USERS' MANUAL OF THE INVERSE EQUILIBRIUM TOOL (IET)

IET logo designed by riccardopirotto.com

Corresponding author

Name: Domenico Abate

Phone: 0039 049 829 5074

e-mail: domenico.abate@igi.cnr.it

Affiliations:

- Consorzio RFX (CNR, ENEA, INFN, Università di Padova, Acciaierie Venete SpA), C.so Stati Uniti 4, 35127, Padova, Italy,
- Centro Ricerche Fusione – Università degli studi di Padova, Padova, Italy

ABSTRACT

In this document we present a documentation for using the so-called inverse equilibrium tool (IET) by means of simple Graphical User Interfaces programmed in MATLAB environment.

IET allows to compute the coil currents needed to obtain a predetermined plasma shape with defined plasma global parameters (i.e. total plasma current and total poloidal magnetic flux at the boundary) by solving a constrained minimization problem.

The IET computational tool is constituted by three main modules, each with a dedicated GUI:

1. The IET SHAPING TOOL: allowing to define/characterize the plasma boundary and to produce the mesh of the plasma domain;
2. The IET EQUILIBRIUM TOOL: allowing to solve the fixed boundary equilibrium problem for the designed plasma shape with arbitrarily define the plasma current density profile;
3. The IET OPTIMIZATION TOOL: allowing to determine and optimize the active coil currents needed to produce the defined plasma equilibrium within the engineering capabilities of the device.

IET was validated for both ITER-like plasma configurations and experimental plasmas in the RFX-mod experiment.

The journal publication of IET code can be here: https://iopscience.iop.org/article/10.1088/1361-6587/ab3f09

CHANGE RECORD

Version	Date	Change Record Description	Authors
v1.00	October 2019	Draft version	D. Abate

Table of Contents

1 IET INSTALLATION AND SYSTEM REQUIREMENTS

The inverse equilibrium tool IET code is constituted by three modules each of its relative GUI: the IET SHAPING TOOL module, IET EQUILIBRIUM TOOL module and the IET OPTIMIZATION TOOL module. The main IET GUI is very simple and launches the three different modules.

IET can be freely downloaded https://github.com/DA2412/IET

For the installation of the stand-alone versions of IET it is required a Windows, Linux or Mac platform and the MATLAB Runtime environment or alternatively an internet connection; in fact, the executable file of the IET code is provided with a MATLAB Runtime Environment installer which requires an internet connection.

The details on how to use each of the three IET modules will be described in next dedicated sections. For any comment about the code and the user's guide please contact domenico.abate@igi.cnr.it.

2 INSTALLATION

IET is provided as an executable file for each OS (i.e. Windows, Linux) in dedicated directories. Each IET version requires the MATLAB Runtime environment (included via internet connection in the IET executable).

2.1 Prerequisites for Deployment

Verify that version 9.7 (R2019b) of the MATLAB Runtime is installed. If not, you can run the MATLAB Runtime installer included in the IET executable; see subsections for details.

2.2 Windows

The installation of MATLAB Runtime can be started by executing the file *IET_Installer_web.exe* which is located in "*IET_StandAloneWin_v1/for_redistribution/IET_Installer_web.exe*"; the installer requires the internet connection.

After that the IET code can be launched by using the installed version (double clicking the icon on desktop) or by launching the standalone executable file in "*IET_StandAloneWin_v1/for_testing/IET.exe.*

2.3 Linux

The installation of MATLAB Runtime can be started by executing the file *IET_Installer_web.install* which is located in "*IET_StandAloneLinux_v1/for_redistribution/ IET_Installer_web.install*"; the installer requires the internet connection.

After that the IET code can be launched by launching the standalone executable file in this way: run the shell script as:

