ingrid

Bryan Max Garcia

Apr 30, 2021

CONTENTS:

INGRID (**IN**teractive **GRID**) is a Python based tokamak edge plasma grid generator capable of automatic generation of grids for magnetic-topologies with up to two x-points anywhere in the domain. The code can be operated in both a GUI mode and within scripts.

This documentation will get a user up and running with operating INGRID in GUI mode. For operating INGRID via scripts, API documentation has been generated (see Module Documentation).

The INGRID code is maintained by LLNL PLS-FESP. Issues with the code be brought to the attention of the current maintainers Bryan Garcia (UCSC), Maxim Umansky (LLNL), and Jerome Guterl (GA).

CHAPTER

ONE

INGRID INTRODUCTION

INGRID is an interactive grid generator for tokamak edge-plasma modeling capable of handling magnetic-configurations with two x-points in the computational domain.

INGRID analyzes EFIT data in order to classify the magnetic-topology, and generate the appropriate `gridue` formatted file for use in simulation code UEDGE.

Key features of INGRID include:

- Support for single-null, unbalanced double-null, and snowflake (15, 45, 75, 105, 135, 165) configurations
- Python based code with GUI and scripting usability
- UEDGE friendly
- Portable YAML parameter file driver
- Modular design pattern for continued development efforts

INGRID was developed at LLNL PLS-FESP by Bryan Garcia (UCSC), Maxim Umansky (LLNL), and Jerome Guterl (GA).

CHAPTER

TWO

GETTING STARTED

In this section, we will explain how to install INGRID and walk you through an example case that generates a grid for a single-null configuration. Examples will be showcasing the GUI operation of INGRID.

2.1 Downloading and installing INGRID

2.1.1 Requirements

To run INGRID on your machine, `anaconda3` and `setuptools` must be installed and up to date. `anaconda3` installers can be found here.

Tip: You can create a new conda environment with the command `conda create --name myenv` (replace `myenv` with the environment name).

Once the Anaconda package manager is installed, `setuptools` can be added to the conda environment by running:

```
conda install -c anaconda setuptools
```

To update `setuptools` run:

```
pip install setuptools --upgrade
```

2.1.2 Obtaining the code

Clone the INGRID repo with the command:

```
git clone username@https://github.com/LLNL/INGRID.git IngridDir
```

where `IngridDir` is the name of the destination clone directory.

2.1.3 Installing INGRID

Warning: Users **not** on MacOS Mojave may skip this warning. **Read on otherwise**. MacOS Mojave has issues with certain backend libraries used in INGRID. These issues have been documented by Apple. As a workaround, a specific Conda evironment has been created and must be installed by Mojave users. Navigate into the cloned repo locate the file `conda_env.yml`. Create the mentioned Conda environment by running `conda env create -f conda_env.yml`. Activate the new Conda environment by running `conda activate ingrid`. When active, the terminal prompt should begin with `(ingrid)`. The `ingrid` Conda environment must be active for the next section.

The user will install INGRID with the `setup.py` file provided in the cloned repo. Installation begins by running:

```
python setup.py install --user
```

2.1.4 Contents

Within the cloned repo are a variety of directories containing source-code, drivers, example/template files for controlling INGRID (will be discussed later), and data that the provided example-files/demos use.

We will be utilizing the directory `example_files` in our tutorials, and we encourage you to utilize the items in directory `template_files` for your own INGRID usage.

2.2 Launching the INGRID GUI

Now that INGRID has been installed on the machine, you can now begin utilizing INGRID in both GUI mode and as an importable module in Python. We will focus on using INGRID's stand-alone GUI for now.

Warning: As a reminder, MacOS Mojave users must ensure the conda environment provided with the INGRID code has been installed. The following information assumes the user has done so (see *Downloading and installing INGRID* if unsure).

2.2.1 Launching from `drivers`

Although INGRID can be imported and launched from a Python session, we will first explain how to launch INGRID from the cloned repo.

We recommend new users launch INGRID in this way since the `example_files` we will explore in the tutorials come pre-set with relative paths to necessary data (eqdsk files, geometry files, etc…).

Note: Once the basic controls for setting paths to data withing the INGRID parameter file is understood, the user should be able to utilize the provided example-files without starting from the `drivers` directory.

To launch via the driver script, navigate into the cloned repository that we used to install INGRID. From here, navigate into the directory `drivers` and run the command:

```
python StartGUI.py
```

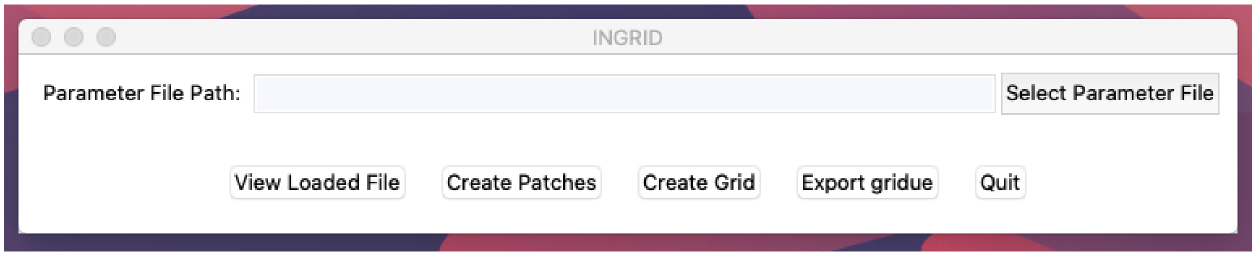
The INGRID GUI should now be visible, and ready for use.

2.2.2 Launching from a Python session

For users who know do not intend on using the YAML files within `example_files`, or those who understand how to set paths within the INGRID parameter file, you can launch the INGRID GUI from a Python prompt as follows:

```
>>> from INGRID import ingrid
>>> ingrid.QuickStart()
```

After executing the above commands, the INGRID GUI should be visible and ready for use (seen below).



Note: Upon launching the INGRID GUI, the user will be prompted with a plenty of terminal output. This is nothing to be alarmed of since this part the `Ingrid` class' initialization. As we load parameter files and modify settings, we should see appropriate changes reflected in the terminal output.

2.3 The INGRID parameter file

2.3.1 Background

INGRID has been designed to be controlled from a single configuration/parameter file when operating in GUI mode. This specially formatted YAML file is similar to Fortran namelist files due to the *key-value* structure it contains.

Here is a snippet of what a YAML formatted file can look like.

```
# Comments are supported and follow python convention!
# YAML entries are a mapping of the form:
#
#   key: value
#
# It follows that python interprets the YAML file as a dict

my_YAML_entry:
    # YAML file use spaces as indentation.
    # 2 or 4 spaces (pick one and stick to it) indicate a nested item.
    # Here we use 4 spaces.
    my_str_key: 'Hello, YAML!'  # my_str_key = 'Hello, YAML!'
    my_int_key: 32  # my_int_key = 32
    some_float: 3.14159  # some_float = 3.14159
    my_true_bool: true # bool case-insensitive
    my_false_bool: 0  # '1' '0' bool values supported.

    # Empty lines within file are ok.

    # Just remember it's the spaces that matter.
```

(continues on next page)

(continued from previous page)

```
    my_sub_dict: # Nested dicts are supported.
        another_key: ending my example code-block here.
        # The above value entry will be interpreted as a string
        # (no quotations needed)
```

Although the INGRID parameter file contains many controls for a user, it does not stray from the patterns illustrated above (no advanced YAML knowledge required).

Tip: While operating INGRID in GUI mode, keep your favorite text-editor handy with the parameter-file in use loaded. You will be making frequent edits to this core parameter/configuration file.

Although not necessary for following tutorial, detailed documentation of the INGRID parameter-file can be found here.

2.3.2 A single-null example file

Users of INGRID have plenty of controls available for fine-tuning their final grid. This section explains how to navigate some key controls within the INGRID parameter file.

We will be using the pre-populated INGRID parameter file `DIIID_SNL.yml` from the `example_files` directory in the single-null walk-through. Open this file in your preferred text-editor. At the time of writing this documentation, the parameter file `DIIID_SNL.yml` contains the following:

```
# ---------------------------------------------------
# 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, distortion_correction)
  # ---------------------------------------------------------------------------
  grid_generation:
    distortion_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
```

(continues on next page)

(continued from previous page)

```
    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
```

(continues on next page)

(continued from previous page)

```
  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
```

Let's highlight some important entries that are often used when operating INGRID for single-null cases (basic usage). Advanced tutorials will also be provided.

2.3.3 Setting data paths

A user can provide a string that indicates the path to certain data. This is used to tell INGRID where to look for EFIT data, target plate coordinate, limiter coordinates, and patch-data (for reconstruction). We can set these paths by editing the entry:

```
# ---------------------------------------------------
# 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
```

Note: INGRID supports both absolute paths and paths relative to where INGRID has been launched.

If `dir_settings` is missing any entries, INGRID will (internally) set the missing values to a default value of `'.'` (current working directory). This holds even if `dir_settings` is omitted from the parameter file.

Note: `dir_settings` entries are the **directory** to look for data and NOT the file itself.

2.3.4 Providing an EQDSK file

The user provides the actual EQDSK file name separate from the `dir_settings` entry. We provide this at the global YAML level under entry `eqdsk`. That is:

```
# ---------------------------------------------------
# eqdsk file name
# ---------------------------------------------------
eqdsk: neqdsk
```

Note: In this example, INGRID searches for the file `neqdsk` within the directory `../data/SNL/DIII-D/` (relative to the launch point) since `dir_settings['eqdsk']` was set to `../data/SNL/DIII-D/` (see above).

2.3.5 Defining target plates

All target plate settings are under the global INGRID parameter file entry `target_plates`. We see this as:

```
# ---------------------------------------------------
# target plate settings
# ---------------------------------------------------
target_plates:
  plate_E1:
    file: d3d_otp.txt
    zshift: -1.6
  plate_W1:
    file: d3d_itp.txt
    zshift: -1.6
```

INGRID adopts a N-S-E-W compass direction notation in order to help generalize and simplify grid generation. It is important for a user to eventually learn these conventions. A detailed discussion of INGRID's naming conventions can be found here.

For now (in the case of a lower single-null configuration), note that entries `plate_E1` and `plate_W1` correspond to the *outer* and *inner* target plates, respectively. Each plate entry recognizes sub-entries `file` (file name to load), `zshift` (z-translation) and `rshift` (r-translation, not utilized and internal to INGRID defaults to `0.0`).

2.3.6 Defining x-points, magnetic-axis, and psi-levels

Settings for x-point coordinates, magnetic-axis coordinates, and psi-levels are found under the global INGRID parameter file entry `grid_settings`.

```
# ---------------------------------------------------
# General grid settings
# ---------------------------------------------------
grid_settings:
  # ...
  # ...
  # Other items currently not of interest...
  # ...
  # ...

  # ---------------------------------------------------
  # num xpts
```

(continues on next page)

(continued from previous page)

```
# ---------------------------------------------------
num_xpt: 1

# ---------------------------------------------------
# Psi levels
# ---------------------------------------------------
psi_1: 1.066  # SOL
psi_core: 0.95  # CORE
psi_pf_1: 0.975  # PRIVATE-FLUX

# ---------------------------------------------------
# magx coordinates
# ---------------------------------------------------
rmagx: 1.75785604
zmagx: -0.0292478683

# ---------------------------------------------------
# primary xpt coordinates
# ---------------------------------------------------
rxpt: 1.300094032687
zxpt: -1.133159375302
```

Warning: The entry `num_xpt` is one of the most important entries in the INGRID parameter file since it determines INGRID's method of analysis. Dealing with more than one x-point requires a more in-depth understanding of the parameter file, so ensure this is set to the correct number of x-points.

2.4 Example: single-null configuration (introduction)

Warning: This tutorial assumes INGRID has been launched from the provided GUI driver. See page *Launching the INGRID GUI* for explaination.

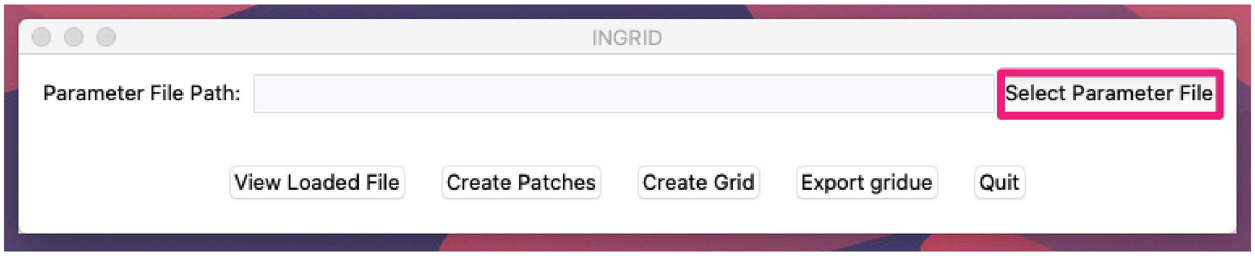
Here we will demonstrate how to generate a grid for a lower single-null (`SNL`) configuration. This tutorial aims to:

- Explain the GUI capabilities
- Illustrate the INGRID workflow (data analysis → patch-generation → grid-generation → exporting gridue)
- Expose the user to key parameter-file controls (see parameter-file documentation for further details)

Note: Although we are creating a lower single-null grid here, INGRID internally treats both lower and upper single-null configurations as `SNL` class instances. This means there is *no difference in user operation for generating a grid for either LSN or USN configurations.*

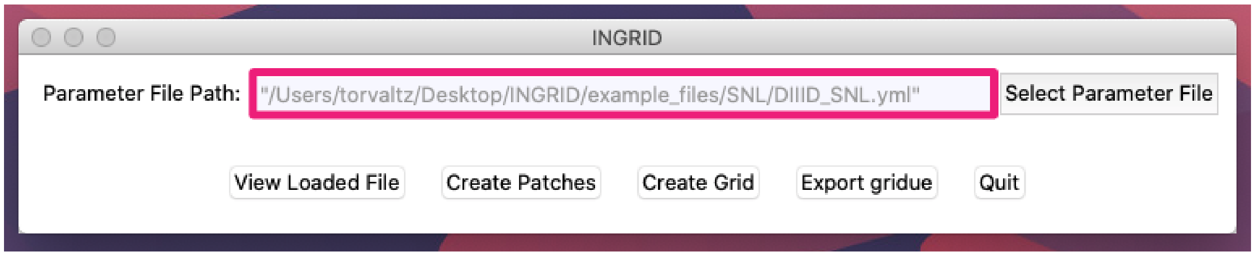
2.4.1 Loading our first example

After getting the INGRID GUI up and running, click the GUI button labeled “Select Parameter File” shown below boxed in red.



From here, your machine’s native file navigator should be on the screen. Navigate into the directory `example_files` the cloned repository to find a collection of example-cases we used for showcasing INGRID’s capabilities (as well as for testing the product).

Now navigate into directory `SNL` and select the file `DIIID_SNL.yml`. The GUI should now be updated with the loaded path to the example-file we selected as seen below boxed in red.



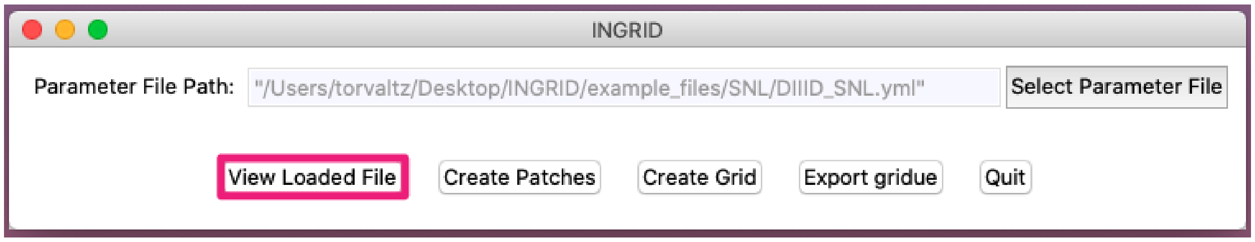
INGRID has now processed the selected parameter-file. Some (of many) actions executed automatically by INGRID include:

- Processing of paths to data (`EFIT`, geometry, patch-data, etc.)
- Loading of `EFIT` data
- Loading of strike-geometry (target plates and/or limiter)
- Refining of x-point coordinates
- Refining of magnetic-axis coordinates
- Initialization of visualization settings

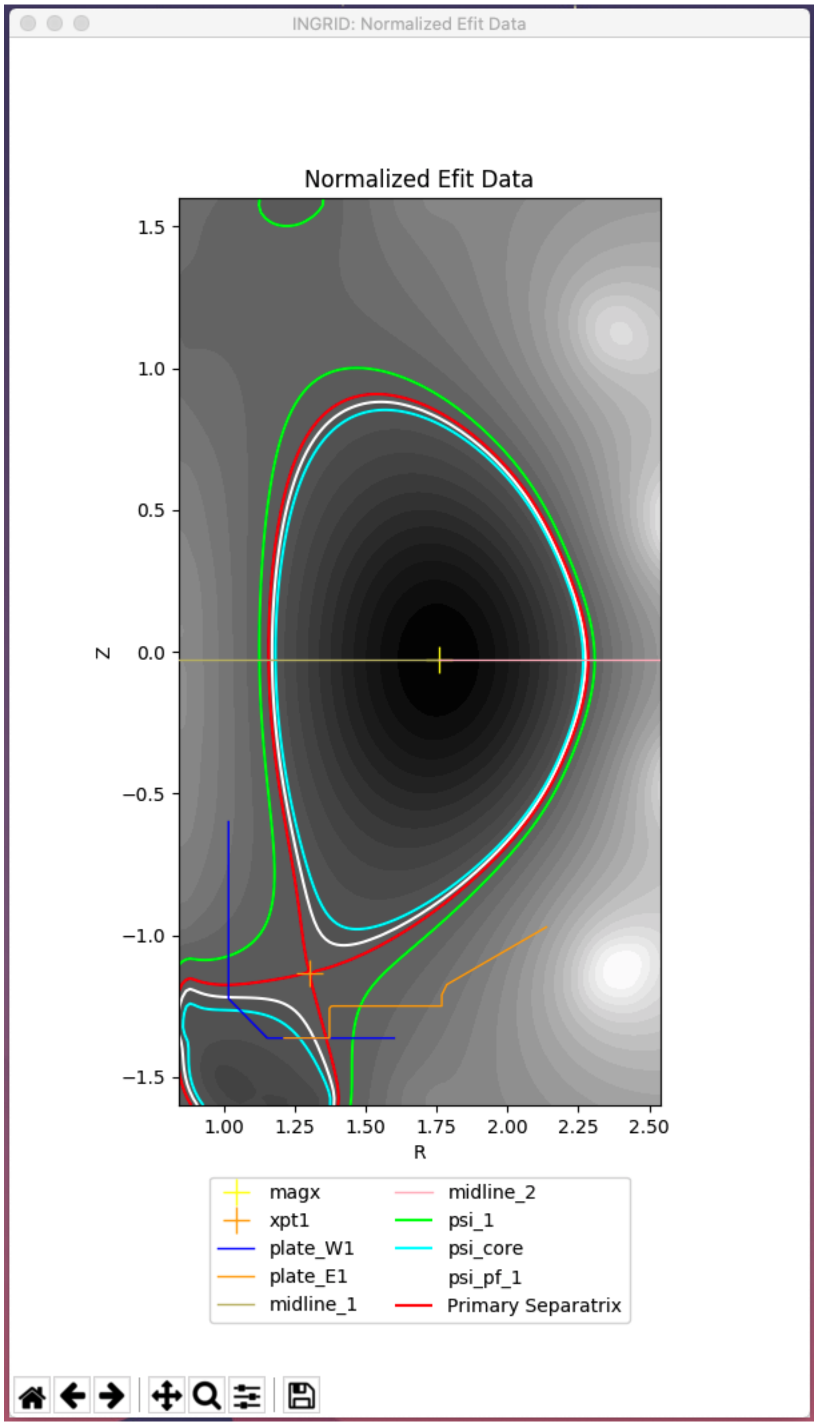
With the data loaded, we can now proceed.

2.4.2 Viewing loaded data

To view the `EFIT` data, loaded strike-geometry, and psi-levels that will dictate our final grid, simply select “View Loaded File” shown below boxed in red.



Once clicked, we are greeted with a new plot window showing the DIII-D data we have loaded.



