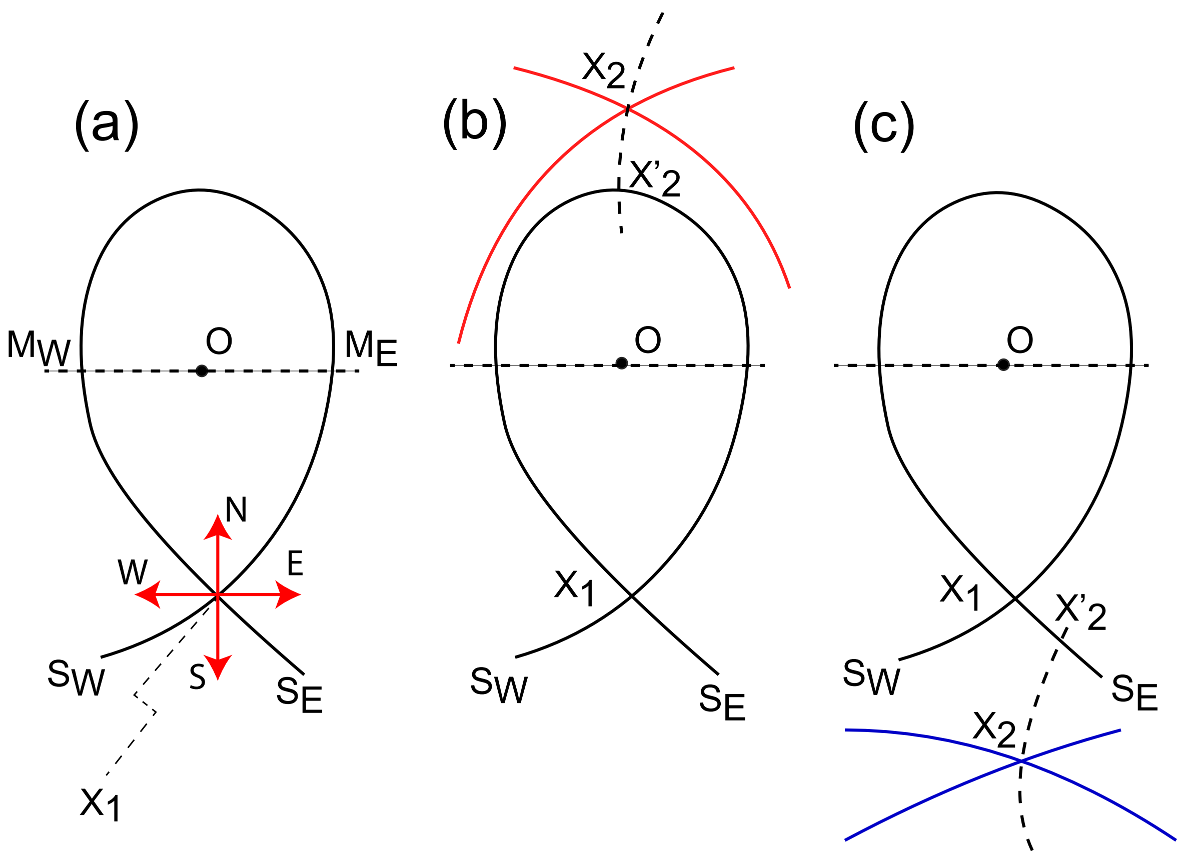


Figure 1: Topological possibilities with one and two X-points. As explained in the main text, case (a) is a single-null configuration; case (b) with the secondary X-point in the common-flux region has five variations, depending on the location of the projection point X_2'; case (c) with the secondary X-point in the private-flux region has two variations depending on the location of the projection point X_2'. The East-West notation for the two midplane points (M_E and M_W) and the two strike points (S_E and S_W) is based on designating the direction along $\nabla\Psi$ from the corresponding X-point toward the O-point as "North", as illustrated in case (a). This is invariant notation, independent on whether the X-point is at the top, at the bottom, or anywhere else.