./run_IET.sh <mcr_directory>

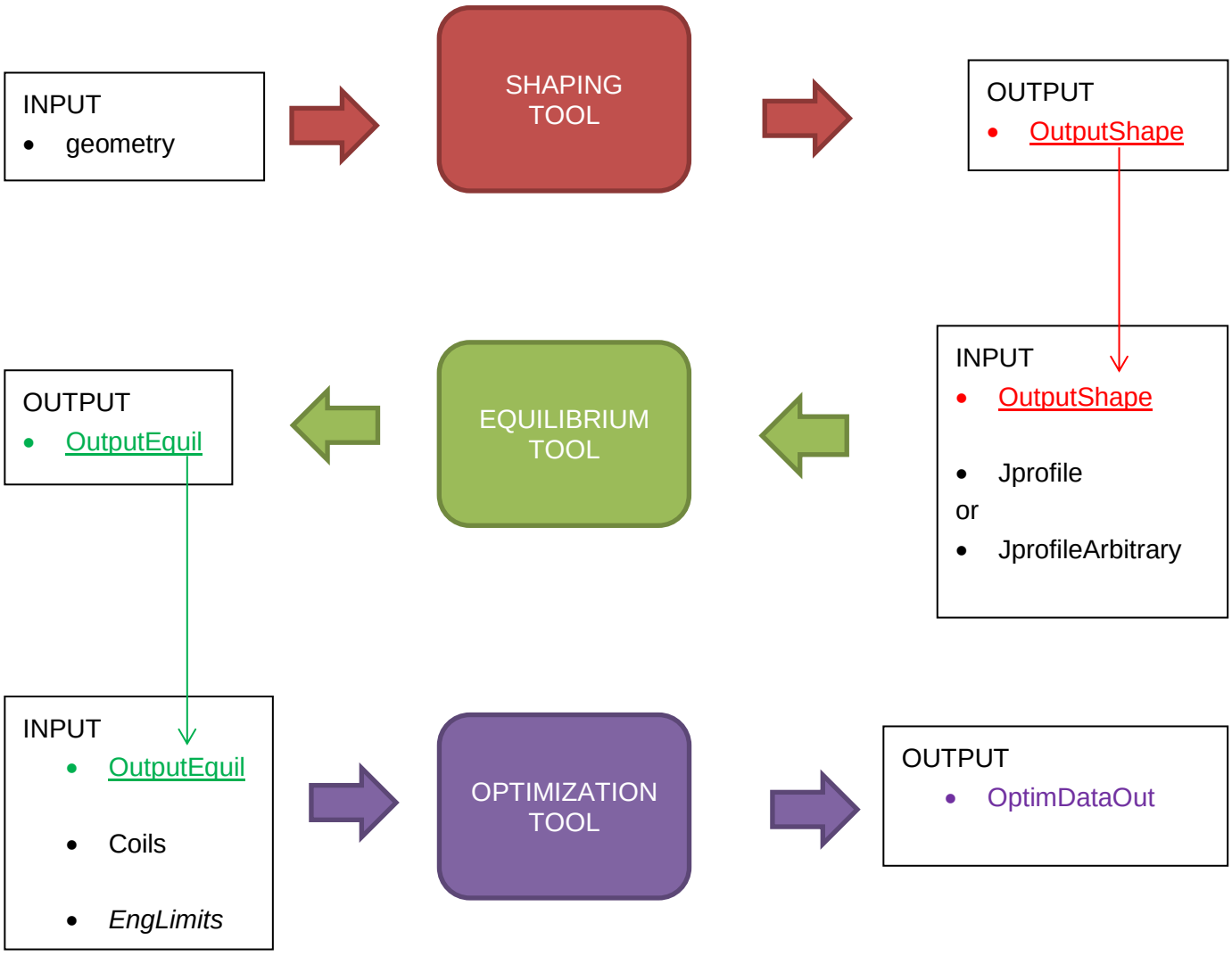
Where <mcr_directory> is the directory where version 9.7 of the MATLAB Runtime is installed or the directory where MATLAB is installed on the machine.

For example, if you have version 9.7 of the MATLAB Runtime installed in /mathworks/home/application/v97, run the shell script as:

./run_IET.sh /mathworks/home/application/v97

3 IET STRUCTURE

IET Input and output variables are described in details for each module and summarized in the scheme below.



4 IET SHAPING TOOL

4.1 Input and Output quantities

The Input MAT file should include a structure variable called *geometry* with the following fields:

geometry. boundaryCoordinates [Nx2]: column vectors of R and Z plasma boundary coordinates;

geometry. FW [Nx2] : column vectors of R and Z first wall coordinates;

The Output MAT file will include a structure variable called *OutputShape* with the following fields:

OutputShape. p [N ×2] : column vectors of R and Z coordinates of the mesh

OutputShape. t: [N ×3] : index of mesh elements vertices

OutputShape. b: [M ×1] : index of boundary elements

OutputShape. R_boundary_finale [1×N]: row vector of R coordinate of the plasma boundary

OutputShape. Z_boundary_finale [1×N]: row vector of Z coordinate of the plasma boundary

OutputShape. Area_duale [54×1 double]: row vector of Area of the dual mesh needed to solve equilibrium

OutputShape. kk_nu [54×54 double]: stiffness matrix needed to solve equilibrium needed to solve equilibrium

4.2 GUI description

The Graphical User Interface (GUI) of the IET SHAPING TOOL is described with the following examples.