Here are some key items that INGRID has plotted (as seen in the legend):

- **Refined** primary x-point coordinate as an orange '+' marker (`xpt1`)
- **Refined** magnetic-axis coordinate as a yellow '+' marker (`magx`)
- **Normalized** `eqdsk` data as black and white filled contours (psi-value of 0 at `magx` and psi-value of 1 at `xpt1`)
- `plate_W1` data as a dark blue line (LSN inner target plate)

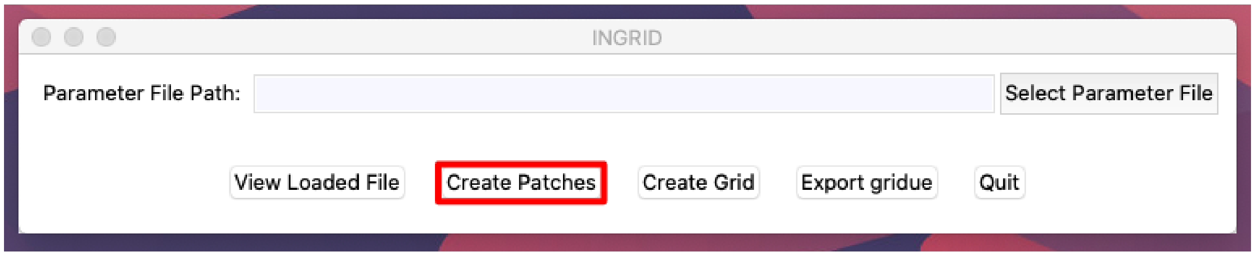
- `plate_E1` data as an orange line (LSN outer target plate)
- The primary separatrix (red contour line)
- SOL boundary (lime contour line)
- CORE boundary (cyan contour line)
- PRIVATE-FLUX boundary (white contour line)

Note: The user provides the approximate primary x-point coordinates (`rxpt1`, `zxpt1`), and magnetic axis coordinates (`rmagx`, `zmagx`) in the INGRID parameter file. INGRID takes these as an initial guess to provide to a root-finder in order to **refine** the user-provided coordinates to high-accuracy. These values are used internally throughout the user-session.

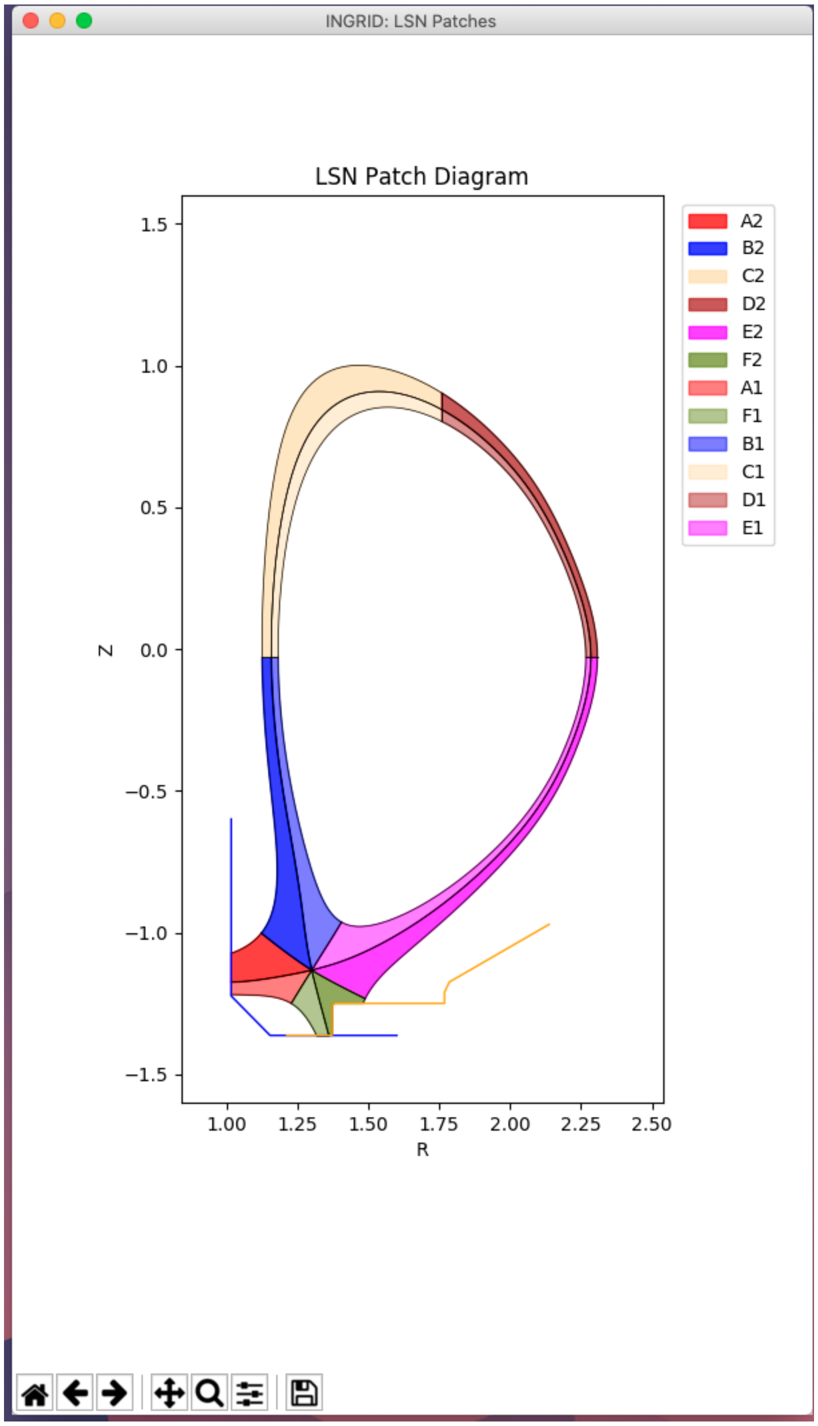
This stage is where the user will interact the most with the INGRID parameter file (tweaking psi-values, target-plate locations, limiter data, etc). Said settings will be used to generate the patch map we will see in the next section. Since these have already been provided for you, let us proceed to creating patches.

2.4.3 Creating patches

INGRID interally uses a geometry object hierarchy (`Point` ∈ `Line` ∈ `Patch` ∈ `TopologyUtils`) to generate the final gridue file. We will now create a collection of `Patch` objects. These `Patch` objects are quadrilaterals that form a *partition* of the region we are interested in generating a grid for. Before elaborating further, let us now create said collection of patches (referred to as a *Patch map*) by clicking the GUI button labeled "Create Patches" shown below boxed in red.



Once clicked, INGRID begins line-tracing in order to generate the Patch map seen below.



The collection of Patch objects are pictured in the Patch map. These Patch objects will generate their own subgrid that will be stitched together to form the exported global grid.

2.4.4 Saving `Patch` data

INGRID provides the user the capability of saving `Patch` data into a specially formatted NumPy `npy` files for later reconstruction. We control this feature within the parameter file by modifying the entries under `patch_data` (seen below):

```
# ---------------------------------------------------
# Saved patch settings
# ---------------------------------------------------
patch_data:
  file: LSN_patches_1597099640.npy
  preferences:
    new_file: true
    new_fname: LSN_patches_1597099640.npy
  use_file: false
```

Here we have:

- `file` - the name of the file to use for `Patch` reconstruction
- `preferences` - settings for configuring final `Patch` data file
  - `new_file` - create a new `Patch` data file
  - `new_fname` - name of new `Patch` data file
- `use_file` - use the provided file for `Patch` reconstruction

Note: Remember to set the directory to search for a `Patch` data *file* by modifying `patch_data` under entry `dir_settings`

Because the parameter file is populated with the above settings, we see that after creation of a `Patch` map the terminal prompts the user with a message stating:

```
# Saved patch data for file LSN_patches_1597099640.npy
```

The user is encouraged to try this feature out. To do so, first change the `use_file` entry within `patch_data` to a value of `True` and save the file. Now, close the `Patch` map window, and click `Create Patches` again. The `Patch` map should now be restored back to the state it was at.

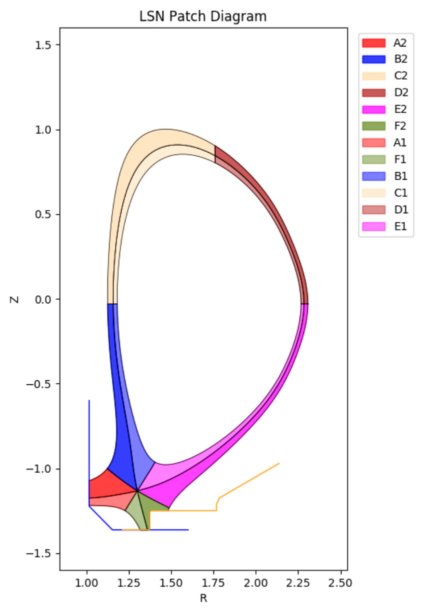
Tip: `Patch` data files expedite the grid generation process by bypassing all line-tracing. This feature is also useful for trading cases with other INGRID users

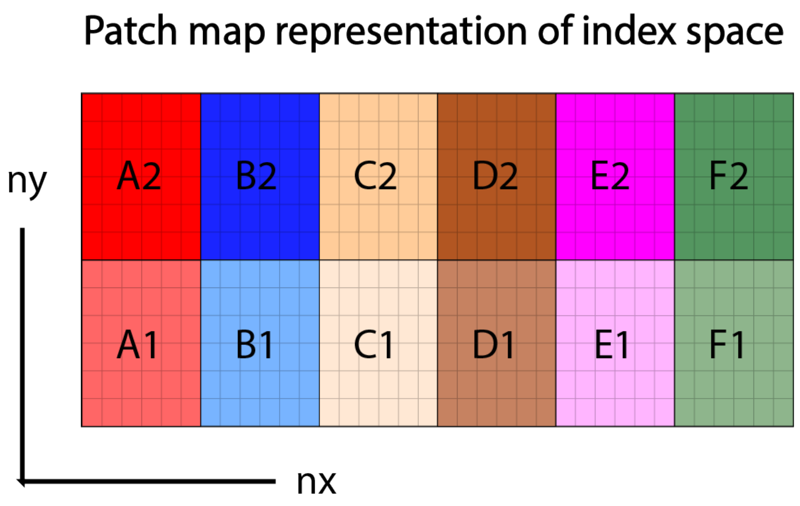
2.4.5 The `Patch` map

In the above plot we can see the "`Patch` map". Each `Patch` is been assigned it's own color, as well as a `Patch` label/tag consisting of a two-character string of the form "<alpha_char><numeric_char>"". This coding directly represents the index space of the final grid with:

- The alpha-char ("A", "B", . . . , "F" here) representing a poloidal "column" in the index space.
- The numeric-char ("1" and "2" here) representing a radial "row" in the index space.

Below is diagram illustrating said notation.





This notation proves to be robust since it holds for not only `SNL` topologies (both LSN and USN), but also all topologies such as `UDN` and the family `SF*`.

For the `SNL` family of configurations, the collection of `Patch` objects with numeric_char == "2" ("**A2**" - "**F2**") represent the SOL, `Patch` objects "**A1**" and "**F1**" represent the PF region, and `Patch` objects "**B1**", "**C1**", "**D1**", and "**E1**" represent the CORE.

Note: We will use this notation extensively for fine-tuning the final grid

Tip: `Patch` objects are ordered alphabetically clock-wise around the magnetic-axis and enumerated in direction of increasing psi

Now that we have partitioned the EFIT domain into the region we wish to model, let us now generate a grid.

2.4.6 Creating a grid

Before generating a grid, let's take a look at some grid controls in the INGRID parameter file.

Below are some entries we will be modifying.

```
# ---------------------------------------------------
# General grid settings
# ---------------------------------------------------
grid_settings:
  # --------------------------------------------------------------------------
  # Settings for grid generation (num cells, transforms, distortion_correction)
  # --------------------------------------------------------------------------
  grid_generation:
    distortion_correction:
      all:
        active: True # true, 1 also valid.
        resolution: 1000
```

(continues on next page)

(continued from previous page)

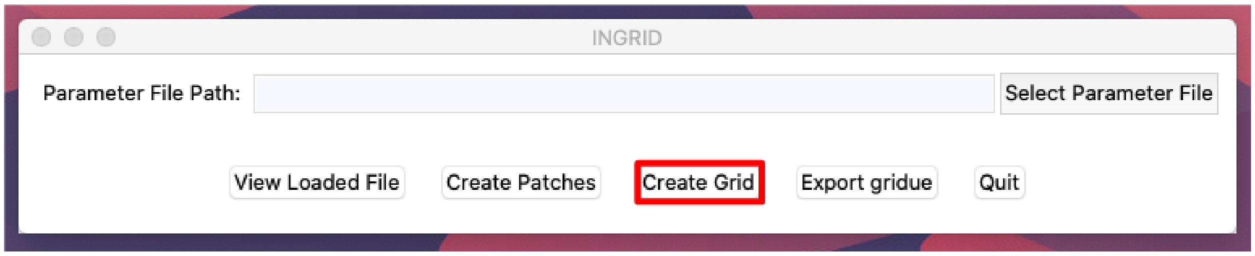
```
        theta_max: 120.0
        theta_min: 80.0
    np_default: 3
    nr_default: 3
    # Other grid settings
```

Within the entry `grid_settings`, we have:

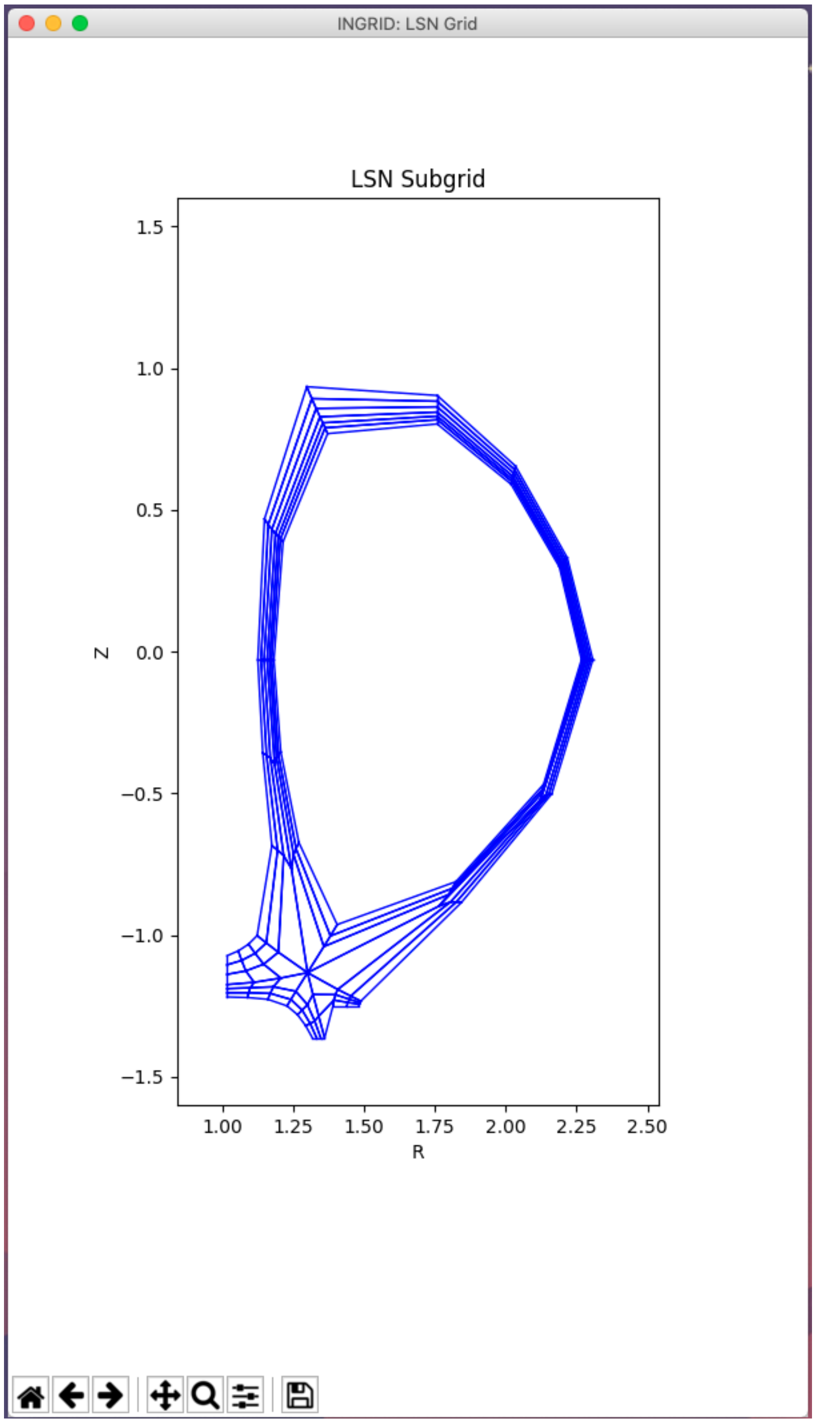
- `grid_generation` - settings for controlling resultant grid
  - `distortion_correction` - settings for controlling shearing in grid
  - `np_default` - default number of poloidal cells per `Patch`
  - `nr_default` - default number of radial cells per `Patch`

Note: We will work with entry `distortion_correction` at a later time (next section). For now, set the entry value to `False` so that we can see it's effects later

To execute refinement of the `Patch` map into a grid, we click the GUI button `Create Grid`.



The terminal will prompt the user with the progress of `Patch` refinement by providing a short summary of the subgrid being generated within each `Patch`. When `Patch` refinement has finalized, we are greeted with a new window showing the resultant grid.

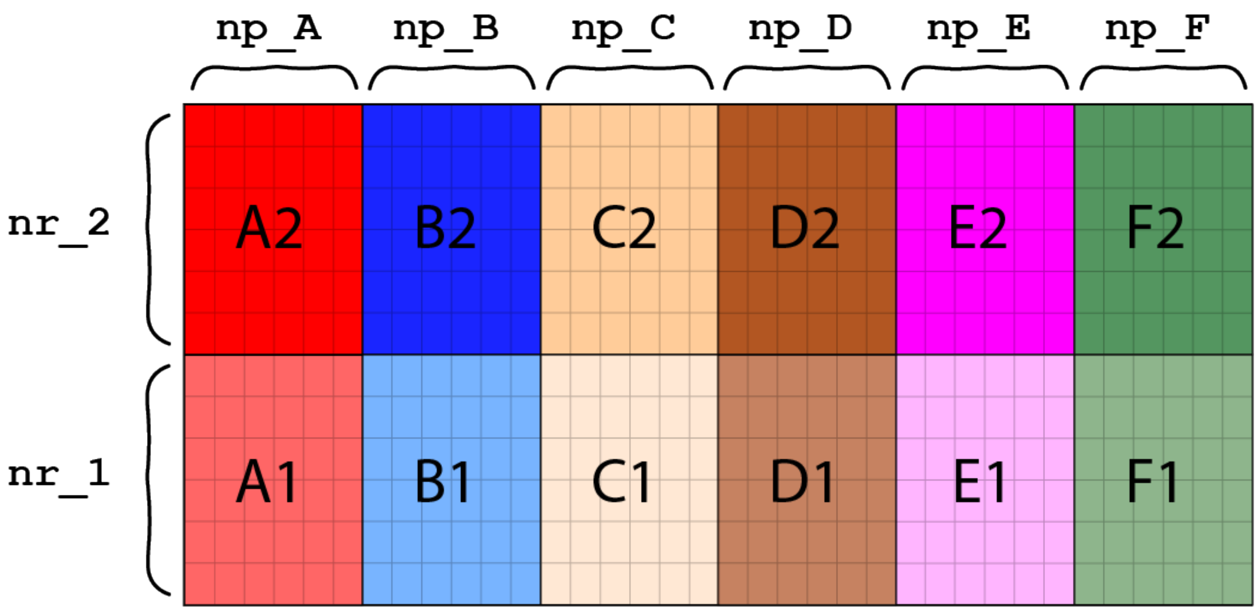


Although this grid can be immediately exported, there are still actions we can take to improve our grid naively generated with only `np_default` and `nr_default`.

2.4.7 Fine-tuning the grid

Generating grids with global values `np_default` and `nr_default` is not enough in many cases. INGRID allows users to specify the number of poloidal and radial cells for particular regions of the `Patch` map. This allows for refining the grid near regions of interest while maintaining global/default grid values per `Patch`.

To utilize this feature, we will fall back on the `Patch` naming convention explained in section patch-map-ref. The figure below shows a collection of keyword entries (`np_A`, `np_B`, … `np_F`, `nr_1`, `nr_2`) that can be added to the INGRID parameter file to control the number cells in a grid.