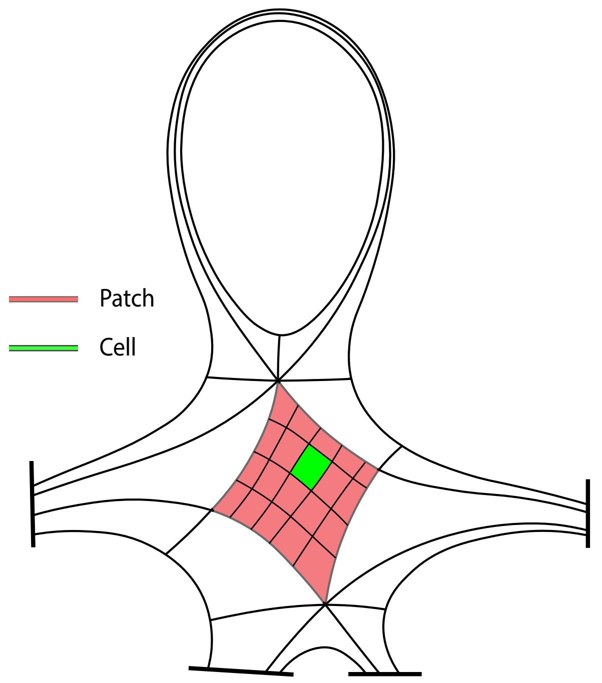


Figure 2: INGRID Patch Map; a subgrid is shown on one of the patches

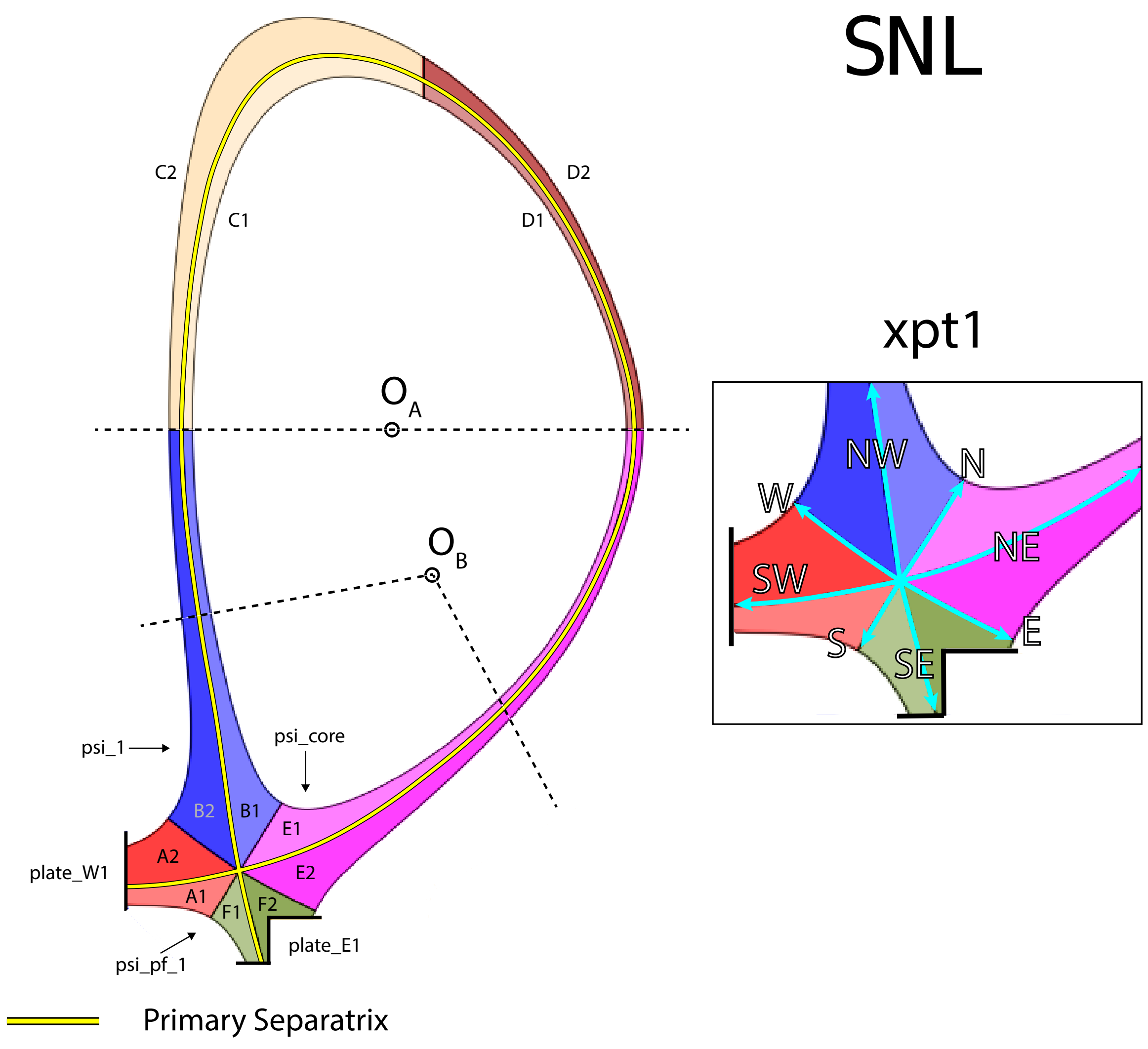


Figure 3: SNL Patch Map with the generalized midplane overlayed. O_A shows the original magnetic axis used to define the horizontal midplane through O_A. The midplane defines the poloidal boundary between patches B/C and D/E. For Patch Map customizability, INGRID allows for a generalized midplane through O_B to be defined (described in Section II-G). This is done by redefining the location of the magnetic axis and the directions of the rays generating the midplane.