4.2.1 How to characterize an existing boundary

First it is needed to load an existing reference plasma boundary by clicking the “Load reference boundary” button, as represented in Fig. 1 (a). Thus, a dialog window opens and the reference boundary should be loaded in terms of MAT file which must include the geometry structure variable as described in Sect. **Error! Reference source not found.**. The reference plasma boundary will appear in magenta line as represented in Fig. 1 (b) and the plasma shape parameters will automatically be computed and displayed as highlighted by the cyan shaded area.

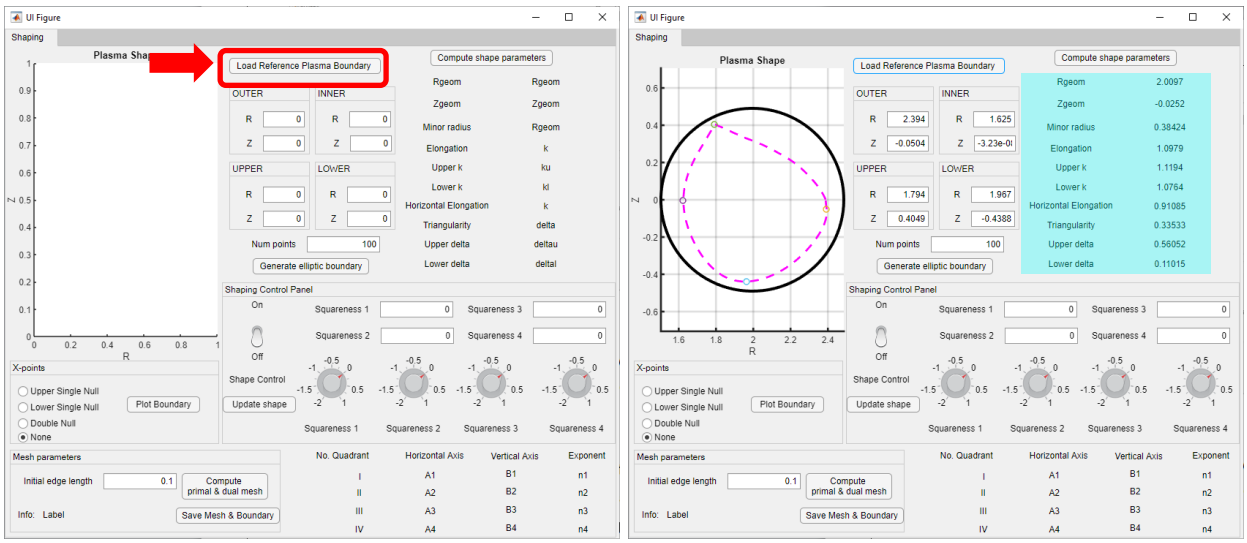


Fig. 1 (a) **(b)**

Then, by clicking the Generate elliptic boundary button, 4 branches of ellipses will be produced and plotted in red line as shown in Fig. 2 (a); these are needed as reference for the superellipses definitions. The number of points used to produce the boundary can be specified by changing the default number of "100" in the Num points box. By enabling the 'shaping control panel', Fig. 2 (b), the four branches of superellipses are computed and plotted in the four different quadrants with each diagonal intersecting the reference plasma boundary. The user can also modify the squareness of each branch of superellipse generating a new family of boundaries in terms of curvature and shape but preserving all the standard shape parameters (i.e. elongation, triangularity, major radius, minor radius).