Note how modifying `np_A` would affect both "A2" and "A1" since they are poloidally dependent in index-space. Similarly, we see how modifying `np_1` would affect "A1" - "F1" since they are radially dependent in index-space.

Let's illustrate this idea by increasing the number of poloidal cells near both target plates. We see by inspecting the `Patch` map that target plates border patches "A*" and "F*". This says we must add entries `np_A` and `np_F` to the INGRID parameter file. That is:

```
# ---------------------------------------------------
# General grid settings
# ---------------------------------------------------
grid_settings:
  # ----------------------------------------------------------------------------
  # Settings for grid generation (num cells, transforms, distortion_correction)
  # ----------------------------------------------------------------------------
  grid_generation:
    distortion_correction:
      all:
        active: false # true, 1 also valid.
        resolution: 1000
        theta_max: 120.0
        theta_min: 80.0

    np_A: 9   # Create 9 poloidal cells in patches A1 and A2
    np_F: 9   # Create 9 poloidal cells in patches F1 and F2

    np_default: 3
```

(continues on next page)

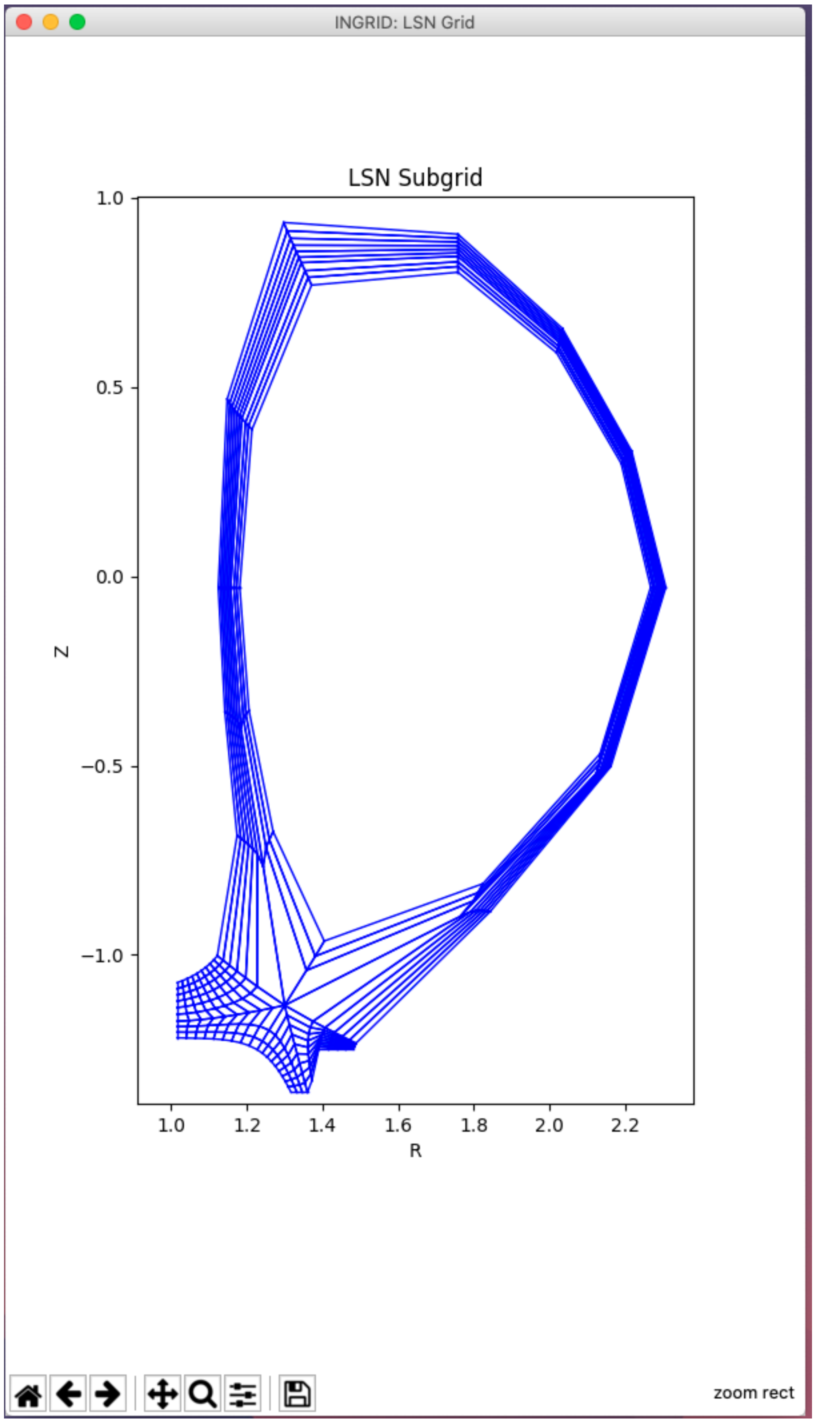
(continued from previous page)

```
    nr_default: 3
    # Other grid settings
```

In addition to refining poloidally, let's increase the radial resolution near the target plates. In this case "A2" and "A1" are *not* dependent on each other (as seen in figure above). On the other hand, since the SOL consists of all patches with numeric-tag "2", modifying "A2" in radial cells will modify all other patches in the SOL radially to keep the index-map consisitent. We will choose to refine "A2". That is:

```
# ---------------------------------------------------
# General grid settings
# ---------------------------------------------------
grid_settings:
  # ----------------------------------------------------------------------------
  # Settings for grid generation (num cells, transforms, distortion_correction)
  # ----------------------------------------------------------------------------
  grid_generation:
    distortion_correction:
      all:
        active: false # true, 1 also valid.
        resolution: 1000
        theta_max: 120.0
        theta_min: 80.0

    np_A: 9  # Create 9 poloidal cells in patches A1 and A2
    np_F: 9  # Create 9 poloidal cells in patches F1 and F2

    nr_2: 6  # Create 6 radial cells in layer 2

    np_default: 3
    nr_default: 3
    # Other grid settings
```

After making the addition, save the file and click "Create Grid". INGRID will detect that an edit was made to the parameter file and apply all changes. When `Patch` refinement has finalized, we are greeted with a new window showing the updated grid.



We can continue to modify the grid in order to allocate more cells near the x-point. A natural choice would be to target np_B and np_E. Doing so with the values np_B = 18 and np_E = 18 (double resolution for the larger patches), we see our parameter file consists of:

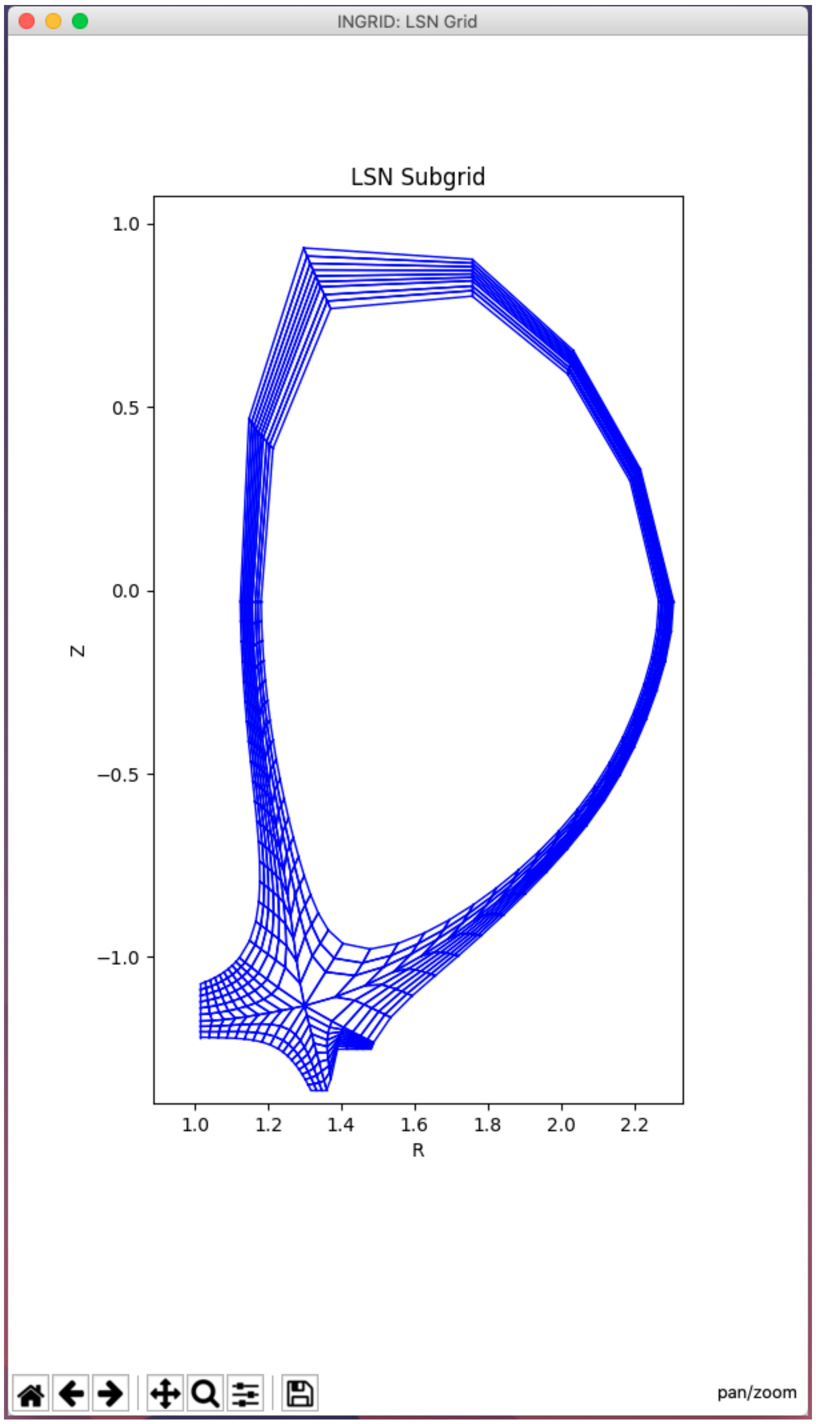
```
# ---------------------------------------------------
# General grid settings
# ---------------------------------------------------
grid_settings:
  # ---------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
# Settings for grid generation (num cells, transforms, distortion_correction)
# ---------------------------------------------------------------------------
grid_generation:
  distortion_correction:
    all:
      active: false # true, 1 also valid.
      resolution: 1000
      theta_max: 120.0
      theta_min: 80.0

  np_A: 9  # Create 9 poloidal cells in patches A1 and A2
  np_F: 9  # Create 9 poloidal cells in patches F1 and F2
  np_B: 18  # Create 18 poloidal cells in patches B1 and B2
  np_E: 18  # Create 18 poloidal cells in patches E1 and E2

  nr_2: 6  # Create 6 radial cells in layer 2

  np_default: 3
  nr_default: 3
  # Other grid settings
```

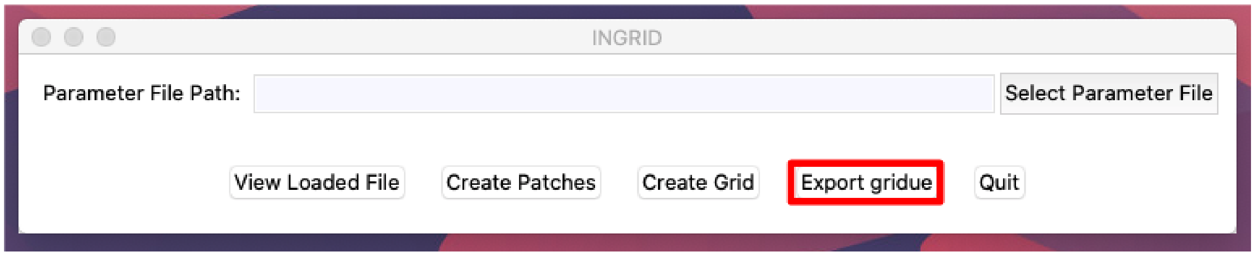
and produces a grid that we can see below (zoomed with Matplotlib toolbar provided in plots).



For the purposes of this introductory tutorial, let us continue to exporting the `gridue` file.

2.4.8 Exporting a `gridue` file

When the user is satisfied with the generated grid, a `gridue` formatted file can be generated by selecting "Export gridue" shown below boxed in red.



From here, the user will be able to select a save location for their INGRID generated `gridue` file.

2.4.9 Summary

In this tutorial, we demonstrated how to generate a `gridue` file for an `SNL` configuration. This introductory tutorial is not an exhaustive demonstration of INGRID's capabilities for grid generation. Other capabilities such as customizing the `Patch` map, applying poloidal/radial grid transformations, and mitigating cell-shearing can be found in the next `SNL` example case.

2.5 Example: single-null configuration (further exploration)

Note: This tutorial assumes the reader has already explored the introductory SNL tutorial.

Some cases **require** enabling of certain attributes in the parameter file in order to successfully produce a grid.

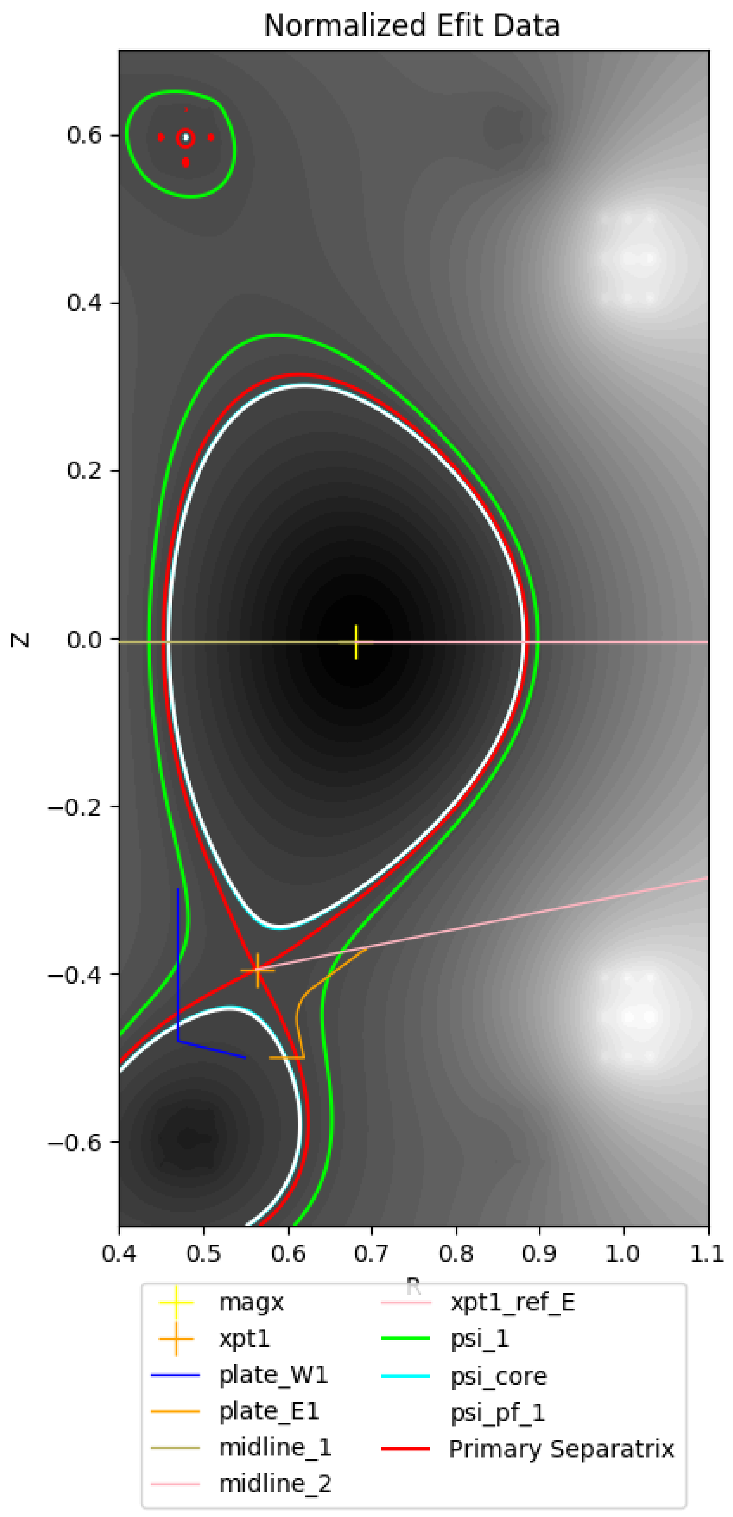
Here we will detail said cases, and also dig deeper into INGRID's capabilities for generating a grid. This tutorial will:

- Detail when adjustment to line-tracing algorithm is required by user
- Illustrate how to make adjustments to a generated `Patch` map
- Illustrate how to apply poloidal/radial transformations for non-uniform grid spacing
- Demonstrate how to reduce cell-shearing (increase orthoganality) of a grid via `distortion_correction`

2.5.1 Loading our example

The parameter file `cmod_param.yml` we will use in this tutorial is located in `example_files/SNL`.

Loading the parameter file in the GUI and viewing the data should show the following.

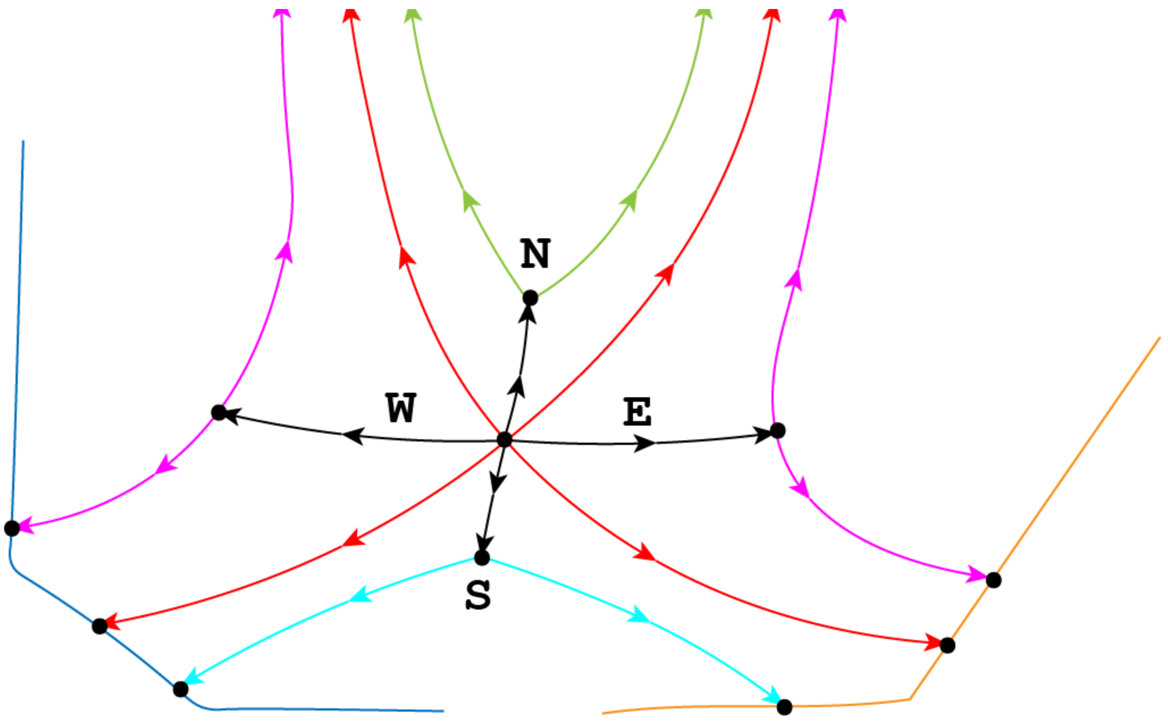

Normalized Efit Data
Z
R
magx
xpt1
plate_W1
plate_E1
midline_1
midline_2
xpt1_ref_E
psi_1
psi_core
psi_pf_1
Primary Separatrix

Immediately we see that there is a line segment originating from the primary x-point and extending to the EFIT domain boundary.

This is an indicator that INGRID will be overriding the default line tracing behavior from the primary x-point. As for why and how we activate this capability will be detailed in the next section.

2.5.2 Standard SNL primary x-point line tracing pattern

INGRID utilizes specific line tracing procedures for each supported topology. Below is a cartoon of line tracing directions from the primary x-point.