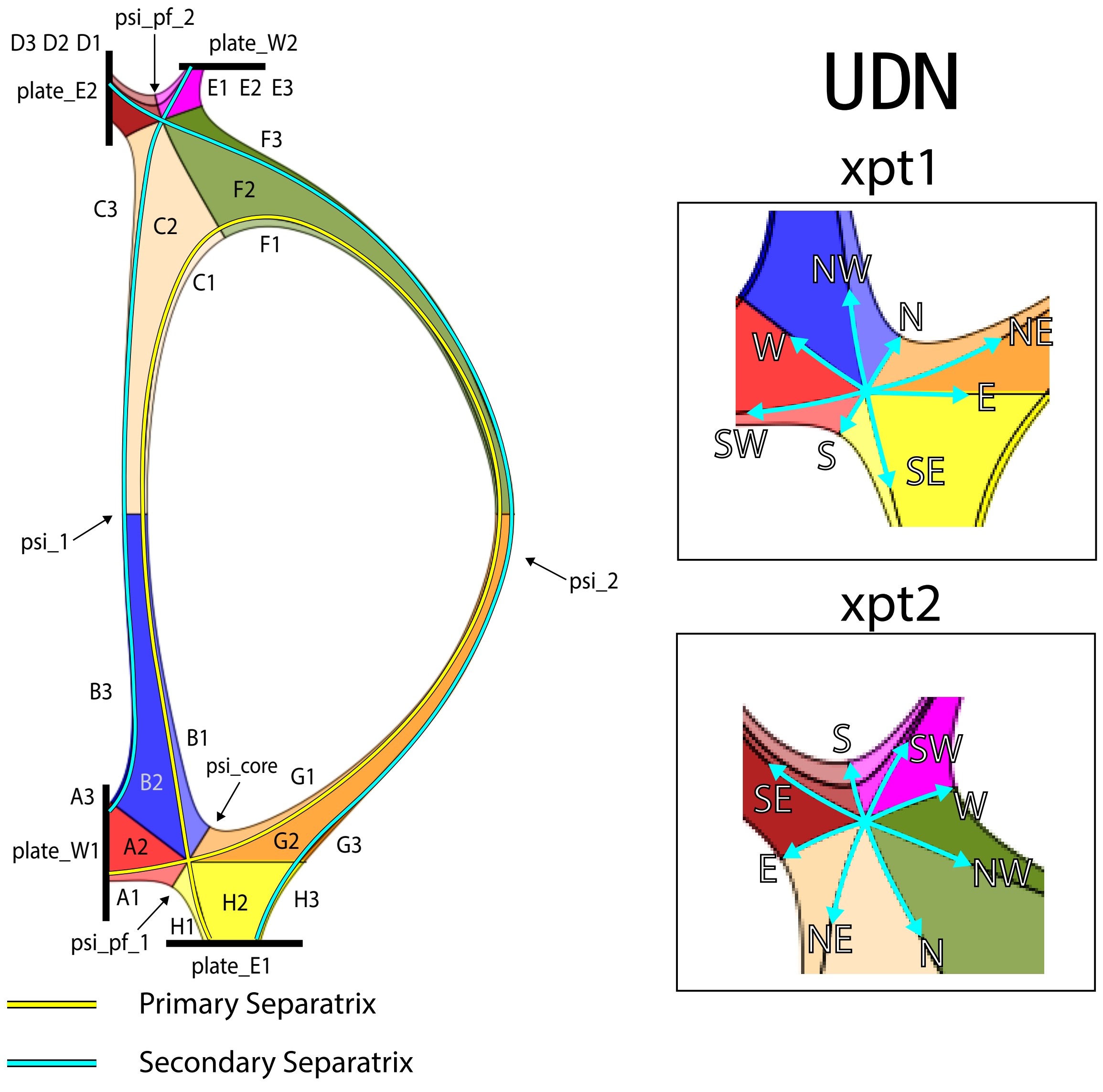


Figure 4: UDN Patch Map

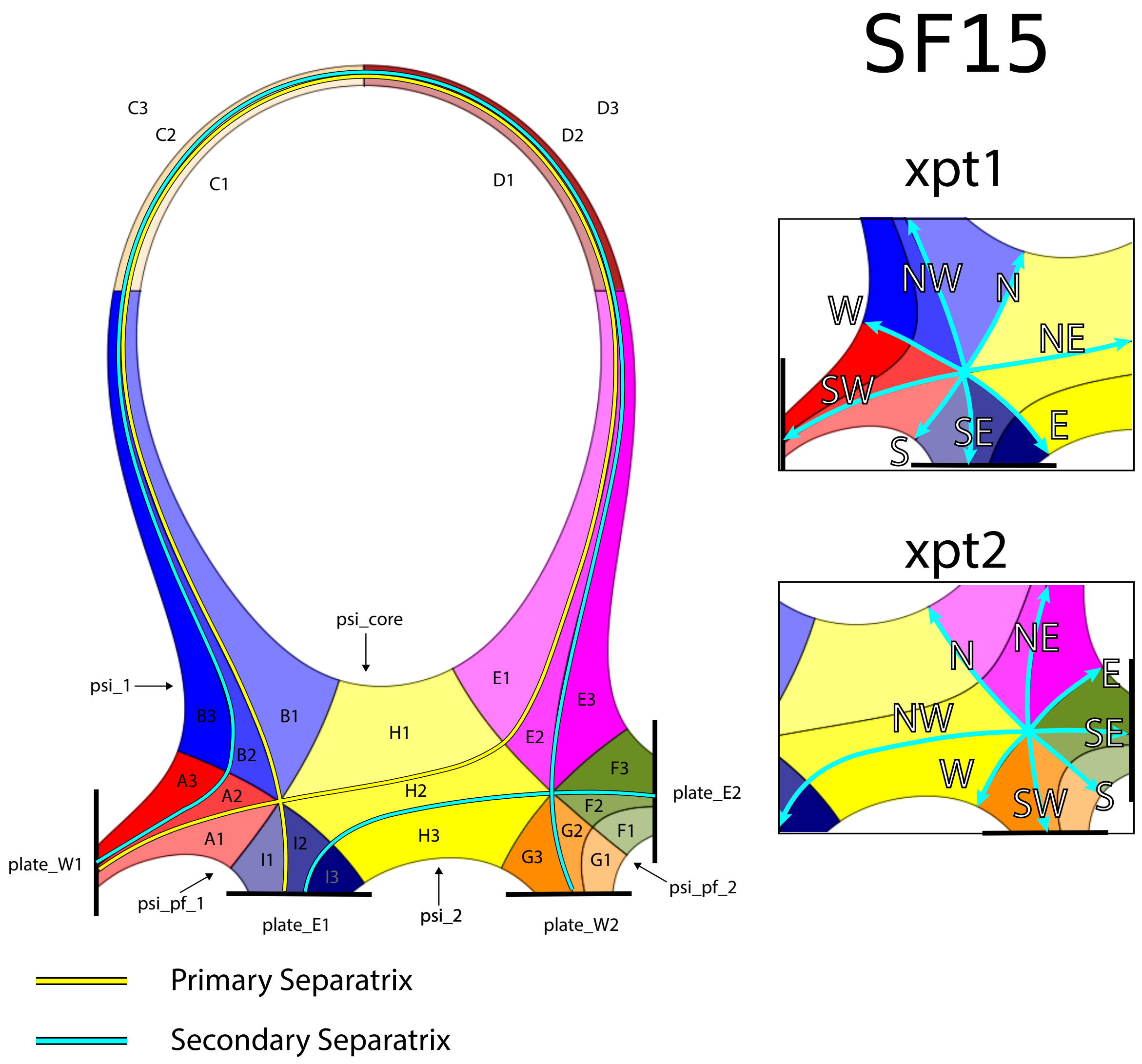


Figure 5: SF15 Patch Map

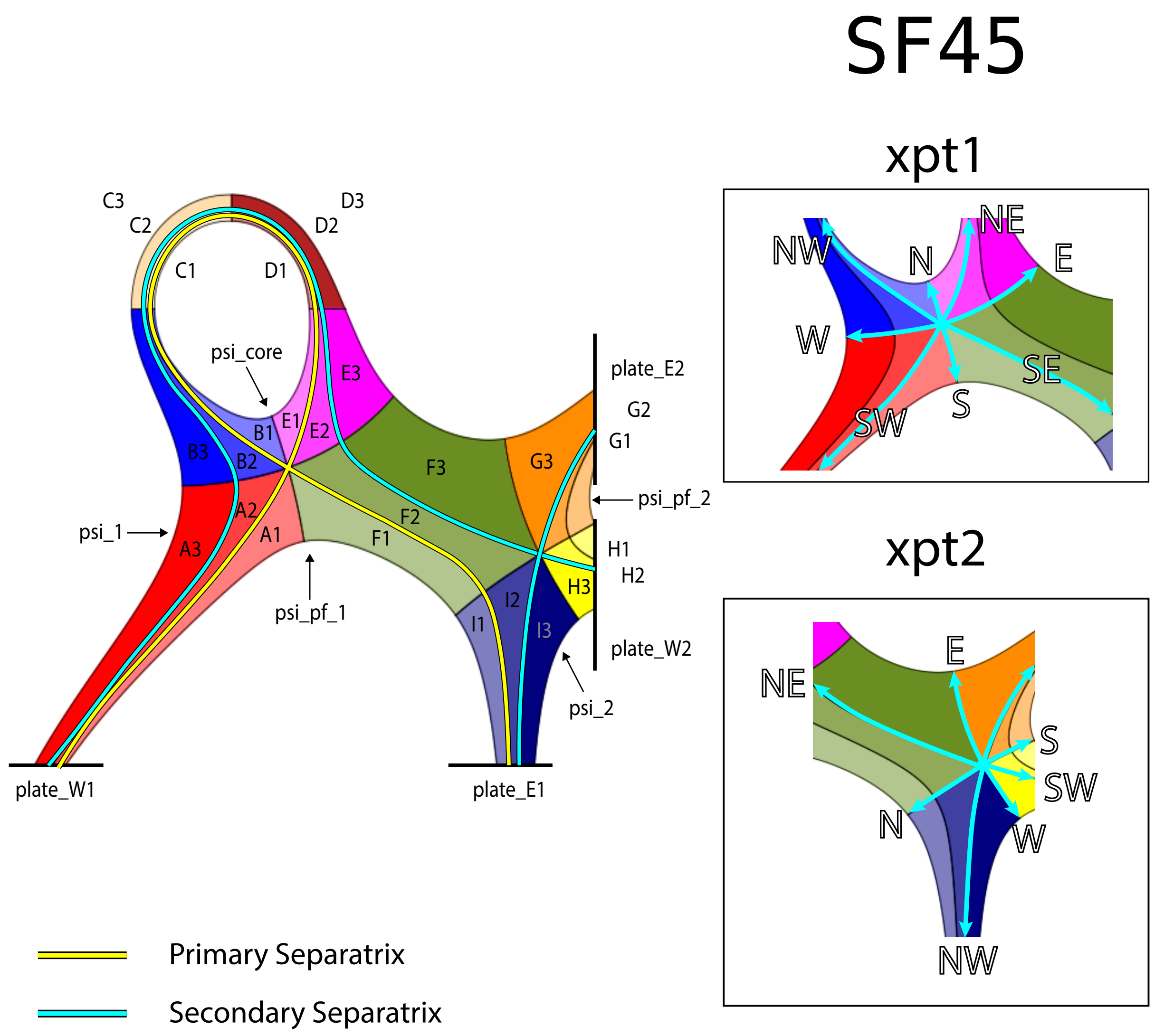