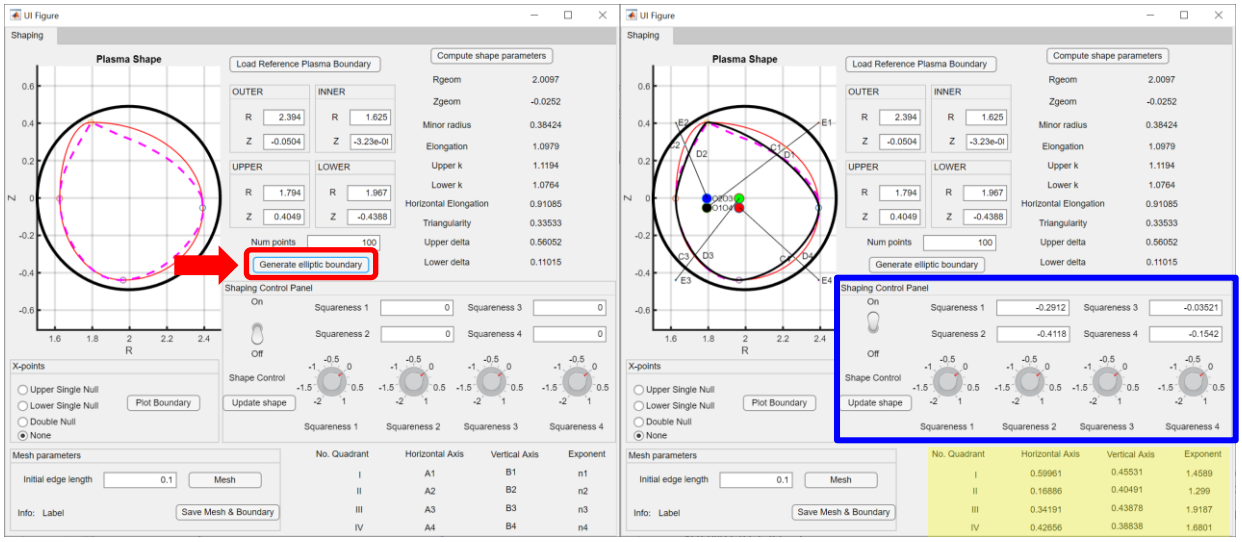


Fig. 2 (a) **(b)**

The parameters defining the equation of each superellipse branches (i.e. $(x/A)^n + (y/B)^n = 1$) are displayed in lower right corner of the GUI (yellow shaded area in Fig. 2 (b)). Single (lower and upper) and double null configurations can be obtained by using the appropriate 'X-points panel' as shown in Fig. 3 by the blue line.

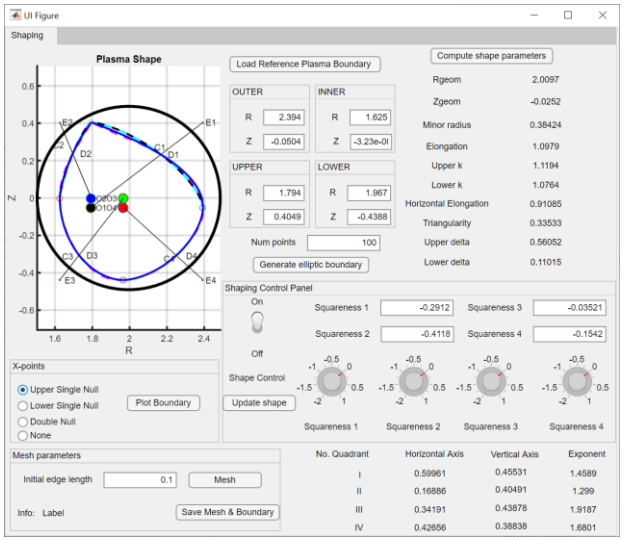


Fig. 3 (a)

4.2.2 How to design a new plasma boundary

In the case of a full design, the user is free to define the four extremal points of the plasma boundary; this can be useful for several cases, such as scaling an old equilibrium to new vacuum chamber dimensions after modifications of the experiment; or to reverse the triangularity of an existing equilibrium in order to assess if such configuration can be produced by the active coils; or to assess the capability of the device to produce a X-point(s) starting from an existing limiter plasma boundary.

For example we would reverse the triangularity of the reference plasma in Fig. 3 by changing the R coordinates of the Upper and Lower extremal points as shown in Fig. 4 (a); in this way the elongation is kept constant and one can acts only on the desired shape parameter. After that, the reference ellipse is produced Fig. 4 (b). Once the shaping panel is applied, the superellipses branch are equal to the ellipses ones (since there is no reference boundary, in fact the squareness are all zero i.e. same as ellipse) as shown in Fig. 5 (a) black line. Thus, by acting on the squareness (>0 more square than ellipse, <0 less square than ellipse) one can produce the desired shape Fig. 5 (a) cyan line. Then, by imposing the Upper Single Null Xpoint condition one obtains the final USN plasma boundary, similar to the starting existing reference one but with negative triangularity, Fig. 5 (b) blue line.