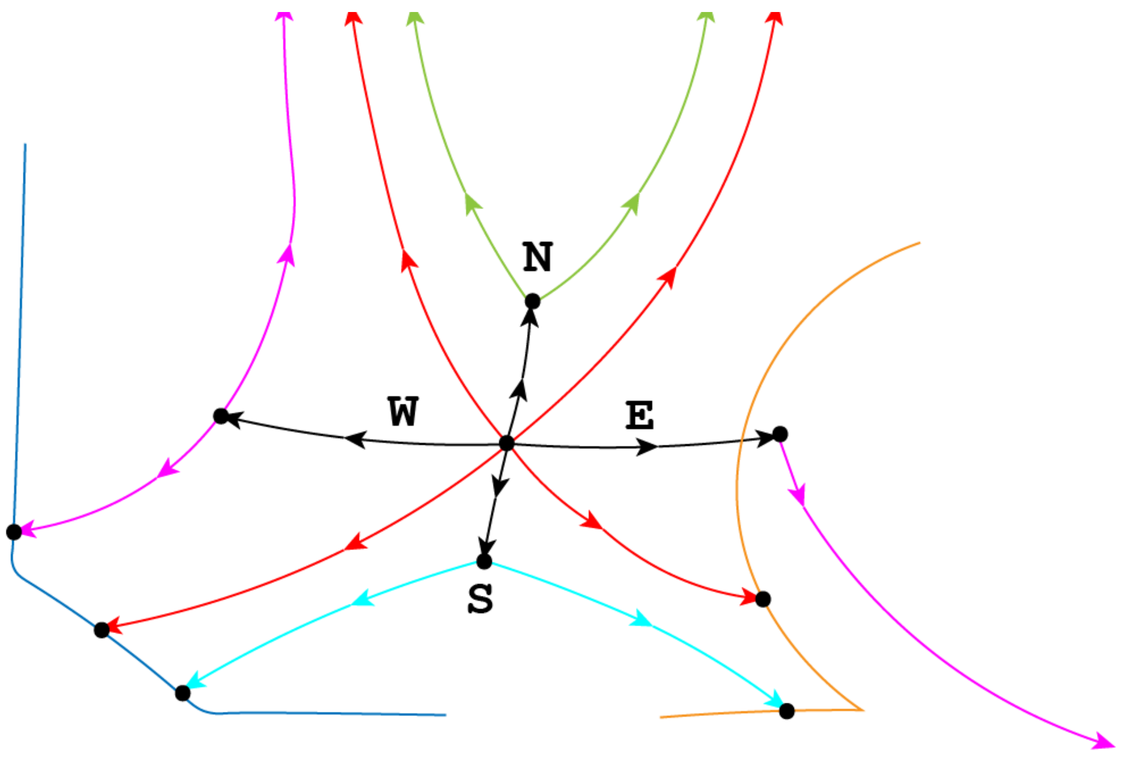


Tracing in direction **N, S, E, W** are orthogonal to flux surfaces.

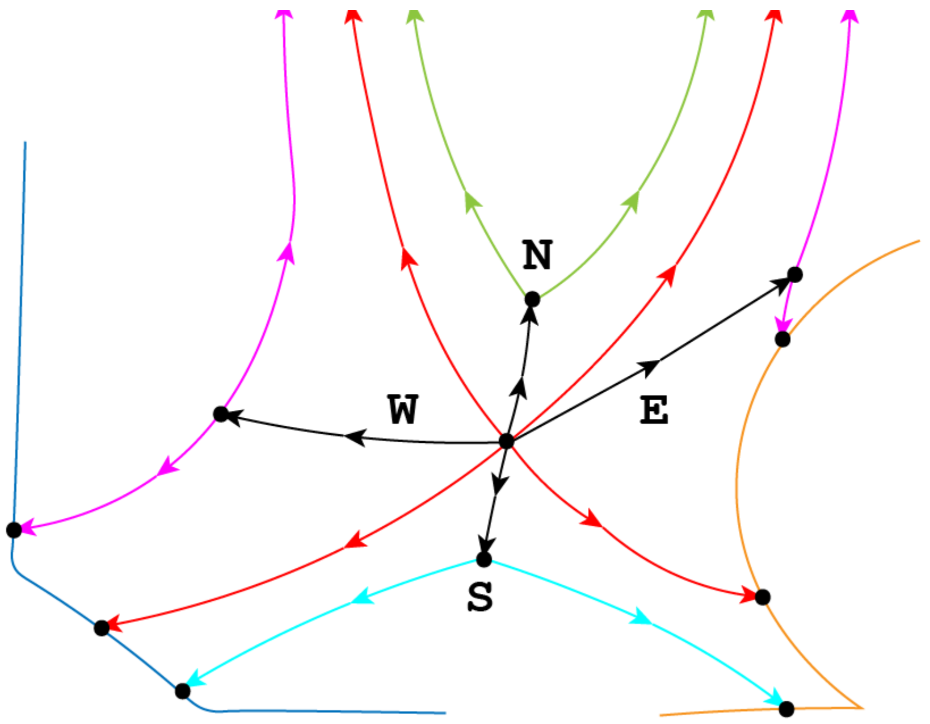
Note that line tracing from the **W** and **E** directions terminate upon intersection with the psi-max surface. Upon intersection with the psi-max surface, line tracing continues along the poloidal line and searches for intersection with a target plate.

In this particular example case we are exploring, intersection with the psi-max surface occurs past the target plate, thus causing line tracing to fail.

We see this in the cartoon below when modifying the target plate geometry.



Although this can indeed be remedied by modifying the target plate geometry or adjusting psi-max surfaces, INGRID allows the user to override the default orthogonal line tracing so that line tracing can continue without error. This remedy is illustrated below.



2.5.3 Overriding SNL primary x-point line tracing pattern

We can override the default orthogonal line tracing for both **E** and **W** directions with the entries `use_xpt1_E` and `use_xpt1_W` that reside within `patch_generation`. This can be seen below.

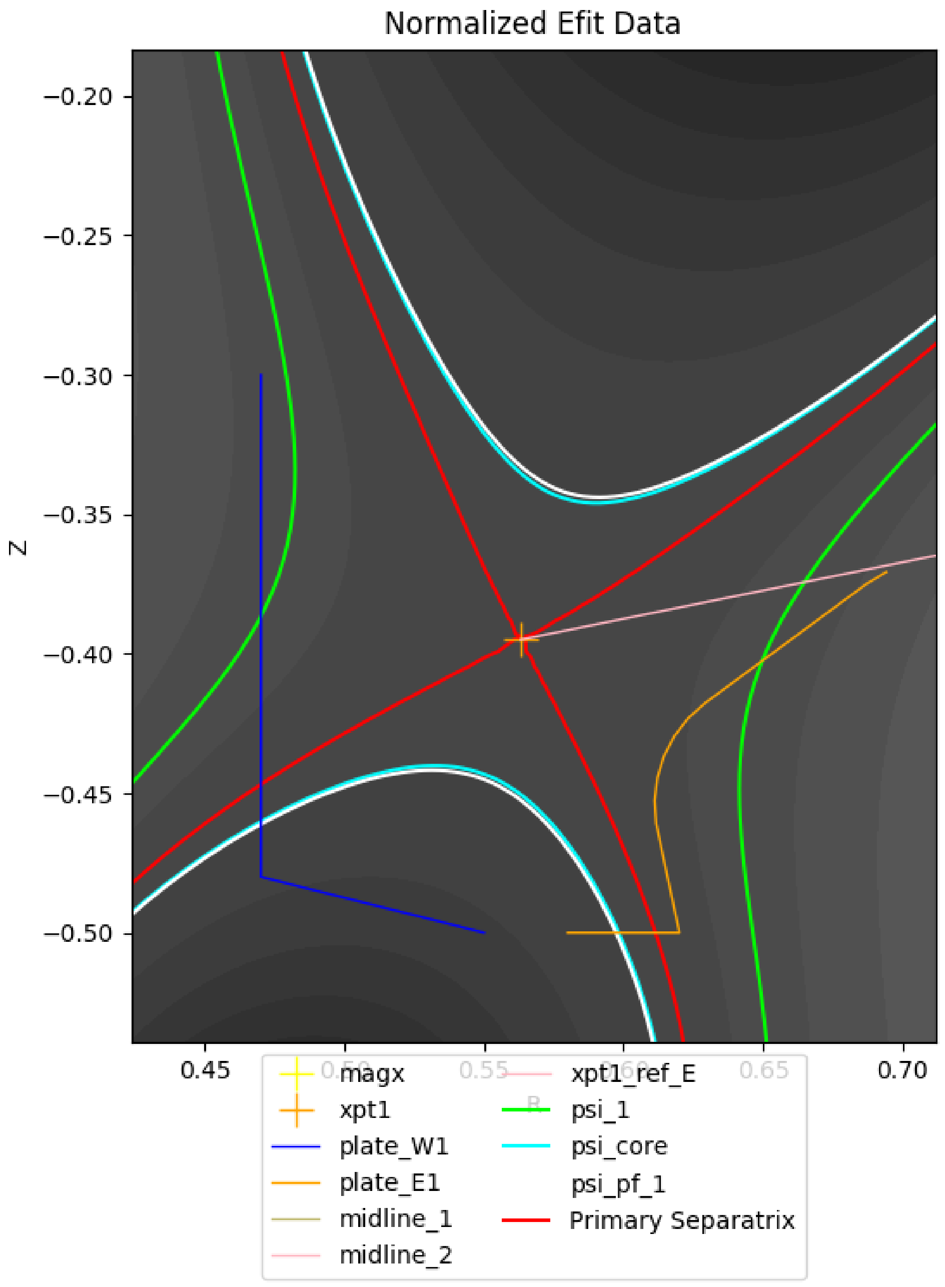
```
grid_settings:
    #...
    # other settings
    #...
    patch_generation:
        use_xpt1_E: true
        use_xpt1_W: false
```

Upon activating either entry and reloading the view into the parameter file, we will see line segment that extends from the primary x-point. This segment is a marker indicating the new line tracing direction.

By default, no rotation is applied to the line tracing direction. We can adjust the direction with the entries `xpt1_E_tilt` and `xpt1_W_tilt`. We see this below.

```
grid_settings:
    #...
    # other settings
    #...
    patch_generation:
        use_xpt1_E: true
        use_xpt1_W: false
        xpt1_E_tilt: 0.2  # radian value for rotation
        xpt1_W_tilt: -0.8  # radian value for rotation
```

The user now has the tools to remedy the above situation. We can see in this case that `xpt1_E_tilt: 0.2` provides enough clearance such that intersection with the target plate will occur.


Normalized Efit Data
−0.20
−0.25
−0.30
−0.35
−0.40
−0.45
−0.50
Z
0.45
0.50
0.55
0.60
0.65
0.70
R
magx
xpt1
plate_W1
plate_E1
midline_1
midline_2
xpt1_ref_E
psi_1
psi_core
psi_pf_1
Primary Separatrix

Tip: When it is not immediately obvious from the loaded EFIT data that orthogonal line tracing will intersect psi-max past the target plate, the user can change the visualization of the INGRID data from

filled contours to unfilled. We do this by changing `view_mode:  filled` to `view_mode:  lines`. We can control the number of contour lines plotted by modifying the `nlevs` entry as well. This can help with visually imagining where orthogonal line tracing will terminate.

2.5.4 Other settings for Patch map modification

Overriding orthogonal line tracing from the primary x-point is just one modification that can be made to influence a final Patch map.

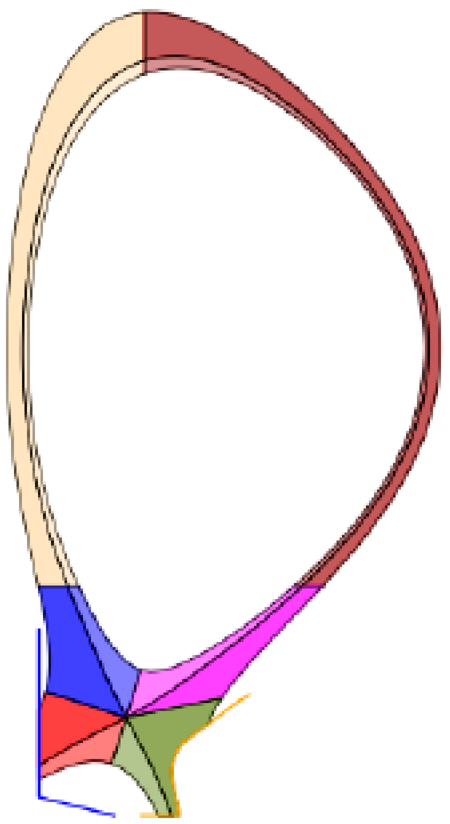
SNL line tracing for certain patches in the core will define boundaries based off intersection with the **horizontal** and **vertical** lines that intersect the magnetic axis (midplane).

One such modification is applying an RZ translation to the magnetic-axis coordinate used to generate said Patch boundaries.

This can be controlled in the parameter file by editing entries `rmagx_shift` and `zmagx_shift` under `patch_generation` in `grid_settings`.

```
grid_settings:
    patch_generation:
        rmagx_shift: 0.0  # Translate R coordinate
        zmagx_shift: 0.0  # Translate Z coordinate
```

Saving the parameter file and reloading the view into the data will reflect the changes. The Patch map generated with the translations above can be seen below.

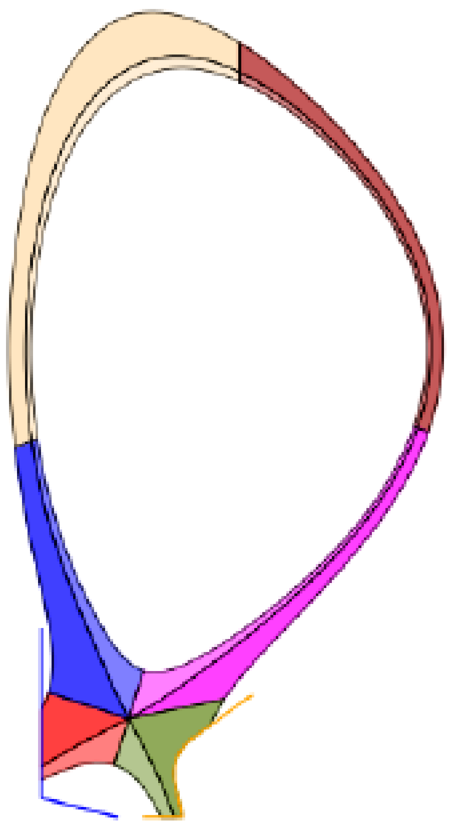


In a similar manner to adjusting the angle of line tracing in the E and W directions from the primary x-point, we can adjust the line segments extending from the magnetic-axis. These line segments define the east faces of patches B1 and B2, as well as the west faces of patches E1 and E2.

The tilt of the inner-midplane and outer-midplane can be controlled with entries magx_tilt_1 and magx_tilt_2 respectively. These entries are contained within patch_generation in grid_settings.

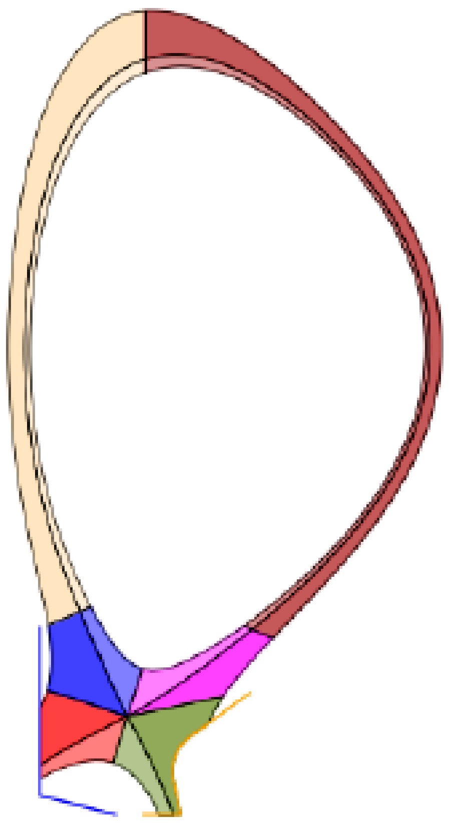
```
grid_settings:
    patch_generation:
        magx_tilt_1: 0.0  # inner-midplane rotation (in radians)
        magx_tilt_2: 0.0  # outer-midplane rotation (in radians)
```

Saving the parameter file and reloading the view into the data will reflect the changes. The Patch map generated with the only the tilt values entered above can be seen below.



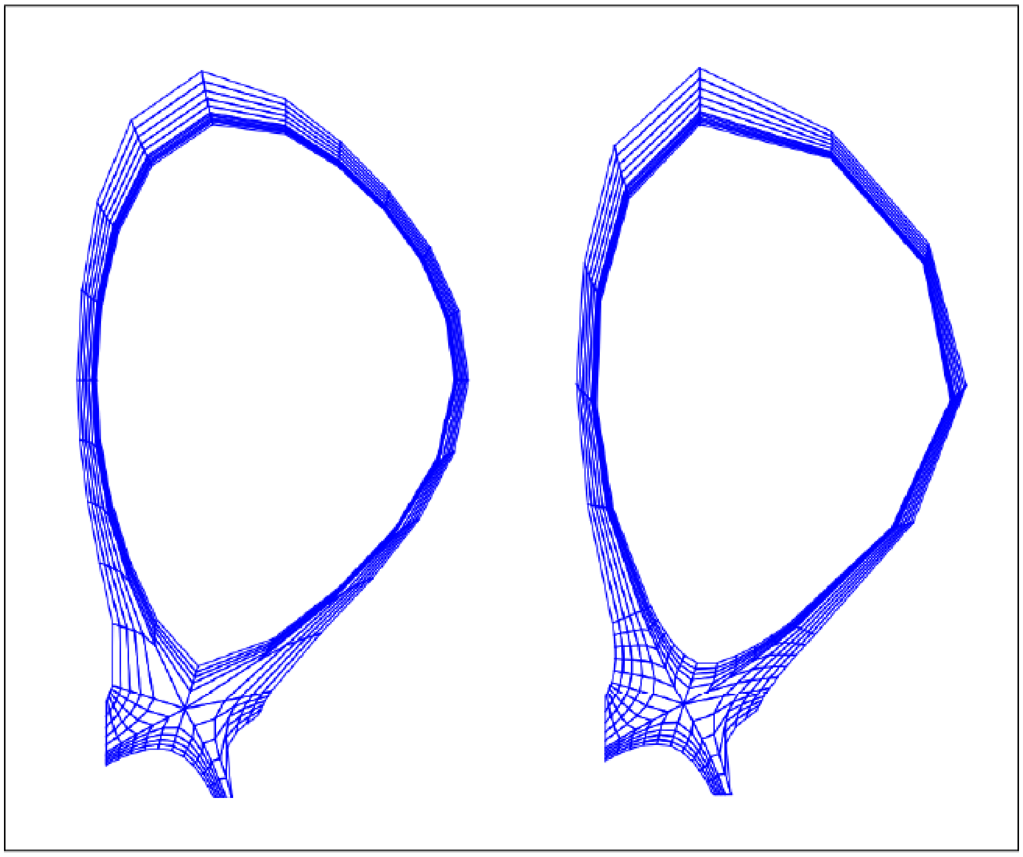
Note: Midplane tilt entries are in radians and follow the standard counter-clockwise rotation direction.

Combining both together yields the following Patch map.



Tip: Applying these Patch modifications appropriately can allow one to increase cell density near primary x-point without modifying np/nr values

On the left is the grid with no Patch map modifications for reference.