Figure 6: SF45 Patch Map

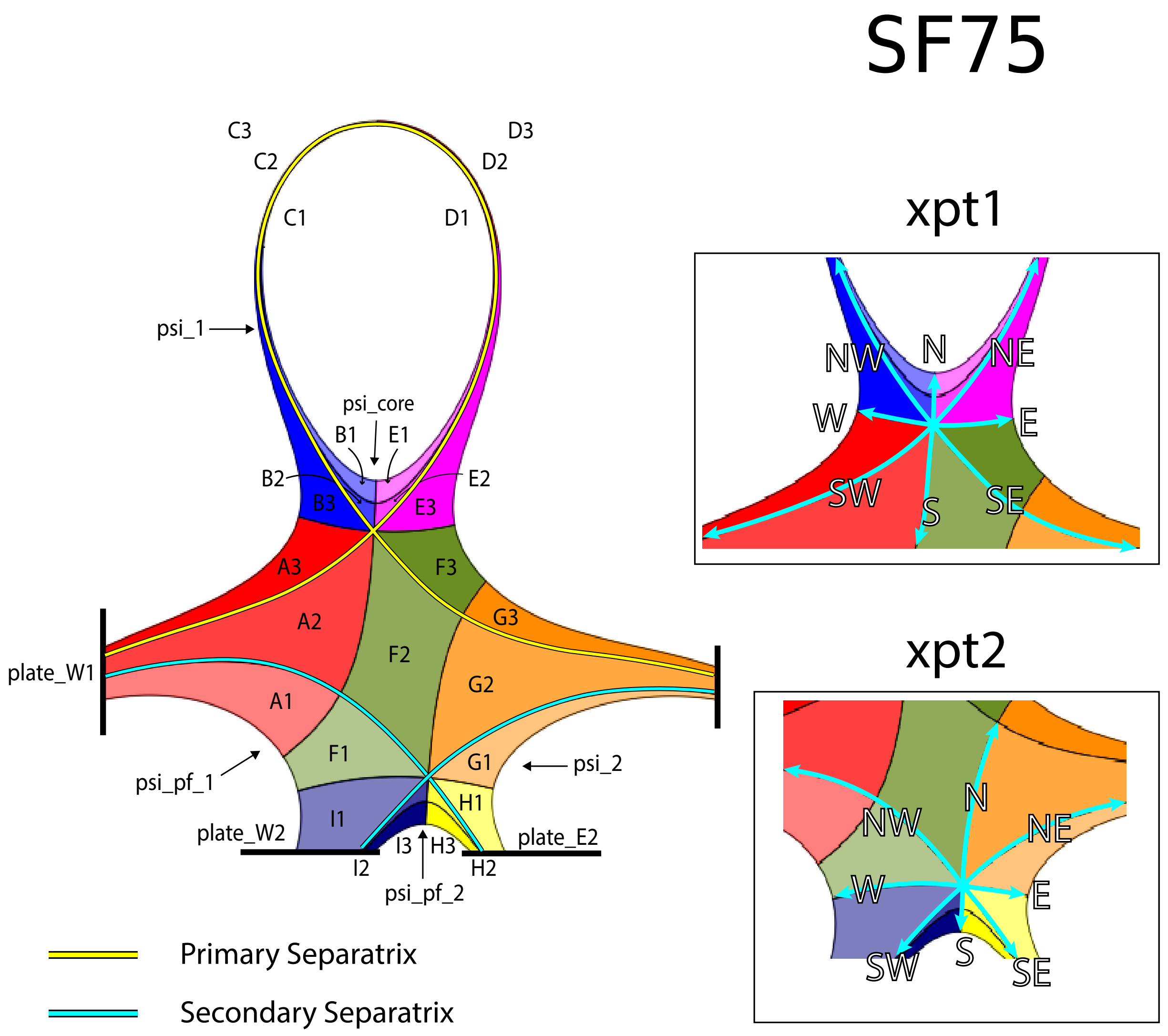


Figure 7: SF75 Patch Map

SF105

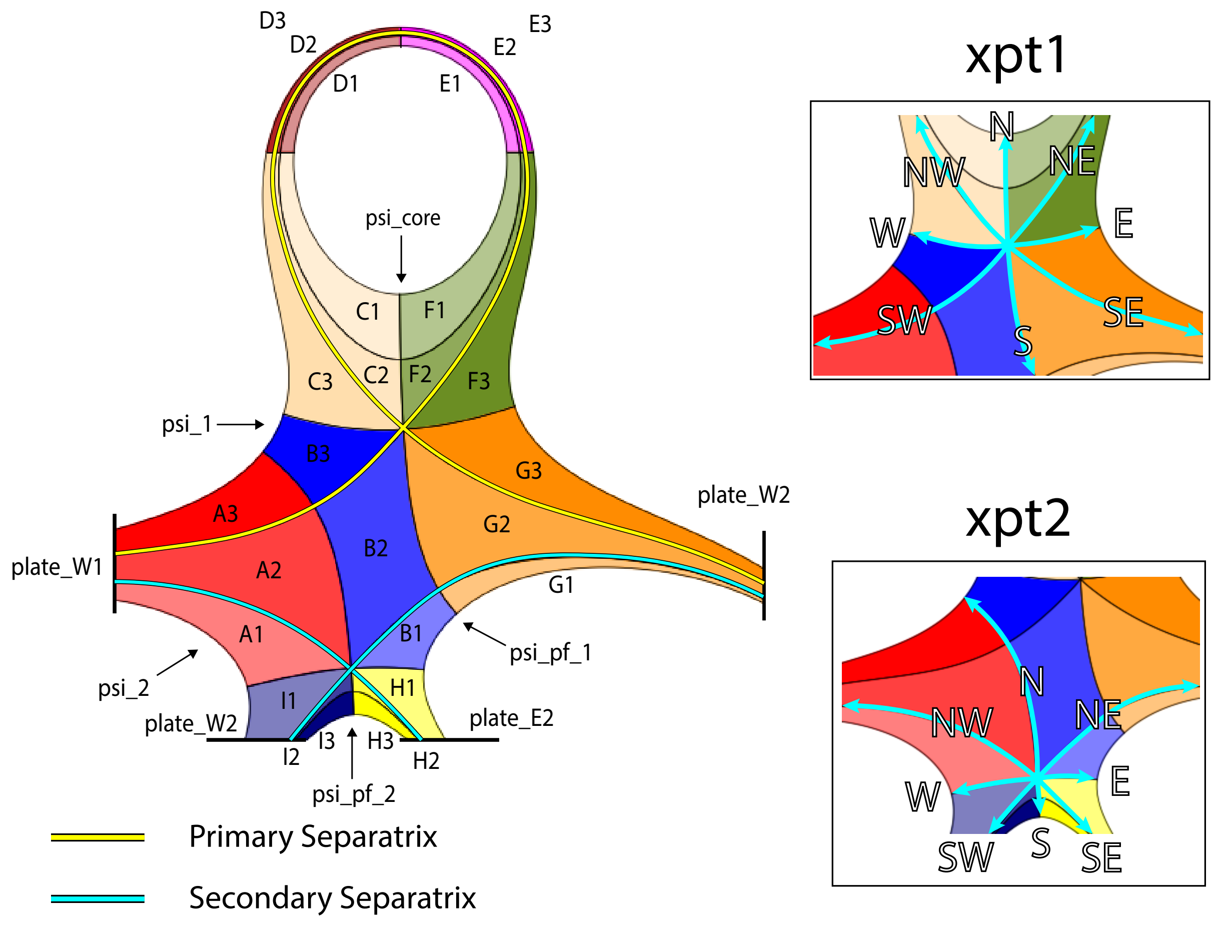


Figure 8: SF105 Patch Map