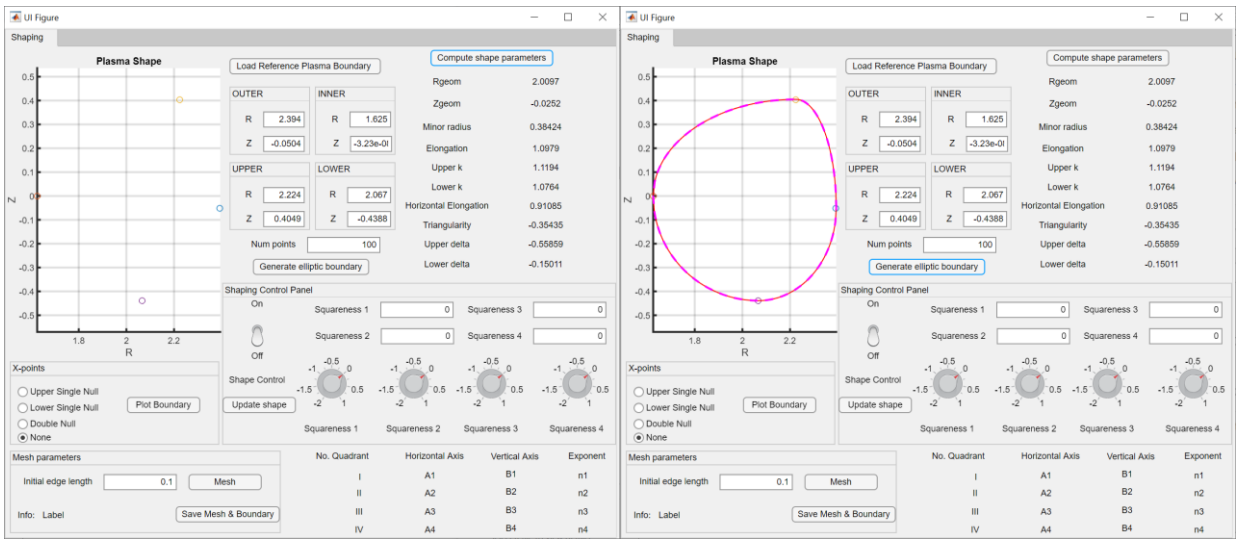


Fig. 4 (a) **(b)**

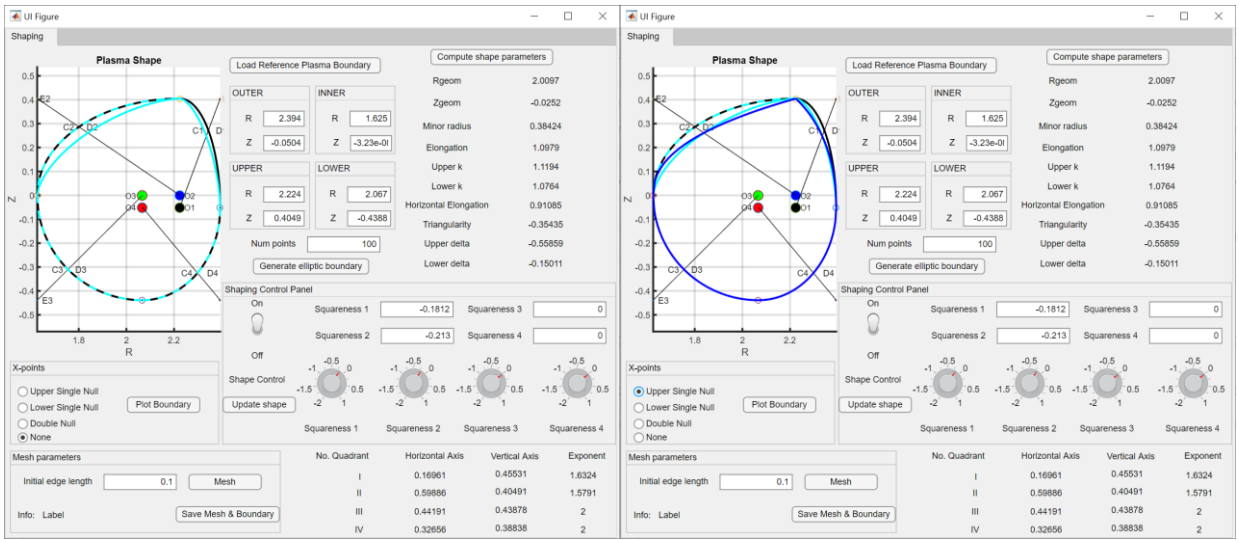


Fig. 5 (a) **(b)**

The final boundary is represented in Fig. 6 and can now be meshed by the meshing tool.