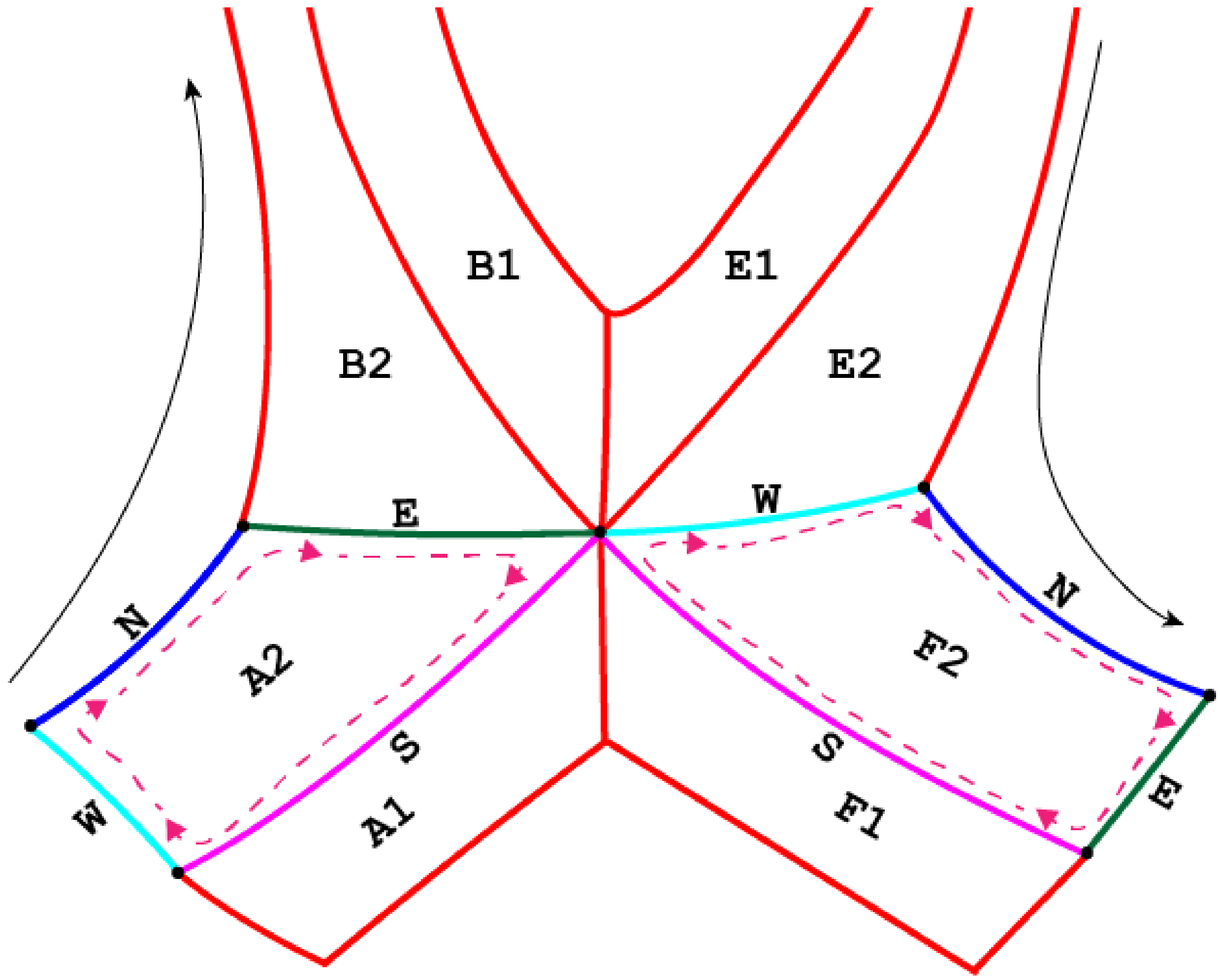


2.5.5 Background knowledge for poloidal and radial grid transformations

INGRID allows the user to provided poloidal and radial grid distribution functions for generating non-uniform grids.

Before detailing how to invoke these features, some background on the Patch object itself.

Each Patch boundary is defined by 4 lines that we refer to as **N**, **E**, **S**, **W**. This allows for us to maintain a clockwise orientation on the boundary of a Patch. Below is a cartoon illustrating the idea.



The **N** boundary (seen in dark blue) for Patch A2 begins at the max-psi strike-point on the target plate west of the primary x-point (inner target plate for SNL case), and terminates at the B2 interface. The **S** boundary (seen in magenta) for Patch A2 is oriented in the opposite direction and terminates upon intersection with the target plate.

The **E** and **W** boundaries (seen in dark green and cyan, respectively) are defined in the radial direction relative to the **N** and **S** boundaries.

Note that this convention holds throughout the entire Patch map. We can see this by noticing that upon reaching Patch `F2`, the **E** boundary is now defined by a portion of the (LSN outer) target plate.

Note: INGRID chooses to parameterize the **N face in length with parameter $s \in [0, 1]$ for poloidal distribution functions. Similarly, INGRID chooses to parameterize the **W** face in increasing psi with parameter $s \in [0, 1]$ for radial distribution functions. **

Now we discuss how the user specifies poloidal and radial grid transformations within the parameter file.

INGRID parses a string from the user in the form ``x, f(x)`` where :math:`x` indicates the dependent variable and :math:`f(x)` is mathematical expression representing the distribution. Within the parameter file, we have seen the string `x, x` utilized for entries `radial_f_default` and `poloidal_f_default`. INGRID interprets this as applying a uniform distribution of vertices for defining the grid (consistent with what we have seen).

Warning: Due to the parameterization $s \in [0, 1]$, defining $f(x)$ such that $f : [0, 1] \to [0, 1]$ is important. Apply appropriate normalization operations when utilizing non-trivial functions (see example below).

INGRID utilizes SymPy for generating a function from the user provided string. Standard Python arithmetic operations are supported (+, −, *, /, **, ...), as well as common mathematical functions such as `exp` and `log`.

2.5.6 Applying poloidal and radial grid transformations

In general, we adopt a notation similar to specifying np/nr cells. Below is a snippet of a YAML file with default poloidal and radial transformation values.

```yaml
grid_settings:
    grid_generation:

        # ...
        # Other grid_generation settings
        # ...

        poloidal_f_default: x, x  # Global uniform poloidal
        radial_f_default: x, x    # Global uniform radial
```

Much like `np_default` and `nr_default`, entries `poloidal_f_default` and `radial_f_default` apply to poloidal "columns" and radial "rows" in index space, respectively. The `default` appended to `poloidal_f_` and `radial_f_` tells INGRID to apply the corresponding transformation globally.

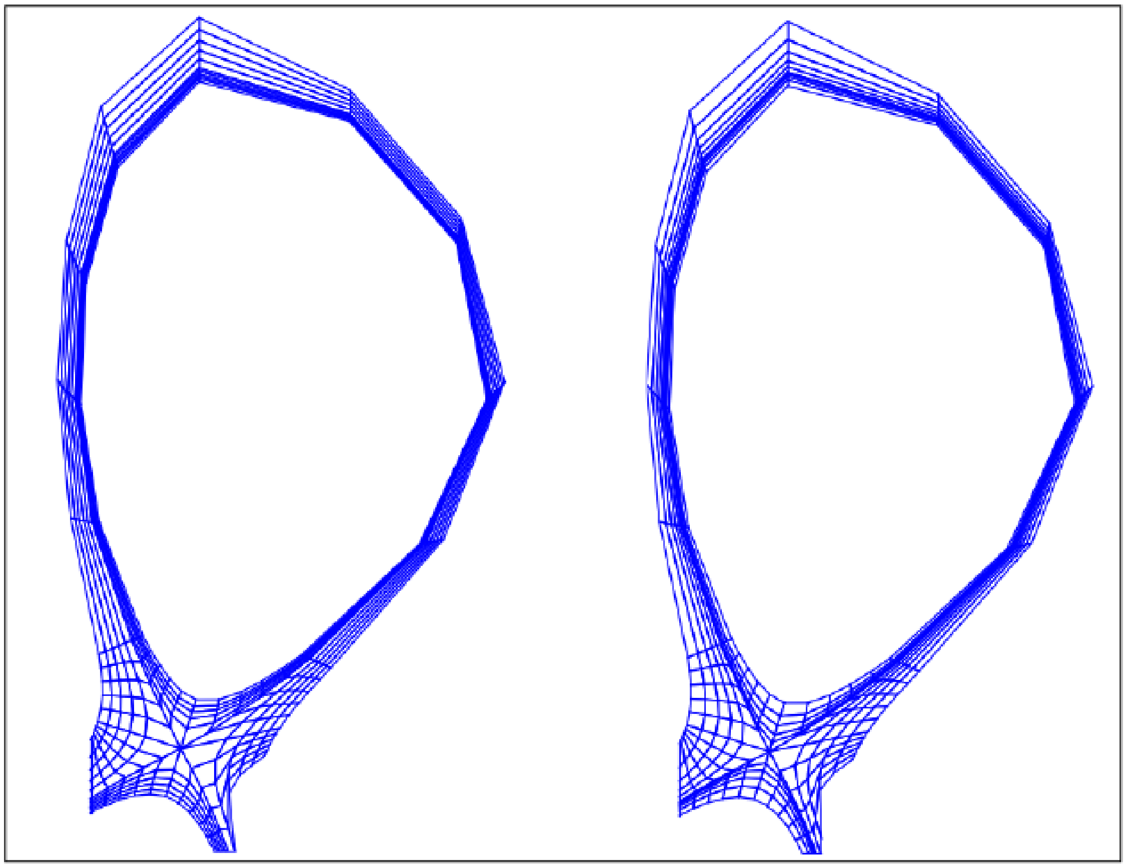
Poloidal transformations can be specified with the same convention as specifying poloidal cells (`poloidal_f_A`, `poloidal_f_B`, ..., `poloidal_f_F`).

Radial transformations follow the same convention (`radial_f_1`, `radial_f_2`), but also have an additional `radial_f_3` specifically for the inner-core region.

The following applies an exponential-like distribution (between 0 and 1) for the SOL, PF, and CORE. These transformations will generate grid cells that hug the primary separatrix slightly more than usual.

```yaml
grid_settings:
    grid_generation:

        # ...
        # Other grid_generation settings
        # ...

        poloidal_f_default: x, x  # Global uniform poloidal
        radial_f_default: x, x    # Global uniform radial
        radial_f_1: x, 1-(1-exp(-(1-x)/0.4))/(1-exp(-1/0.4))
        radial_f_2: x, (1-exp(-(x)/0.8))/(1-exp(-1/0.8))
        radial_f_2: x, (1-exp(-(x)/0.8))/(1-exp(-1/0.8))
```

The resulting grid with transformations can be seen on the left, and the original grid with no transformations can be seen on the right.



2.5.7 Reducing cell shearing via `distortion_correction`

INGRID does not enforce an orthogonality condition when generating a grid. INGRID allows the user to impose angle constraints on cells within a generated grid in order to increase orthogonality. We do this via the `distortion_correction` feature.

Below is an example of cell shearing and the motivation for INGRID's `distortion_correction`.

This `distortion_correction` tool allows the user to specify angle constraints `theta_min` and `theta_max` in order to mitigate cell shearing. INGRID will shift the cell vertex by increments of 1 / `resolution` until the resultant angle is within the user constraints.

If the constraint cannot be satisfied (vertex leaves the Patch), INGRID will backtrack until the vertex is within the Patch bounds.

Below is a snippet of the parameter file format showing `distortion_correction` applied globally.

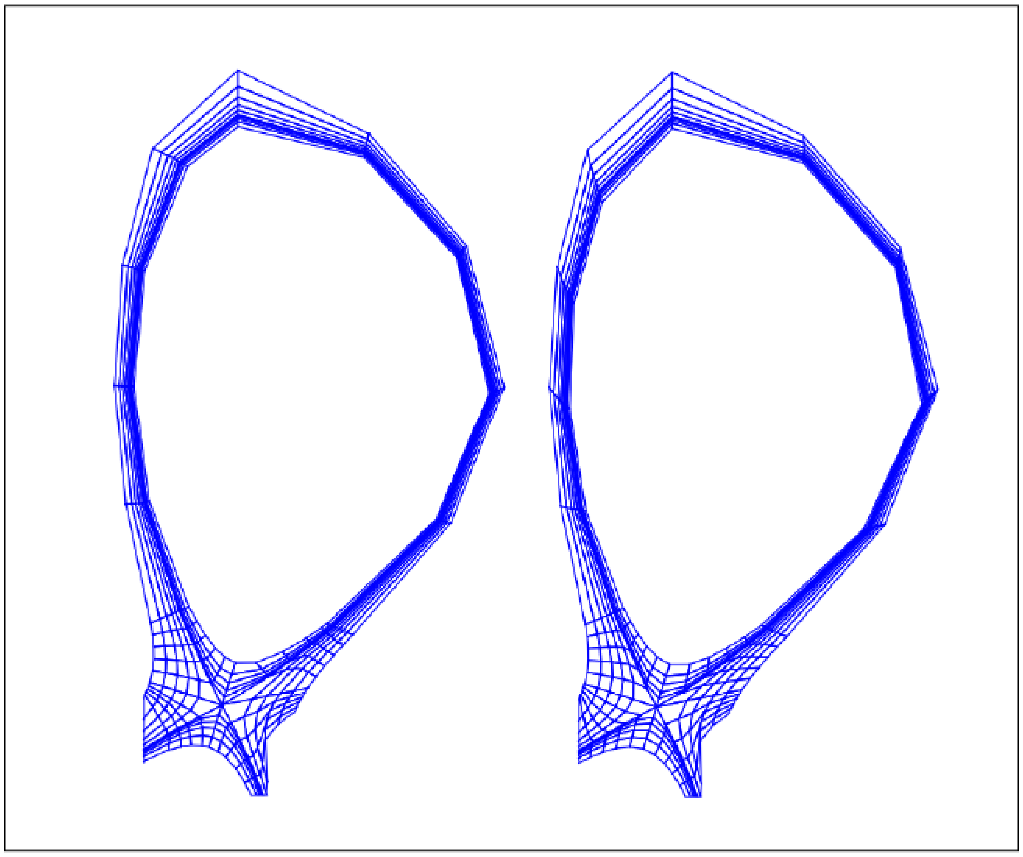
```
grid_settings:
    grid_generation:
        distortion_correction:
```

(continues on next page)

(continued from previous page)

```
            # Global settings
            all:
                active: True  # toggle distortion_correction
                resolution: 1000  # 1 / resolution step-size for shifting vertex
                theta_max: 120.0  # angle constraint
                theta_min: 80.0   # angle constraint

            # Patch specific settings can be provided in addition to global settings
            # (similar to how we specify np/nr for Patches on top of default np/nr)

            # Example: Specify distortion_correction for Patch A1

            # A1:  # <-- (Patch name here can be changed)
            #    active: false  # toggle distortion_correction
            #    resolution: 1000  # 1 / resolution step-size for shifting vertex
            #    theta_max: 120.0  # angle constraint
            #    theta_min: 80.0   # angle constraint

        np_default: 5
        nr_default: 5
        poloidal_f_default: x, x
        radial_f_default: x, x
```

Below is a side-by-side comparison of `distortion_correction` toggled on and off respectively.



We can see that the radial lines are indeed more orthogonal to the poloidal contours. Although only a mild effect in this case, we have seen that `distortion_correction` can significantly reduce shearing and generate tidier grids in others (example SF cases seen below).

This feature (in addition to those detailed above) is just another tool at a user's disposal that need not be utilized in every case.

2.5.8 Adjusting guard cell size

Guard cell size for a generated grid can be specified within the parameter file by editing entry `guard_cell_eps` within `grid_settings`. That is:

```
# ---------------------------------------------------
# General grid settings
# ---------------------------------------------------
grid_settings:
    # ...
    # ... Other grid_settings entries
```

(continues on next page)

(continued from previous page)

```
    # ...
    guard_cell_eps: 0.00001  # Size of guard cells.
```

Note: Specification of guard cell size must be done prior to initiating grid generation (clicking `Create Grid`).

2.5.9 Summary

In this tutorial, we encountered a situation where parameter file modification is required for INGRID to successfully generate a Patch map. This was resolved by modifying the line tracing procedure in order to accomodate the provided geometry.

We also saw how the modification of the line tracing procedure falls into the over-arching category of INGRID tools that allow the user to customize a Patch map.

Finally, we dove deeper into grid customization capabilities such as applying `distortion_correction`, poloidal and radial transformations, and specifying guard cell size.

2.6 Example: Two x-points in domain (SF75 example)

Note: This tutorial assumes the reader has already read both the introductory SNL tutorial and the further exploration SNL tutorial.

Here we will demonstrate how to generate a grid when there are two x-points in the domain. In particular, we will generate a grid for a snowflake-75 (SF75) configuration.

Because INGRID internally handles the identification and classification of magnetic topology, the general steps detailed here for generating a grid apply to **all** the other configurations with two x-points in the domain. The user will only need to refer to provided diagrams of a configuration's Patch map, x-point NSEW directions, and psi labels for a specific configuration.

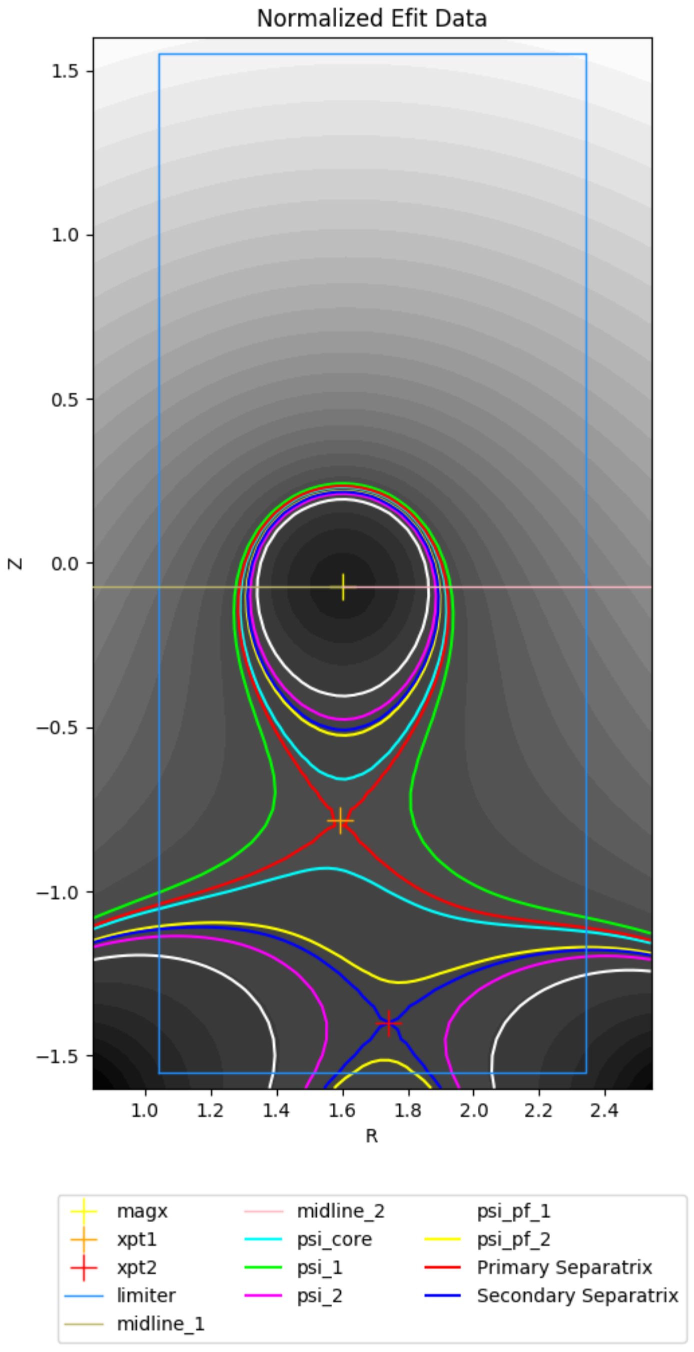
2.6.1 Loading our example

The parameter file `SF75.yml` we will use in this tutorial is located in `example_files/SF75`.

A key difference inside the parameter file is that now the entry `num_xpt` has an associated value of `2`. This **activates** INGRID's topology classification when two x-points are present in the domain. With this, INGRID now expects more parameter file entries from the user. These include:

- Approximate R coordinate of secondary x-point `rxpt2`
- Approximate Z coordinate of secondary x-point `zxpt2`
- Additional psi entries `psi_2` and `psi_pf_2`
- Limiter data (or usage of EFIT domain bounds to act as a psuedo-limiter)
- Two additional target plates (when `strike_pt_loc` has an associated value of `target plates`)

Loading the parameter file in the GUI and viewing the data should show the following.