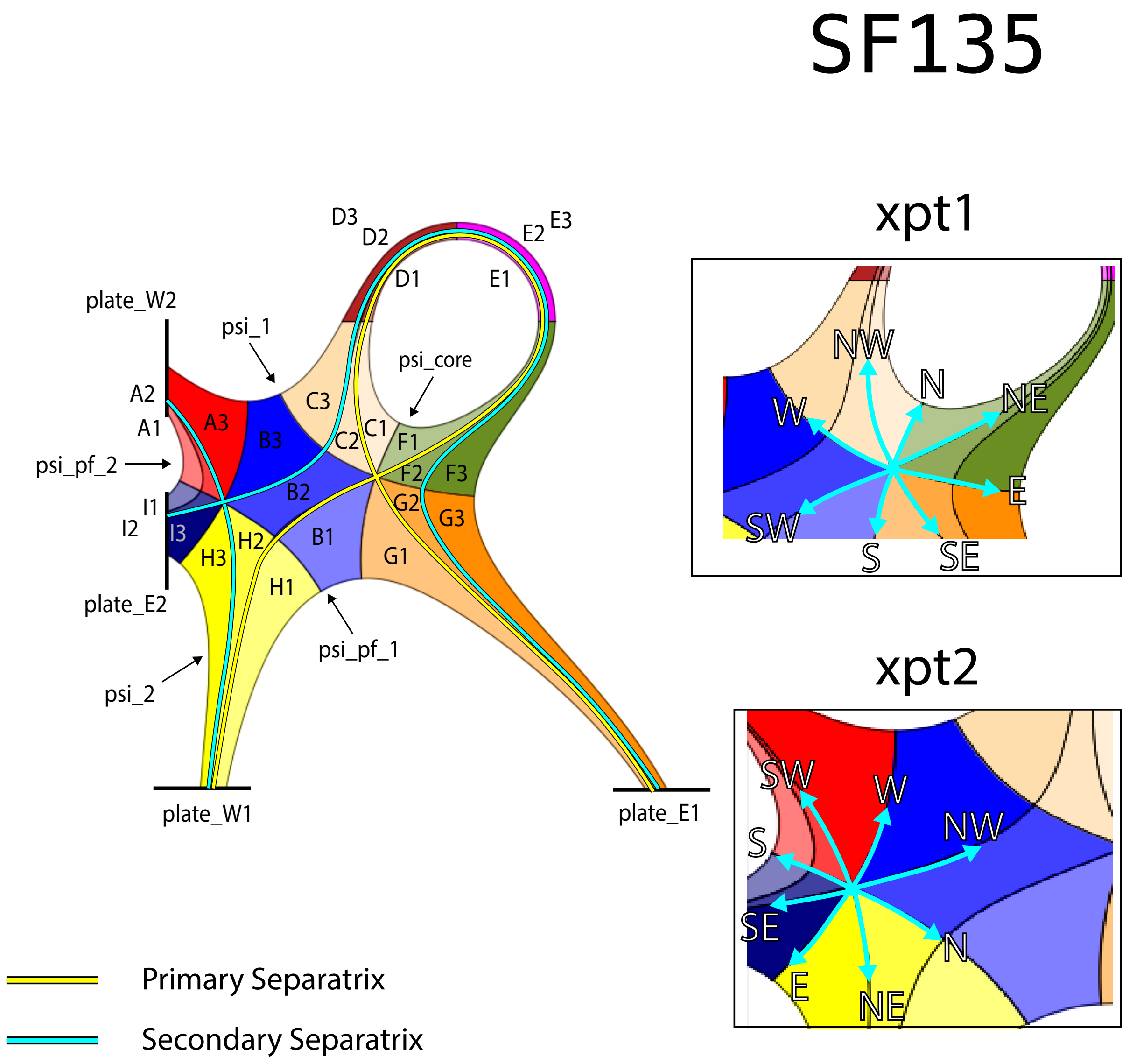


Figure 9: SF135 Patch Map

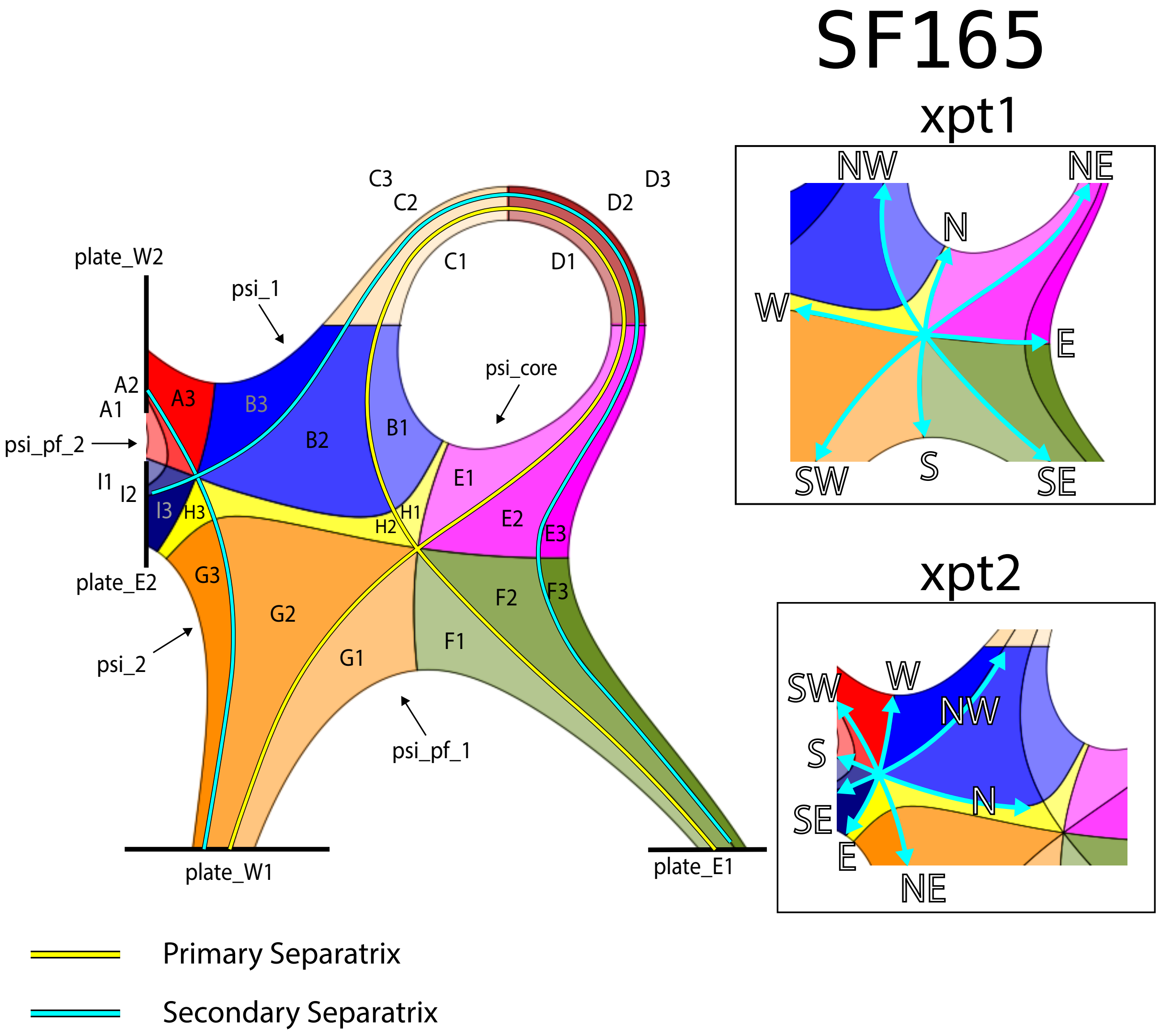


Figure 10: SF165 Patch Map

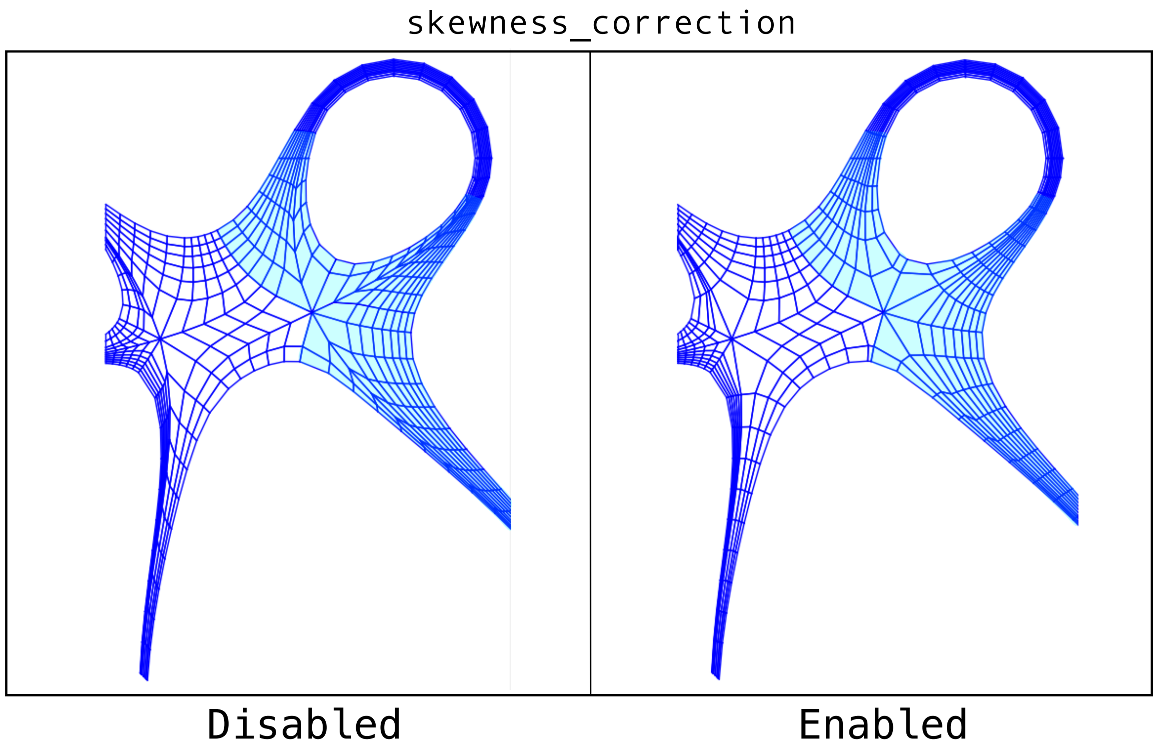


Figure 11: Comparison of SF135 grids generated with and without activation of the skewness_correction tool. Highlighted regions illustrate regions of notably improved grid orthogonality.