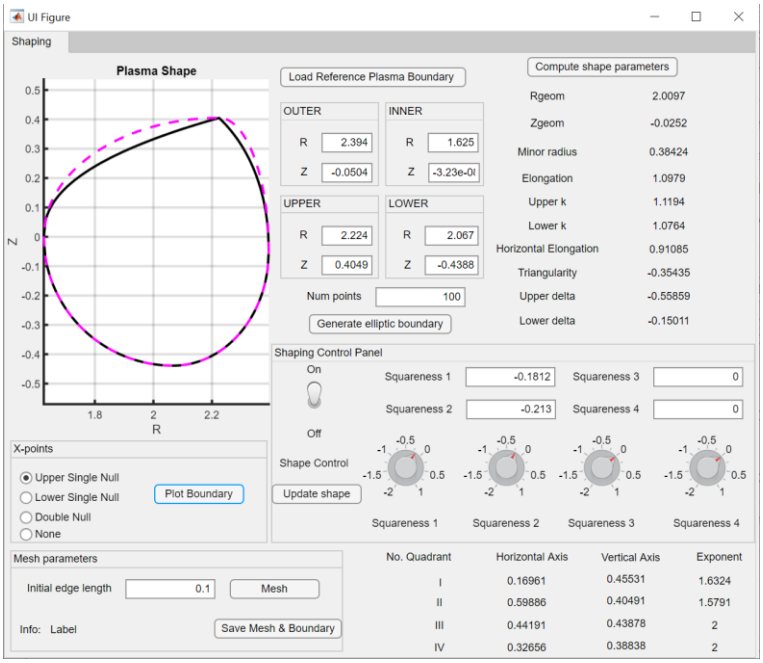


Fig. 6

4.3 Mesh of the plasma domain

The mesh of the plasma domain is computed by the distmesh code (https://github.com/ionhandshaker/distmesh , Per-Olof Persson et al. *SIAM Rev.*, 46(2), 329–345. https://doi.org/10.1137/S0036144503429121) by prescribing the initial edge length. The default value is 0.1 but a higher resolution can be obtained by lower values. The meshing tool could take some seconds, depending on the resolution you prescribed. The final result example is represented in Fig. 7. By clicking on the Save Mesh & boundary button, you will save the OutputShape structure containing both the boundary coordinates and the mesh in a MAT file.

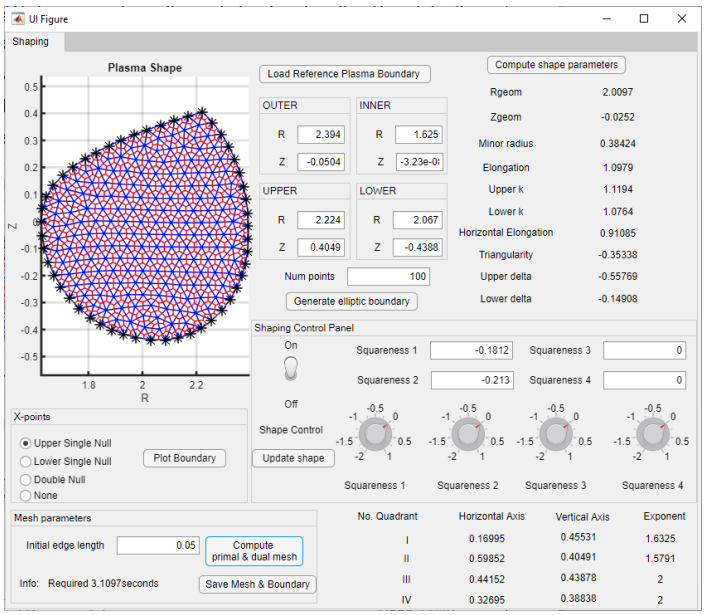


Fig. 7

5 IET EQUILIBRIUM TOOL

5.1 Input and Output quantities

The input MAT file should include the structure variable called *OutputShape* produced by the IET SHAPING TOOL module which includes the plasma mesh and boundary details.