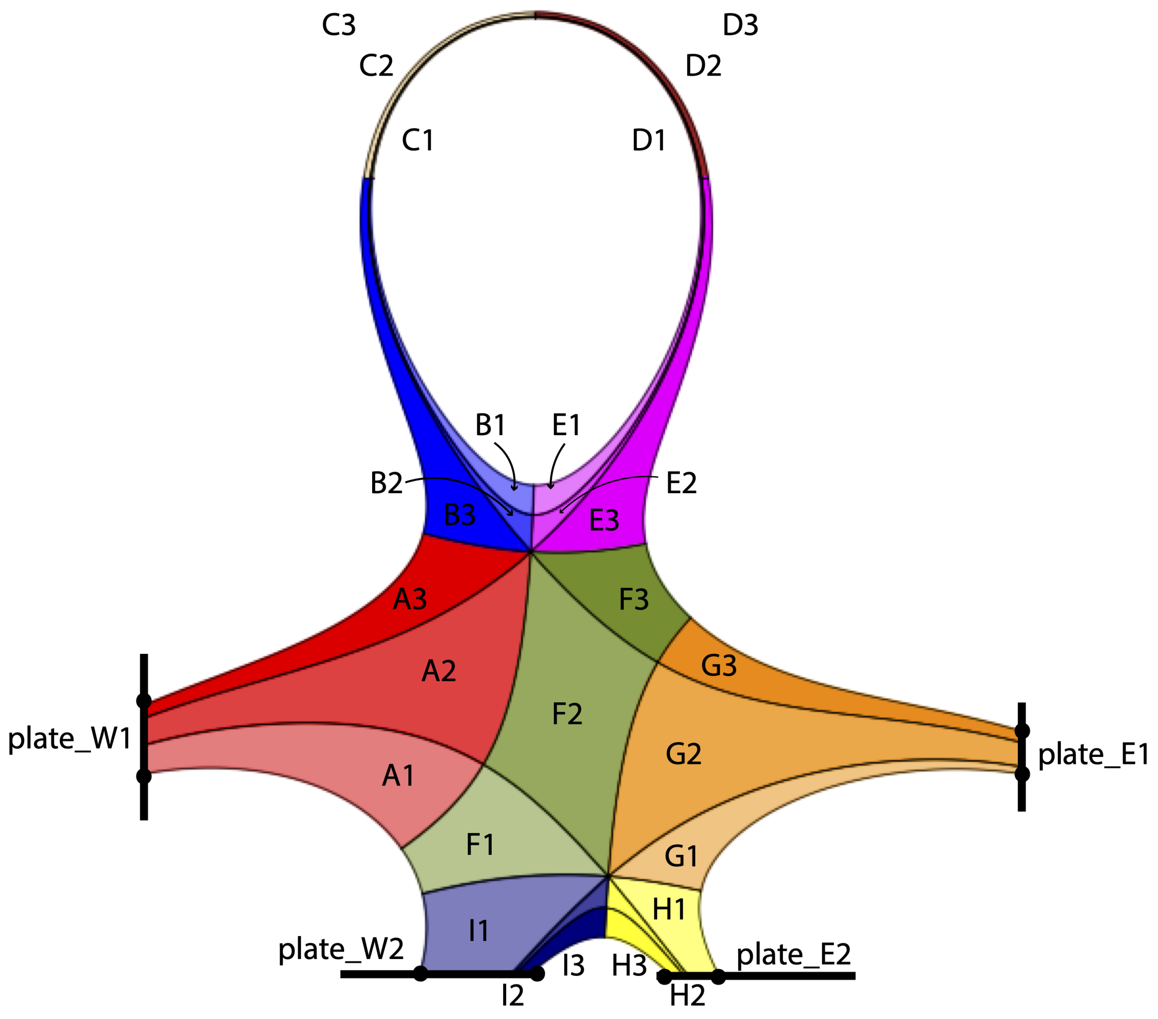


We can immediately see the presence of a secondary x-point, it's corresponding separatrix, a rectangular (psuedo) limiter and additionaly psi lines that we will use to generate a Patch map.

Before generating a Patch map, we discuss the SF75 layout.

2.6.2 The SF75 line-tracing pattern, x-point directions, and psi labels

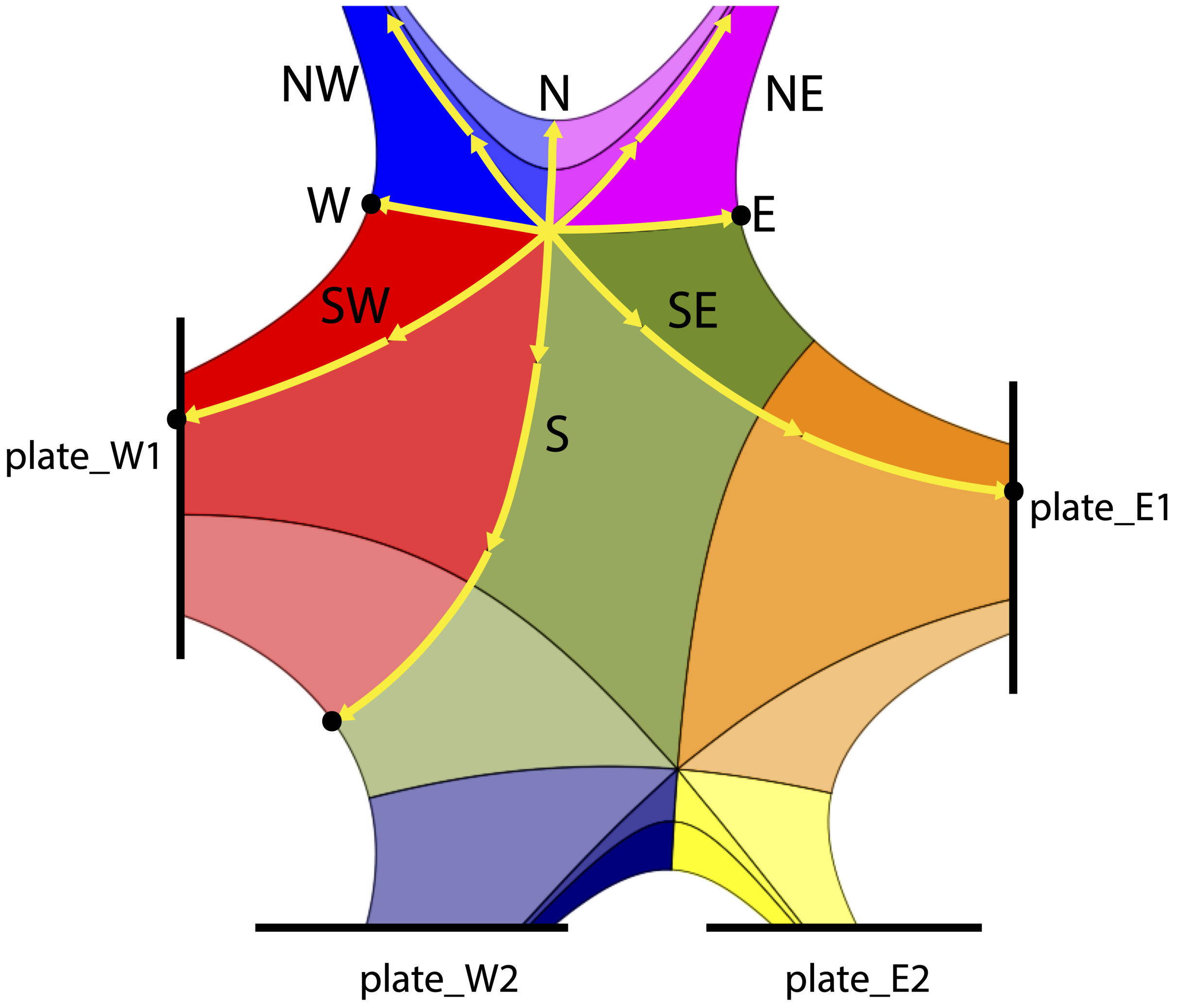
The figure below shows a Patch map for general SF75 configuration.

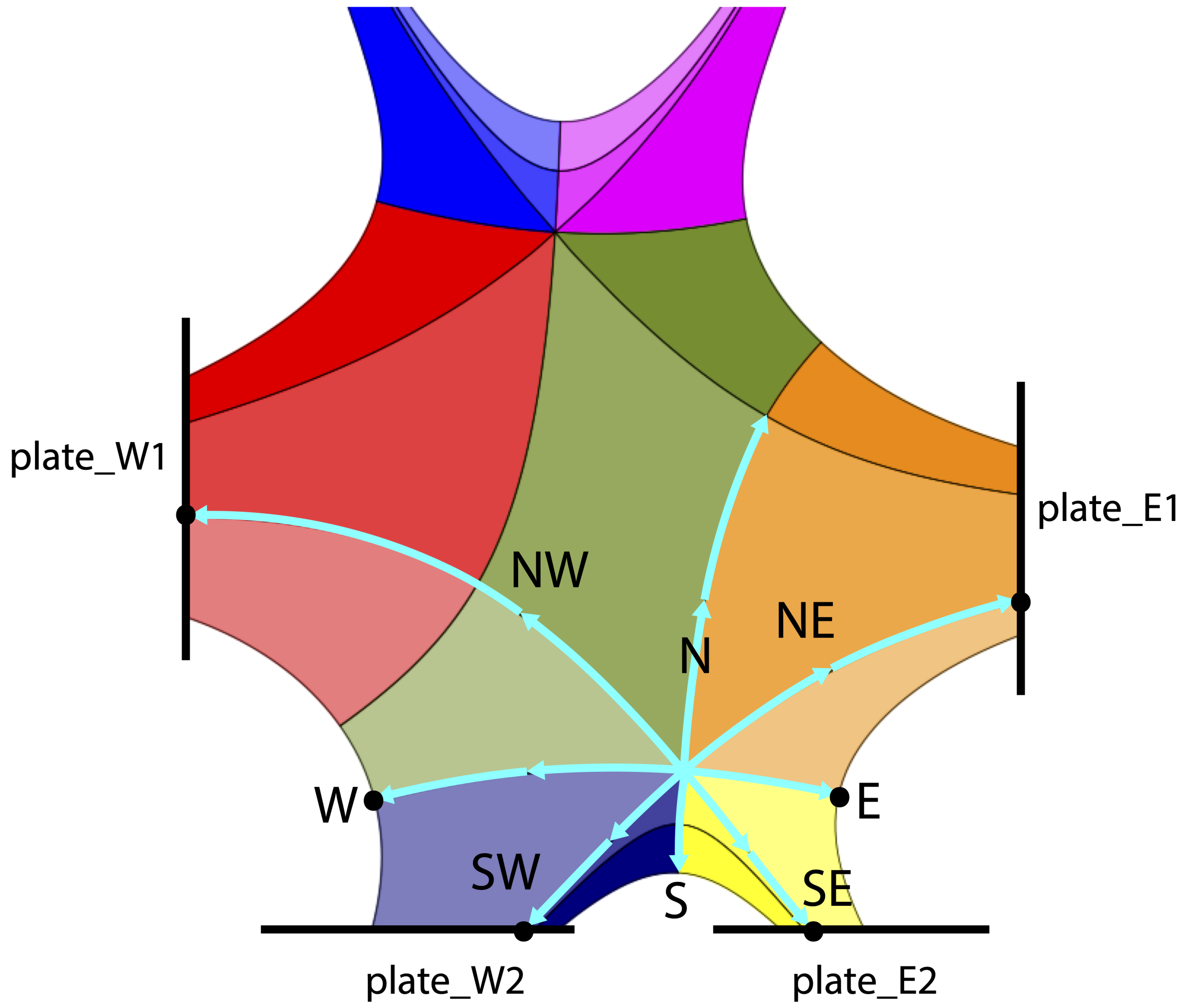


The user should reference the above figure when prepping the parameter file for generating a Patch map by verifying target plates are in valid locations relative to x-points and psi-boundaries.

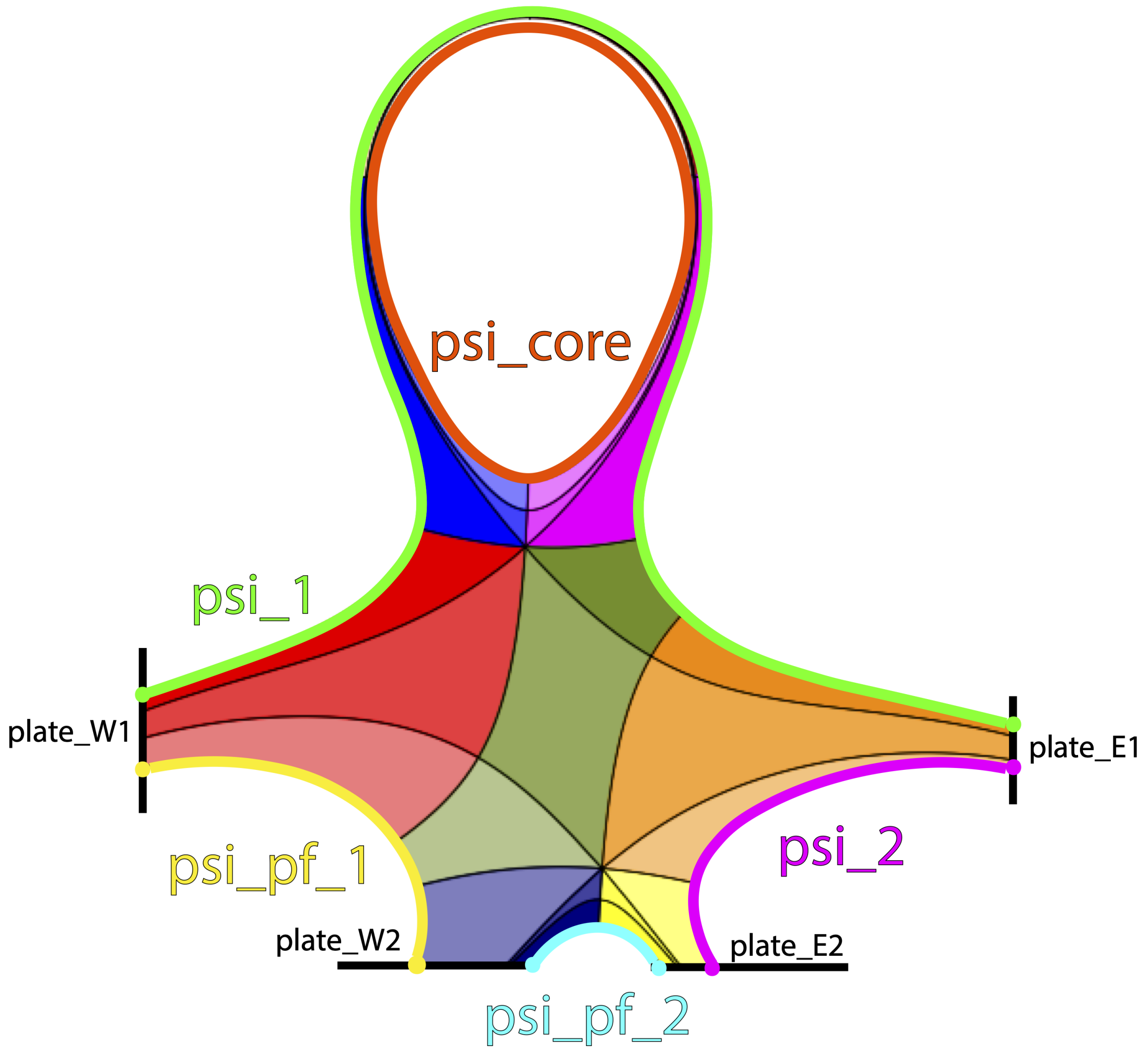
First, note the target plate naming convention `plate_W1`, `plate_E1`, `plate_W2`, and `plate_E2`. Again, by removing notions of "inner" and "outer" plates in the `SNL` case, we see that we now have a general naming convention that can be applied to all configurations with two x-points in the domain.

This naming convention is defined by the NSEW directions for both the primary x-point (`xpt1`) and secondary x-point (`xpt2`). Below we can see two figures that illustrate the NSEW directions for `xpt1` and `xpt2`, respectively.


NW
N
NE
W
E
SW
SE
S
plate_W1
plate_E1
plate_W2
plate_E2



Finally, we illustrate the psi-labels associated with the psi-boundaries of the SF75 configuration.



The above shows we now have parameter file entries `psi_2` and `psi_pf_2` to assign values to.

We will see in other configurations (e.g. `UDN`, `SF45`, etc) that the location of the psi labels **will** vary. Because the location of psi boundaries vary with each configuration, the user must carefully note which psi boundaries they are dealing with.

In general, `psi_1` will correspond to psi-max, `psi_core` remains set to the core psi value, `psi_pf_1` will correspond to the psi-surface we intersect by tracing `S` of `xpt1`, and `psi_pf_2` will correspond to the psi-surface we intersect by tracing `S` of `xpt2`. Typically, the `psi_2` psi-boundary and strike-point locations for the psi-entries listed above often vary.

2.6.3 Activating the limiter for Patch generation

Understanding INGRID limiter usage is an essential part of generating a Patch map for cases with two x-points in the domain.

INGRID does not require the use of target plates for generating a Patch map if the user opts for using a limiter.

This is controlled within the parameter file with `strike_pt_loc` within the `patch_generation` block. We see this below.

```
grid_settings:

  # ...
  # other settings
  # ...

  patch_generation:
    # ...
    # other settings
    # ...

    # strike_pt_loc takes the values of 'limiter' or 'target_plates'
    strike_pt_loc: limiter  # generates a Patch map with a limiter rather than target␣
↪plates
```

When `strike_pt_loc` is set to a value of `limiter`, INGRID will utilize all geometry data provided in the parameter file block labeled `limiter`.

We can see the `limiter` settings in `SF75.yml` below.

```
grid_settings:

  # ...
  # other settings
  # ...

  patch_generation:
    # ...
    # other settings
    # ...

    # strike_pt_loc takes the values of 'limiter' or 'target_plates'
    strike_pt_loc: limiter  # generates a Patch map with a limiter rather than target␣
↪plates

# Specifications for using a limiter
limiter:

  file: ''  # File name of .txt file with coordinates specifying limiter geometry.

  use_efit_bounds: true  # Use the EFIT domain boundary as a limiter

  # Coordinates: [(rmin, zmin), (rmax, zmin), (rmax, zmax), (rmin, zmax), (rmin,␣
↪zmin)]

  zshift: 0.0  # Shift the limiter geometry in the z-direction
  rshift: 0.0  # Shift the limiter geometry in the z-direction

  # Adjust shape of EFIT domain boundary psuedo-limiter in r coordinate
```

(continues on next page)

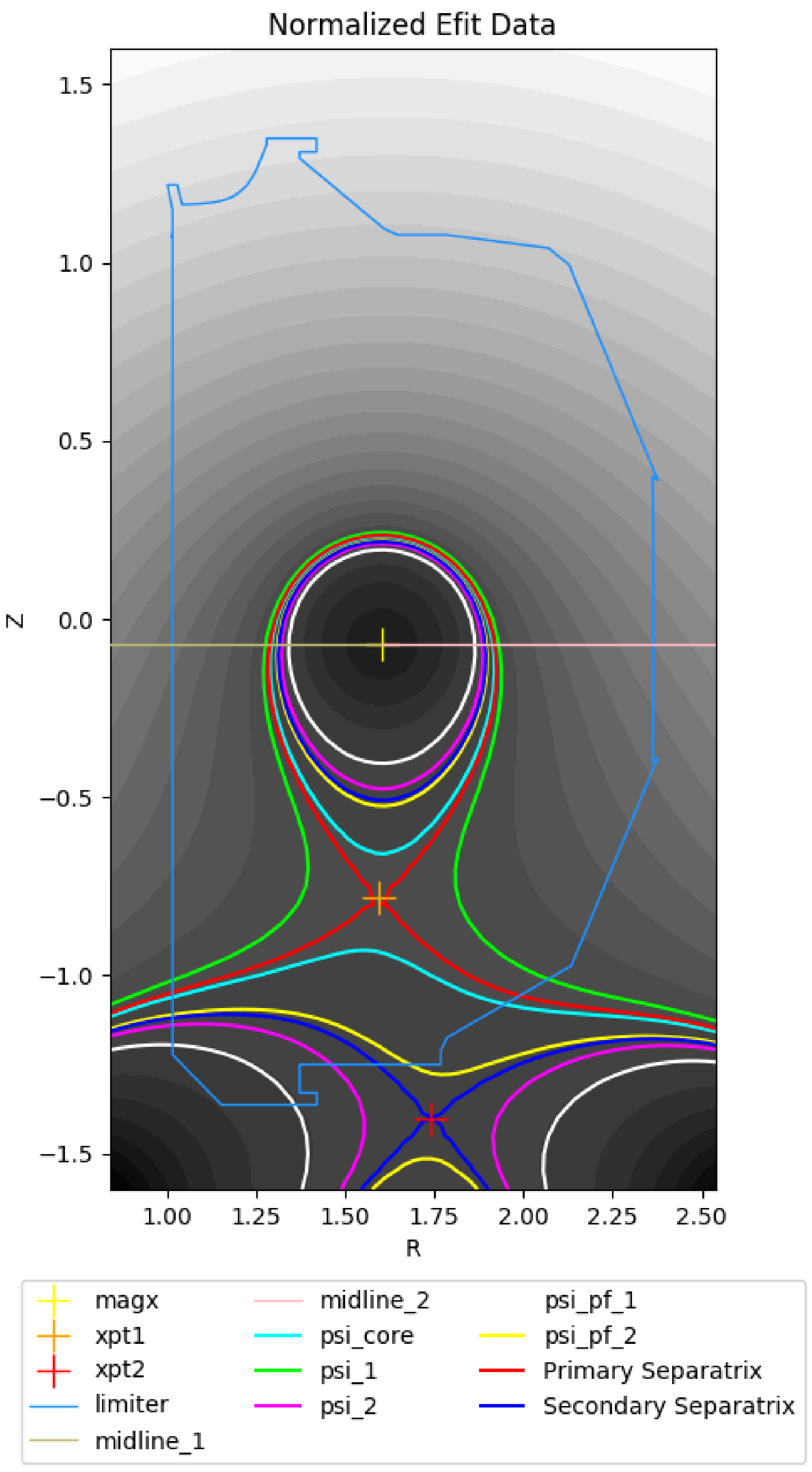
(continued from previous page)

```
  efit_buffer_r: 0.2  # Default value: 1.0e-2

  # Coordinates: [(rmin + efit_buffer_r, zmin), (rmax - efit_buffer_r, zmin),
  #               (rmax - efit_buffer_r, zmax), (rmin + efit_buffer_r, zmax), (rmin +␣
↪efit_buffer_r, zmin)]

  # Adjust shape of EFIT domain boundary psuedo-limiter in z coordinate
  efit_buffer_z: 0.05  # Default value: 1.0e-2

  # Coordinates: [(rmin, zmin + efit_buffer_z), (rmax, zmin + efit_buffer_z),
  #               (rmax, zmax - efit_buffer_z), (rmin, zmax - efit_buffer_z), (rmin,␣
↪zmin + efit_buffer_z)]
```