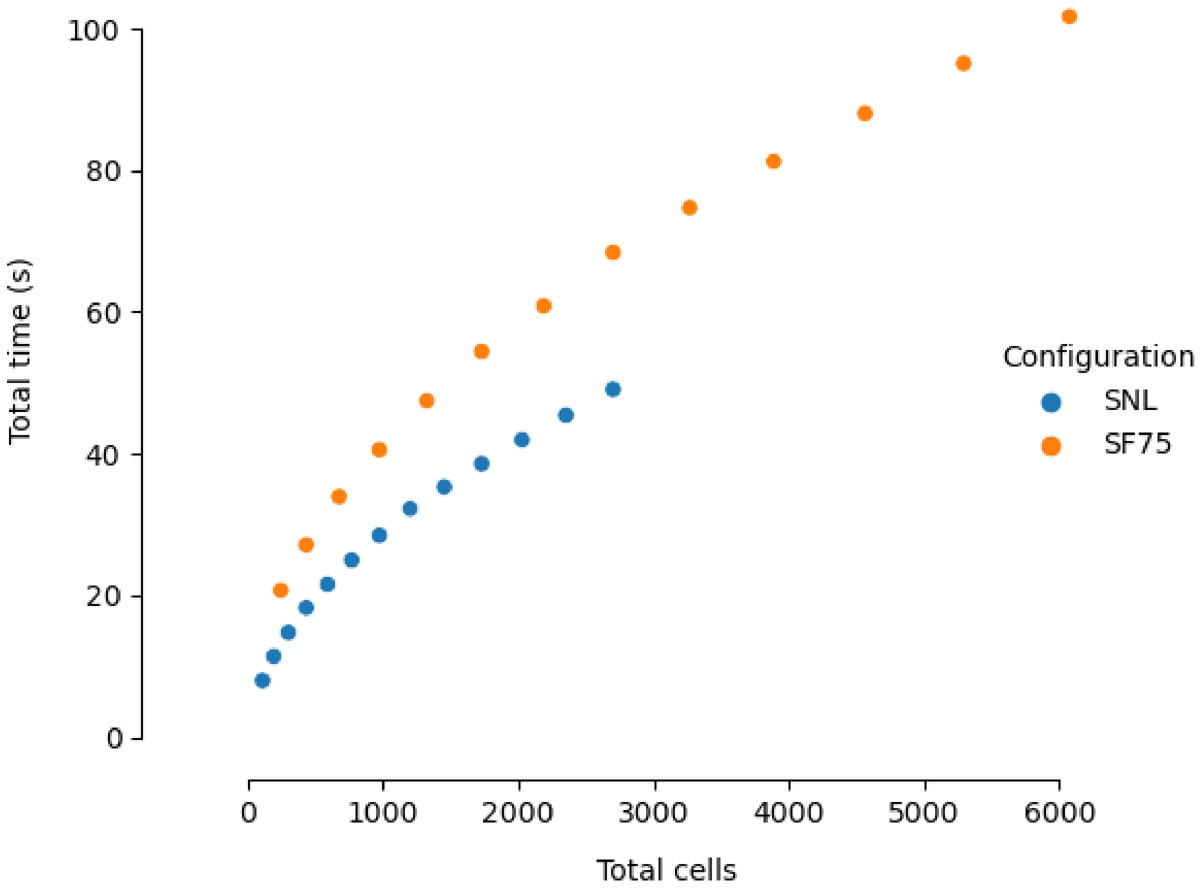


Figure 12: Scaling of grid generation follows a sublinear trend independent of configuration. Grids were generated with $n \times n$ many cells per Patch with $n = \{3, 4, 5, \ldots, 15\}$. With $n \times n$ subgrid dimensions, SNL configurations contain $12n^2$ many cells, whereas SF75 configurations contain $27n^2$ many cells.

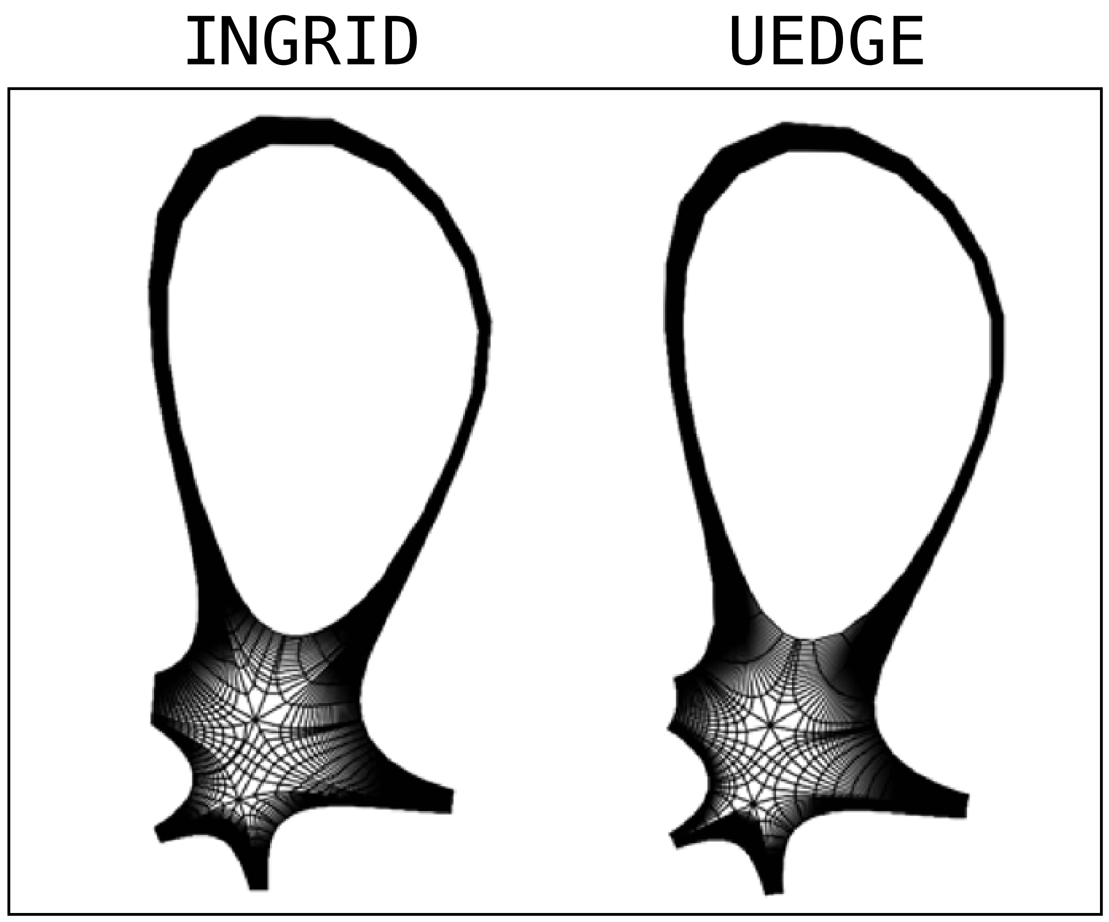


Figure 13: INGRID and UEDGE generated grids used for the comparison calculations.