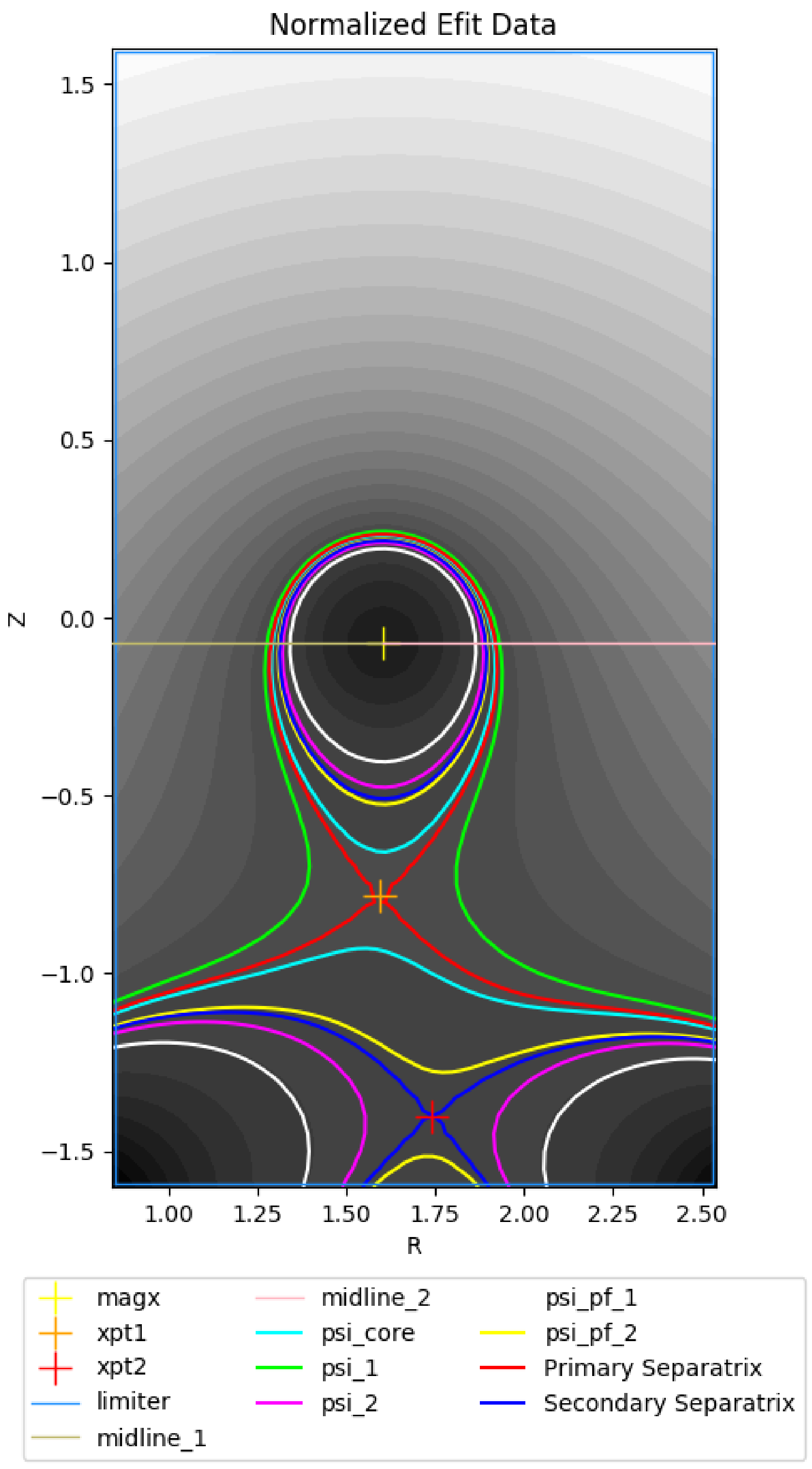
Note: INGRID will utilize the default limiter data provided within the `eqdsk` file if no file is provided and `use_efit_bounds` is set to `False`. If no limiter data is available in the `eqdsk` file, INGRID will set `use_efit_bounds` to `True`.

Below are figures illustrating possible edits to the parameter-file `limiter` block entry (we will **not** be using these values for the remainder of the tutorial).

First, setting `use_efit_bounds` to `False` (default `eqdsk` limiter data)


Normalized Efit Data
Z
R
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.00
1.25
1.50
1.75
2.00
2.25
2.50
magx
xpt1
xpt2
limiter
midline_1
midline_2
psi_core
psi_1
psi_2
psi_pf_1
psi_pf_2
Primary Separatrix
Secondary Separatrix

Next, setting `use_efit_bounds` back to `True`, but setting both `efit_buffer_r` and `efit_buffer_z` back to their **default** values of `1.0e-2`.


Normalized Efit Data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Z
1.00
1.25
1.50
1.75
2.00
2.25
2.50
R
magx
xpt1
xpt2
limiter
midline_1
midline_2
psi_core
psi_1
psi_2
psi_pf_1
psi_pf_2
Primary Separatrix
Secondary Separatrix

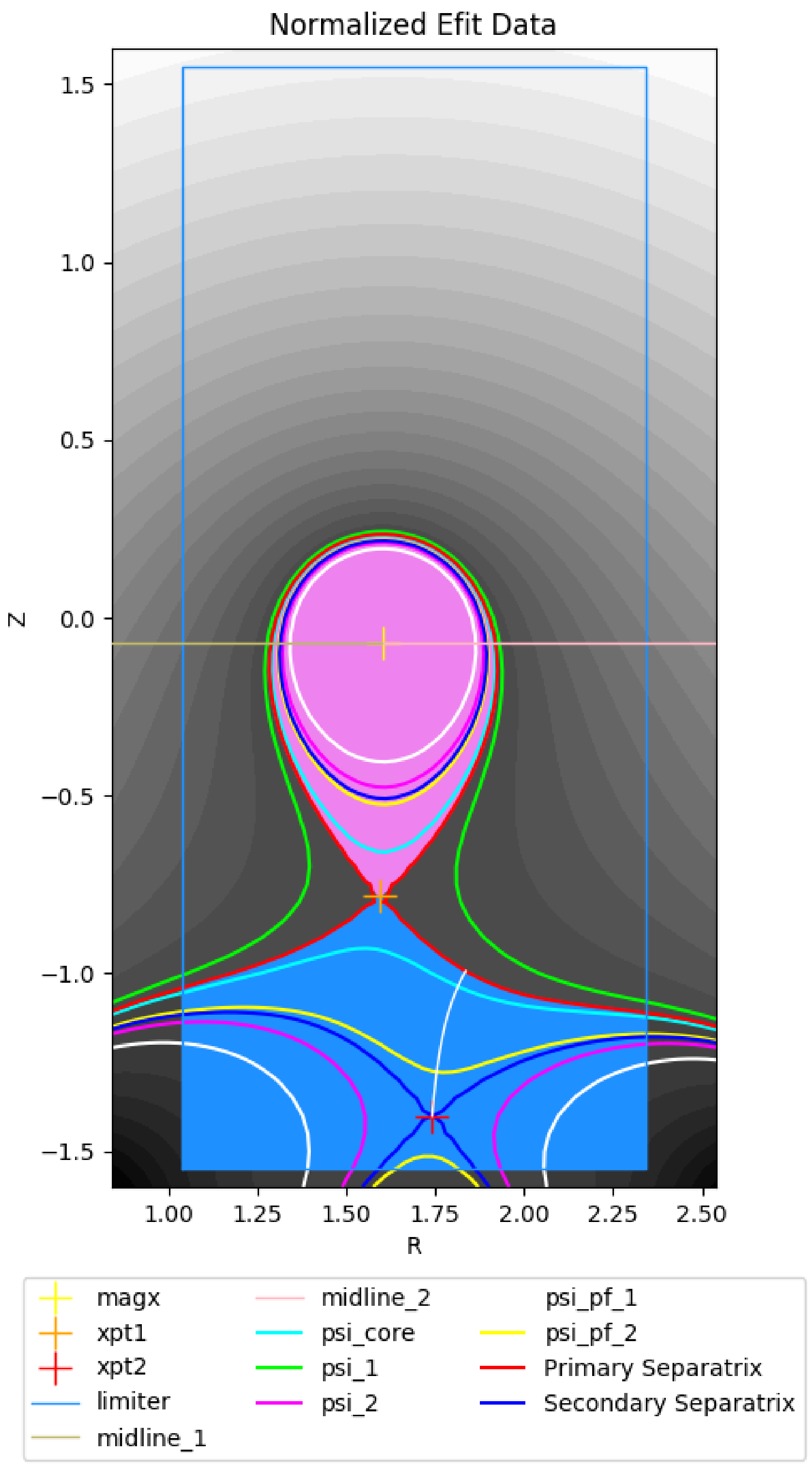
Warning: When setting `use_efit_bounds` to `True`, the user must provide a non-zero value for values `efit_buffer_r` and `efit_buffer_z`.

2.6.4 INGRID identification of configuration

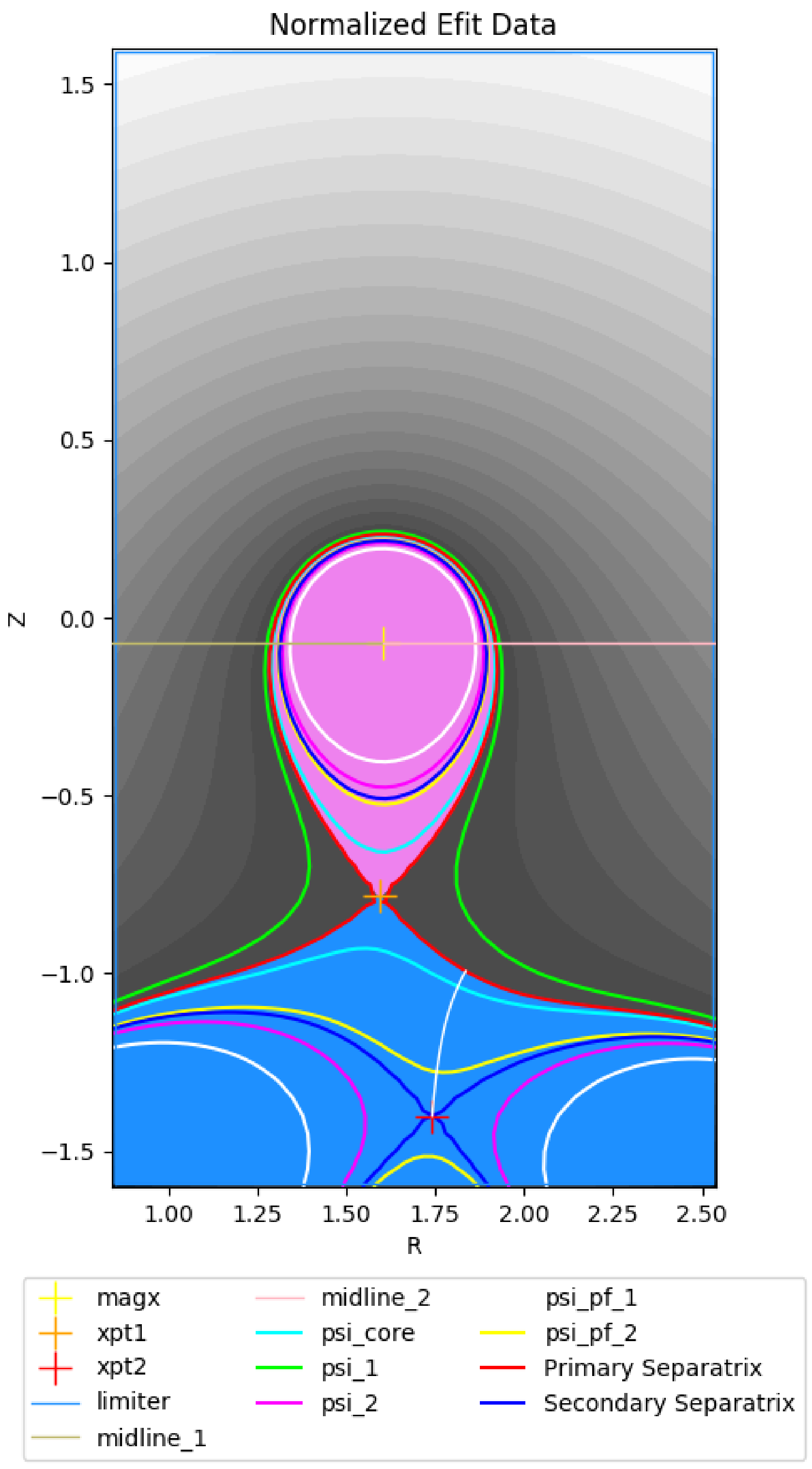
As mentioned in the previous section, understanding INGRID limiter usage is an essential part of generating a Patch map for cases with two x-points in the domain.

This is because INGRID identifies a configuration by decomposing subset of the domain contained within the limiter into three distinct regions: core, private-flux, and separatrix exterior.

We see this below with the core shaded in magenta, private-flux shaded in blue, and separatrix exterior with no filled shading.


Normalized Efit Data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Z
1.00
1.25
1.50
1.75
2.00
2.25
2.50
R
magx
xpt1
xpt2
limiter
midline_1
midline_2
psi_core
psi_1
psi_2
psi_pf_1
psi_pf_2
Primary Separatrix
Secondary Separatrix

To emphasize the dependence on the limiter for decomposition, we also illustrate with setting both `efit_buffer_r` and `efit_buffer_z` back to their **default** values of `1.0e-2`.

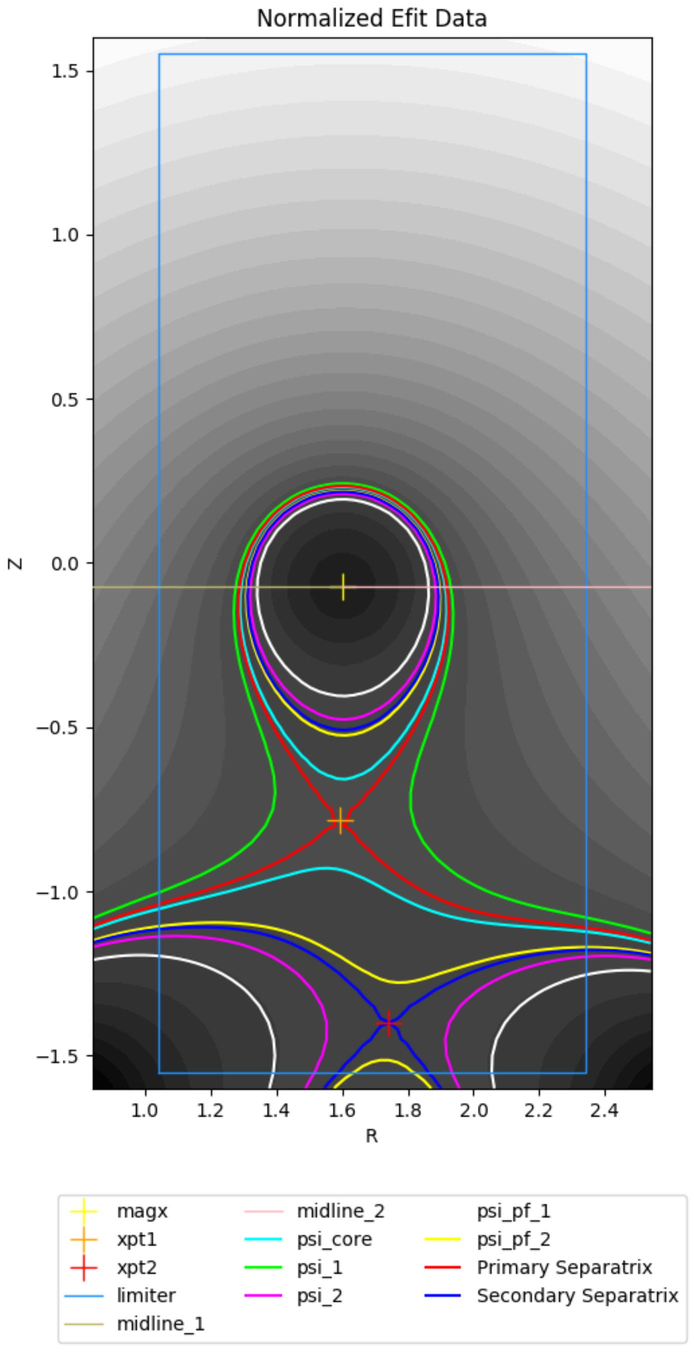


Normalized Efit Data
Z
R
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.00
1.25
1.50
1.75
2.00
2.25
2.50
magx
xpt1
xpt2
limiter
midline_1
midline_2
psi_core
psi_1
psi_2
psi_pf_1
psi_pf_2
Primary Separatrix
Secondary Separatrix

It is up to the user to ensure the following is satisfied for successful identification of magnetic topology:

- The provided limiter forms a closed loop (note EFIT bounds do this by default and is therefore a useful tool for classification)
- The Magnetic-axis is contained within the closed limiter geometry
- The primary x-point is contained within the closed limiter geometry
- The primary separatrix “legs” intersect the limiter walls and form a closed region
- The secondary x-point is contained within the closed limiter geometry

We can indeed see that our parameter file has been preset to satisfy all of the above.