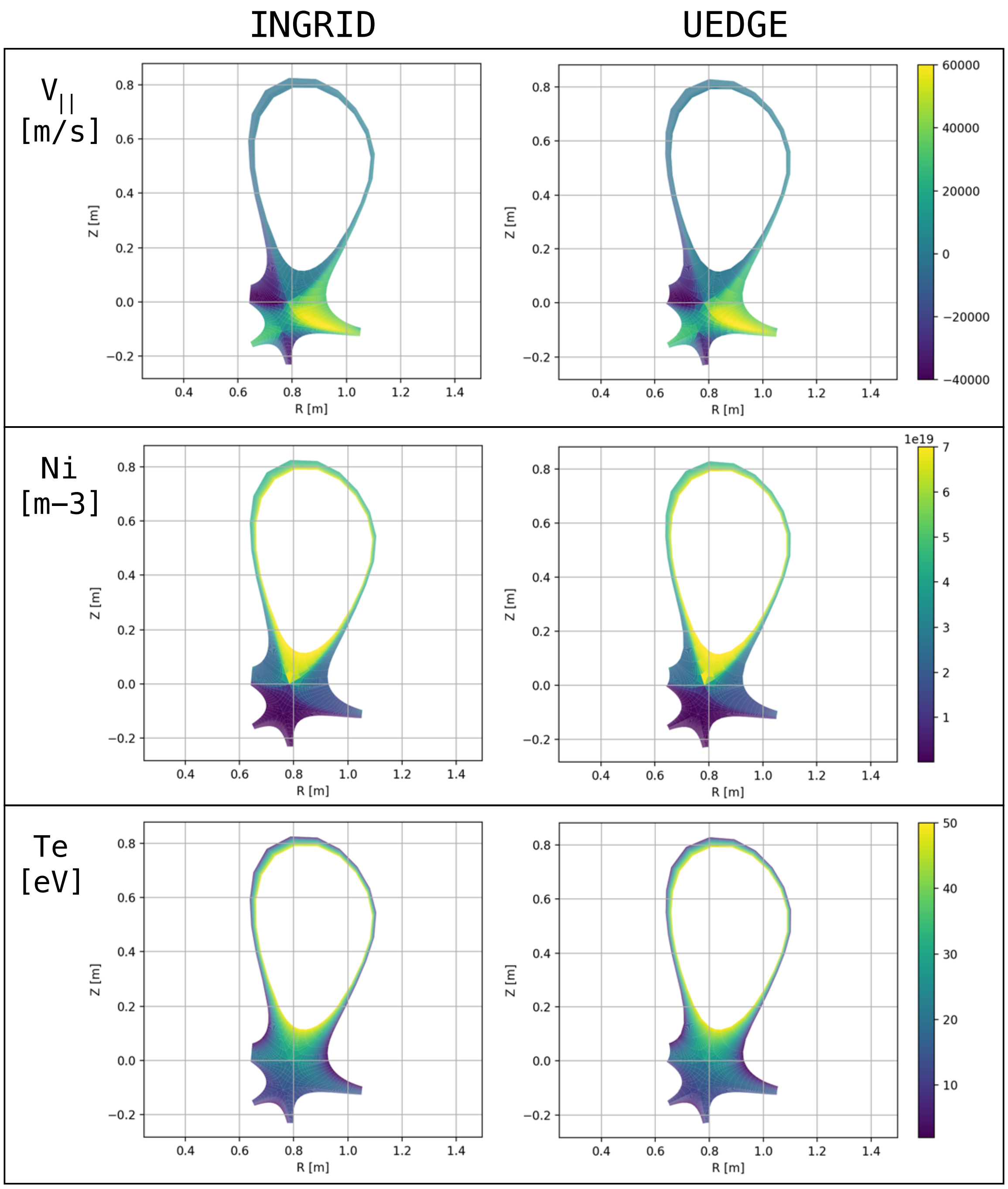
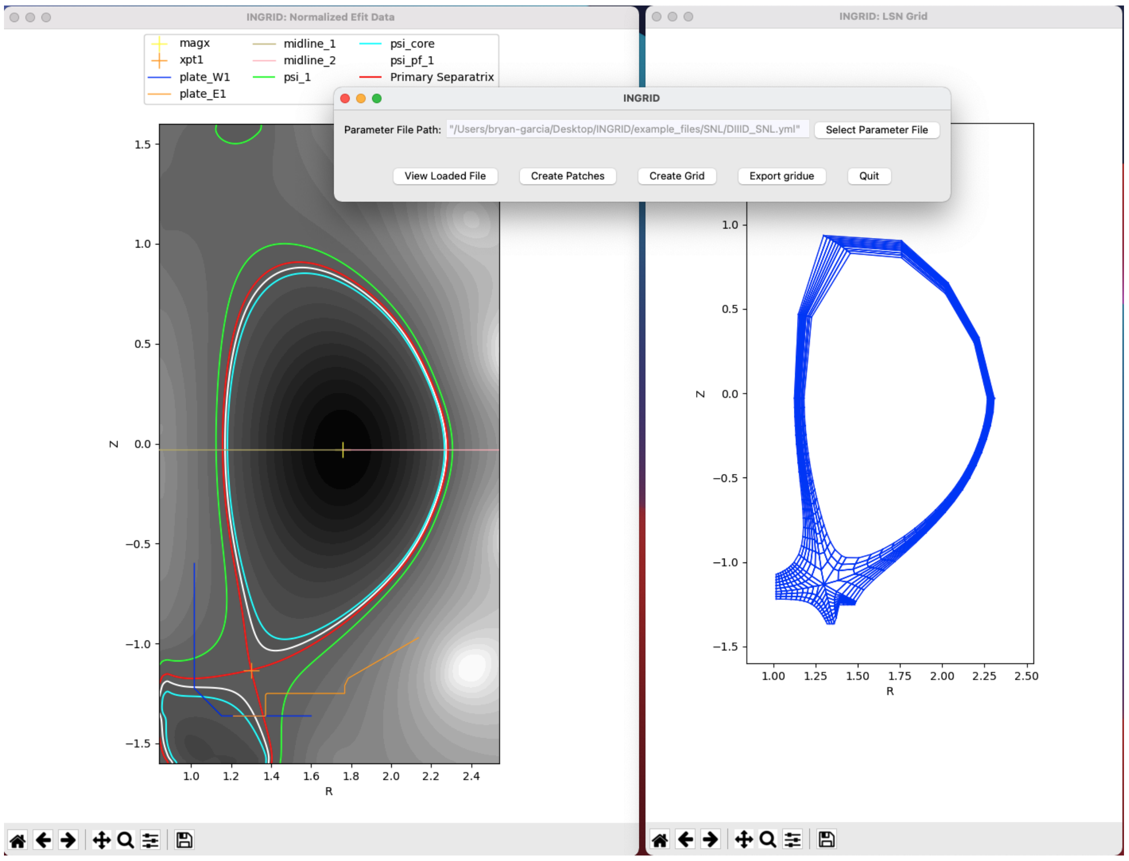


Figure 14: Results of the comparison calculations run on INGRID and UEDGE generated grids.



Parameter File Path: "/Users/bryan-garcia/Desktop/INGRID/example_files/SNL/DIIID_SNL.yml"

Figure 15: The INGRID GUI with loaded EFIT data and an INGRID generated grid plotted in separate windows. This interface allows users to load a parameter file, plot the contents, create a Patch Map, create a grid, and export a gridue.