Normalized Efit Data
Z
R
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
magx
xpt1
xpt2
limiter
midline_1
midline_2
psi_core
psi_1
psi_2
psi_pf_1
psi_pf_2
Primary Separatrix
Secondary Separatrix

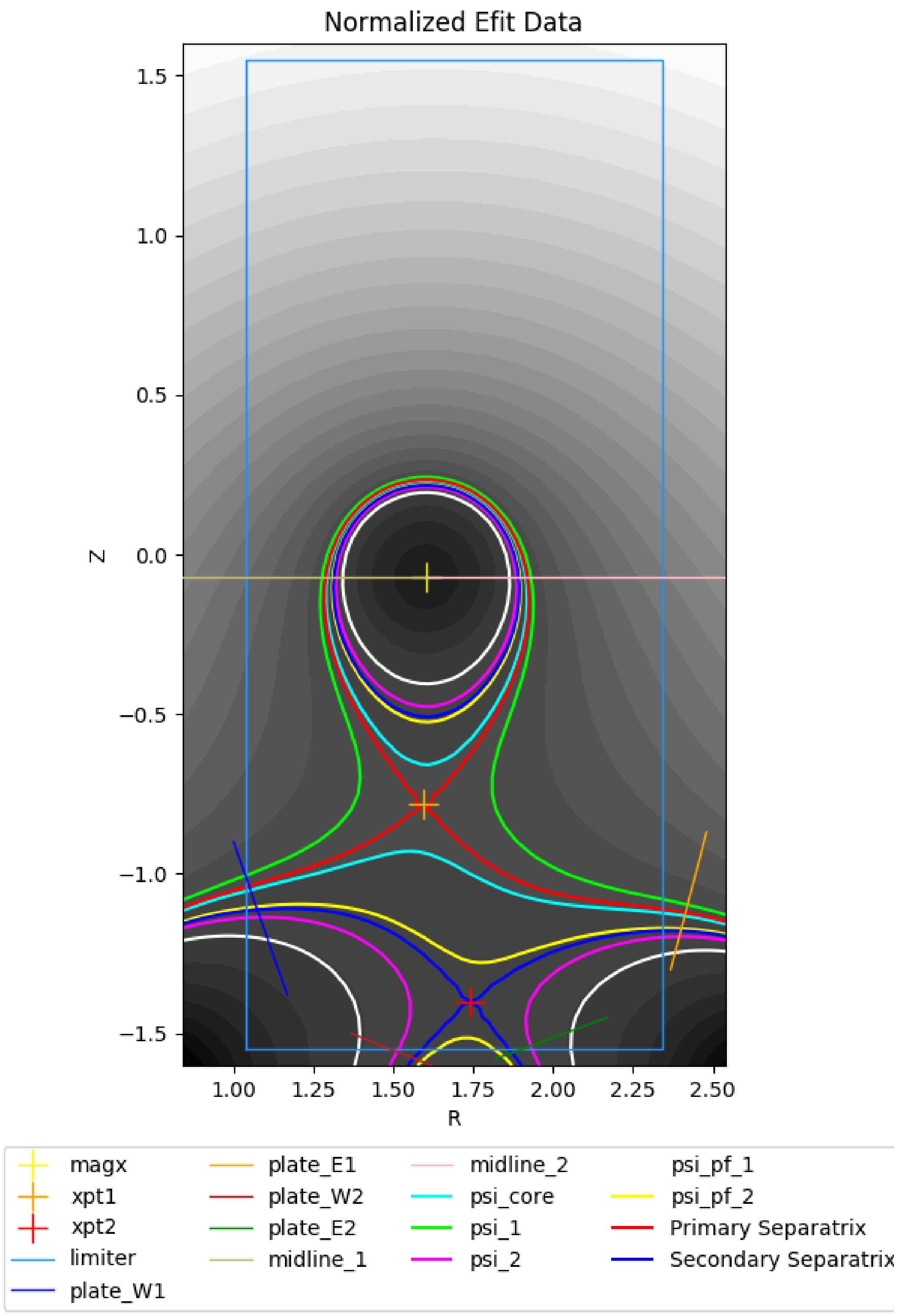
2.6.5 Using target-plates for generating a Patch map

The user can opt for using target-plates rather than a limiter for generating a Patch map for cases with two x-points in the domain. Setting the entry `strike_pt_loc` to `target_plates` tells INGRID to generate the Patch map using geometry provided in the parameter file block `target_plates`. We see this snippet below.

```
grid_settings:

  # ...
  # other settings
  # ...

  patch_generation:
    # ...
    # other settings
    # ...

    # strike_pt_loc takes the values of 'limiter' or 'target_plates'
    strike_pt_loc: target_plates  # generates a Patch map with a target_plates rather␣
↪than limiter block

# Specifications for using target plates
target_plates:

  # Target plate E of xpt1
  plate_E1:
    file: ../data/SF75/plate_E1.txt
    rshift: 1.0
    zshift: 0.28

  # Target plate E of xpt2
  plate_E2:
    file: ../data/SF75/plate_E2.txt
    rshift: 0.45
    zshift: 0.00

  # Target plate W of xpt1
  plate_W1:
    file: ../data/SF75/plate_W1.txt
    rshift: -0.2
    zshift: 0.2

  # Target plate W of xpt2
  plate_W2:
    file: ../data/SF75/plate_W2.txt
    zshift: 0.0
    rshift: 0.00
```

The user can refer to the diagrams earlier in the tutorial to see where the target plates above should reside in the domain.

By making the above edits, we then refresh our view of the data to obtain the following plot.


Normalized Efit Data
1.5
1.0
0.5
Z
0.0
−0.5
−1.0
−1.5
1.00
1.25
1.50
1.75
2.00
2.25
2.50
R
magx
xpt1
xpt2
limiter
plate_W1
plate_E1
plate_W2
plate_E2
midline_1
midline_2
psi_core
psi_1
psi_2
psi_pf_1
psi_pf_2
Primary Separatrix
Secondary Separatrix

Indeed, we see there are now four target plates in the EFIT domain.

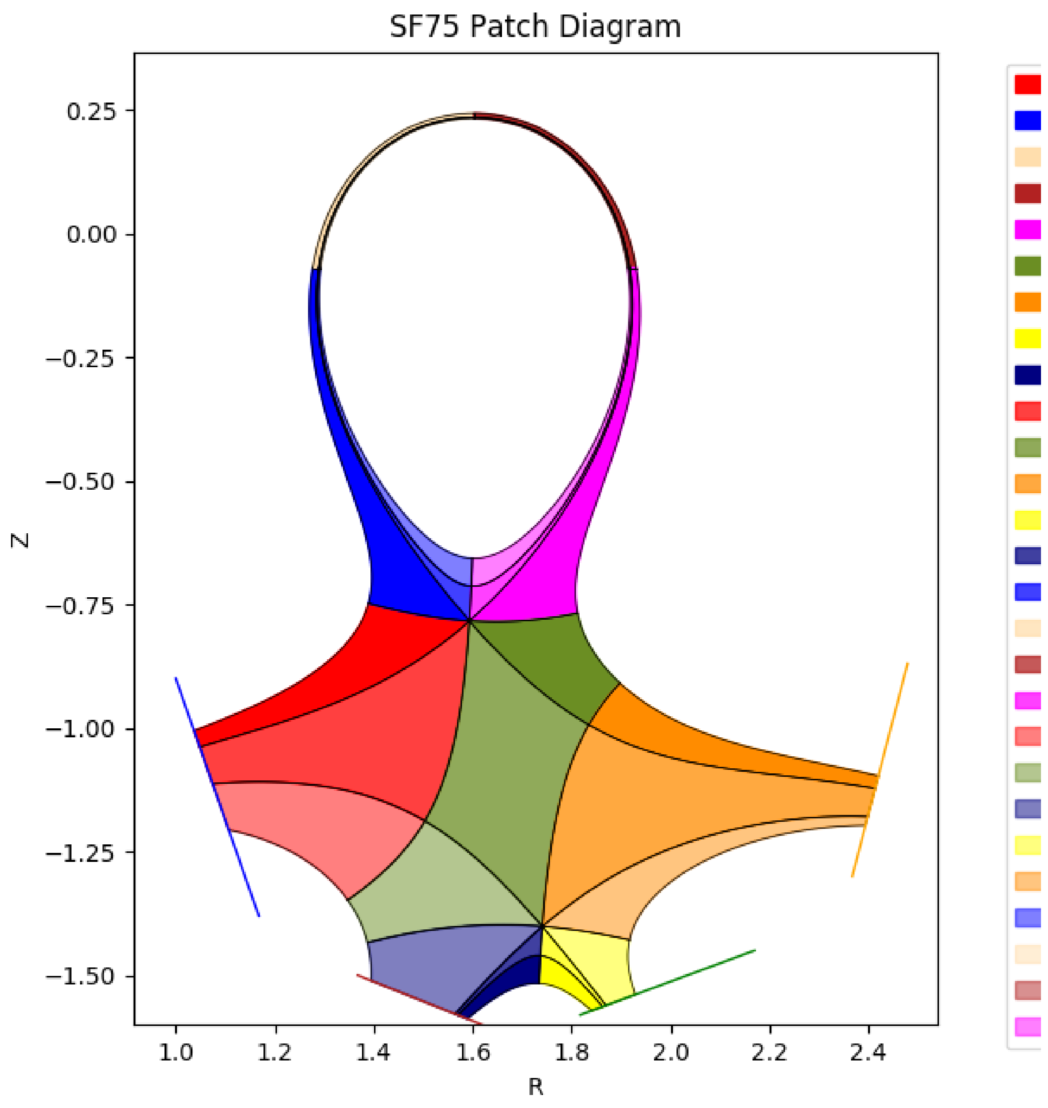
Note: The presence of the limiter in the figure is not a bug. Regardless of whether we decide to utilize the limiter or target plates for generating a Patch map, INGRID still relies on the limiter to classify the magnetic topology. This is why the user must be comfortable with limiter controls.

2.6.6 Creating the Patch map and grid

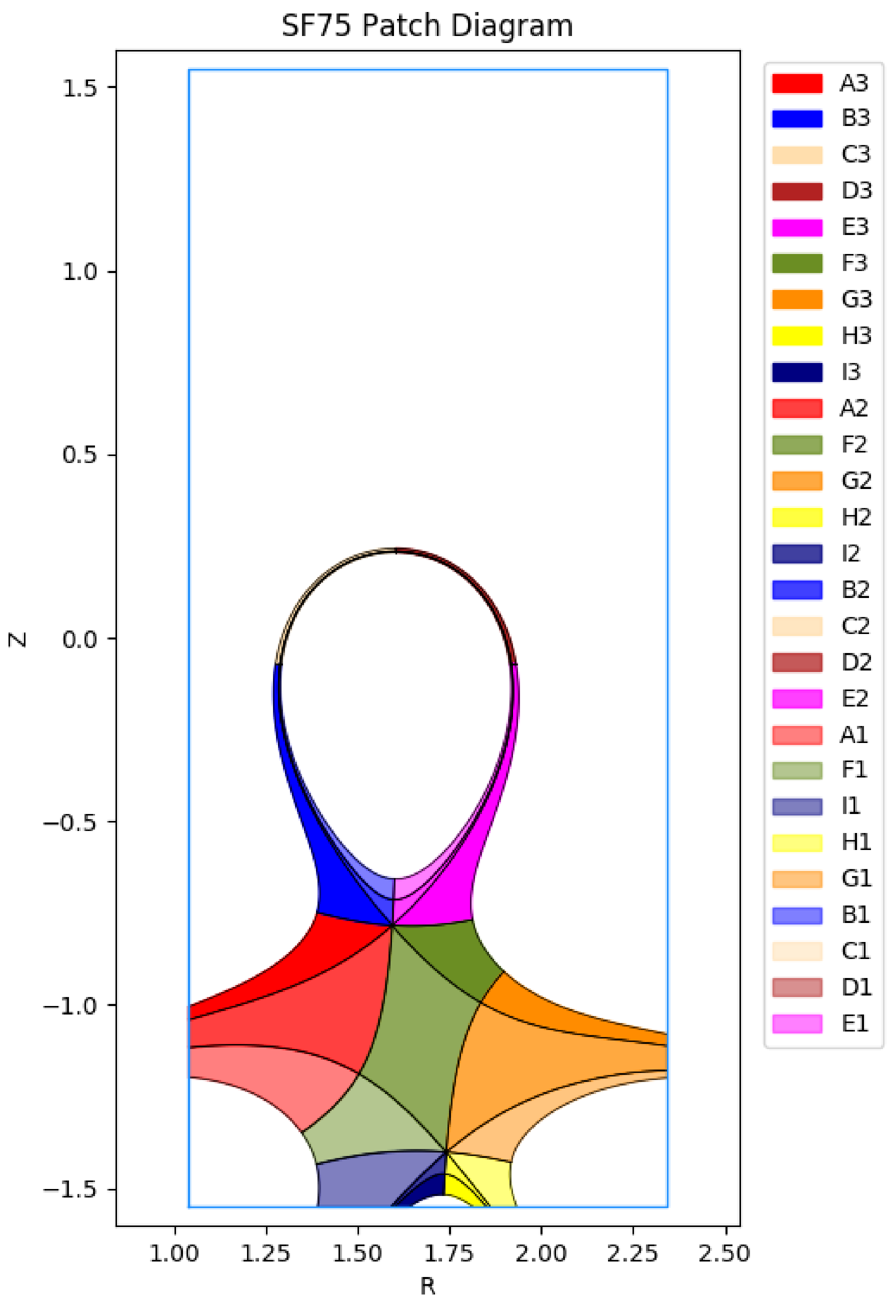
The majority of the effort for generating a Patch map goes into the preparation of the EFIT data. We indeed saw this in the previous sections. From here, we carry out the usual process of generating a Patch map that we saw in the SNL case.

All of the Patch map edit capabilities that we performed on the `SNL` cases are also available for all other cases with no further explanation.

We proceed with Patch map generation and obtain the following Patch map when keeping `strike_pt_loc` set with a value of `target_plates`.

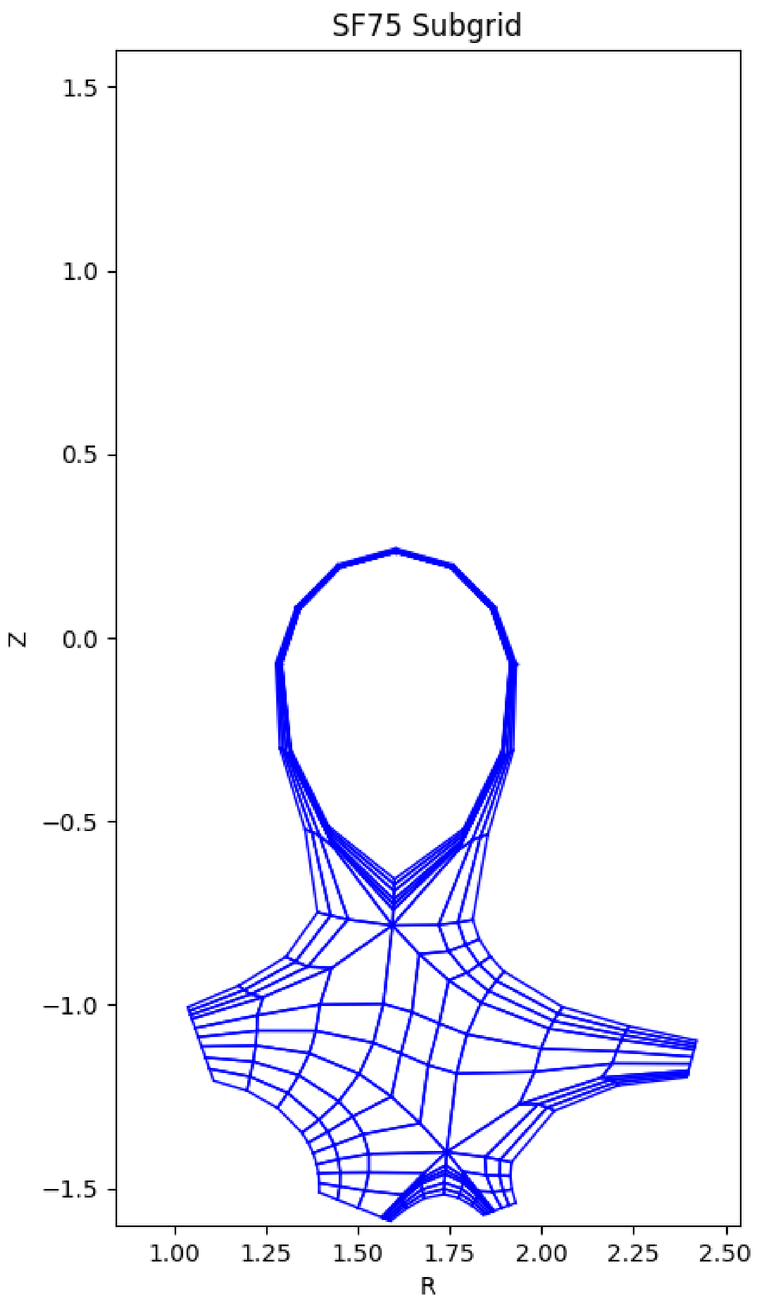

SF75 Patch Diagram
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
Z
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
R

If we had kept `strike_pt_loc` set to `limiter`, we would obtain the Patch map below.

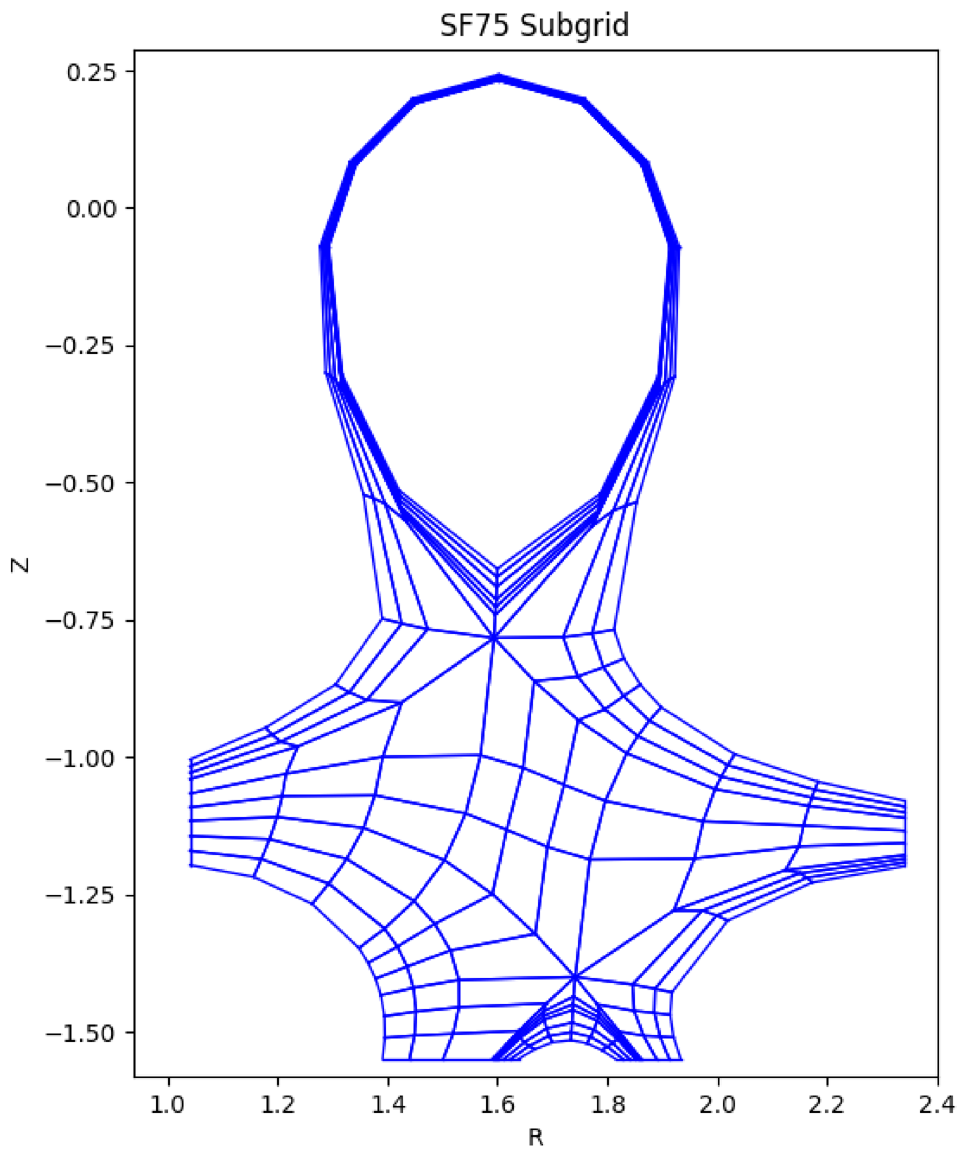

SF75 Patch Diagram
Z
R
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.00
1.25
1.50
1.75
2.00
2.25
2.50
A3
B3
C3
D3
E3
F3
G3
H3
I3
A2
F2
G2
H2
I2
B2
C2
D2
E2
A1
F1
I1
H1
G1
B1
C1
D1
E1

All grid generating instructions detailed in the `SNL` case also apply to the `SF` cases. The only difference is the addition of an additional control `nr_3` that controls the Patch objects with name ending with `3`.

Using `np_default` and `nr_default` with values of `3` produces the following grid when using target plates.


SF75 Subgrid
Z
R
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.00
1.25
1.50
1.75
2.00
2.25
2.50

Using `np_default` and `nr_default` with values of `3` produces the following grid when using the limiter.


SF75 Subgrid
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
Z
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
R

All other grid generation customization tools such as distortion_correction, poloidal_f_A - poloidal_f_I, and radial_f_1 - radial_f_3 are utilized in the same ways we saw in the earlier SNL cases.

CHAPTER

THREE

MODULE DOCUMENTATION

3.1 ingrid module

3.2 topologies package

3.2.1 Submodules

3.2.2 topologies.sf105 module

3.2.3 topologies.sf135 module

3.2.4 topologies.sf15 module

3.2.5 topologies.sf165 module

3.2.6 topologies.sf45 module

3.2.7 topologies.sf75 module

3.2.8 topologies.snl module

3.2.9 topologies.udn module

3.2.10 Module contents

3.3 utils module

3.4 geometry module

3.5 line_tracing module

3.6 interpol module

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search