In order to solve the fixed boundary equilibrium problem, the plasma current density profile has to be defined. IET gives you three different ways to do that:

- Shafranov's parametrization: by prescribing the P' and FF' profiles as a function of the normalized poloidal flux. In this case the input MAT file should include a structure variable called *Jprofile* with the following fields:
  - *Pprime* [1×101 double]: pressure gradient profile
  - *FFprime* [1×101 double]: FF' function profile
  - *Psibar* [1×101 double]: normalized poloidal magnetic flux (0 at axis 1 at boundary).
- Blum's parametriztion: by prescribing three parameters, following (Blum, J. (1989). Numerical simulation and optimal control in plasma physics. United States: John Wiley and Sons Inc.), which have to be defined by inserting the related values in the GUI fields.
- Arbitrary profile by points: by prescribing a set of current density profile points which will be interpolated by the code; the profile can be provided with a MAT file that should include the variable *JprofileArbitrary* [Npointsx2] whose columns are the normalized poloidal flux and the plasma current density profile.

The output of the IET Equilibrium Tool is a MAT file containing the structure variable *OutputEquil* whose fields are:

- *OutputEquil.OutputShape* = containing the same fields of the *OutputShape* variable produced by the Shaping tool
- *OutputEquil. Equil* = containing the following fields:
  - *iter*: 8 Number of iteration for convergence
  - *psi_tot_direct*: [2329×1 double] poloidal magnetic flux solution
  - *errorAxis*: [8×1 double] error at magnetic axis as a function of iteration
  - *EqSolution.Raxis* [1x1 double] magnetic axis R coordinate
  - *EqSolution.Zaxis* [1x1 double] magnetic axis Z coordinate
  - *EqSolution.Psi_axis* [1x1 double] poloidal flux at magnetic axis
  - *Iphi_plasma*: [2329×1 double] plasma current at node of mesh (i.e. plasma current density profle)
  - *RR*: [500×500 double] square grid for plotting psi contours

- *ZZ*: [500×500 double] square grid for plotting psi contours
- *PSI_final*: [500×500 double] magnetic flux at square grid points for contour
- *JP*: [1×1 struct] structure for patching J profile
- *psin*: [2329×1 double] normalized poloidal flux for replotting the P',FF' profiles
- *Jphi*: [2329×1 double]: 1D J profile related to psin
- *Ipla* [1x1]: total plasma current value
- *psiBoundary* [1x1]: total poloidal magnetic flux at boundary

5.2 GUI description

The first step is to load the plasma domain boundary and mesh, previously produced by the IET Shaping Tool. If it is necessary, the code automatically computes the dual mesh needed to solve the equilibrium; this phase could be particularly time consuming depending on the dimension of the mesh. The new IETShapingToolOutput.mat file will be produced with the new fields "Area_duale" and "kk_nu".

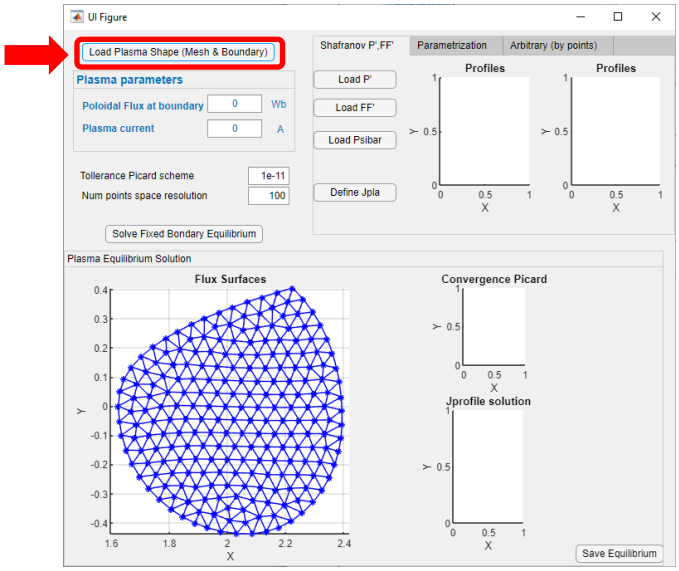


Fig. 8

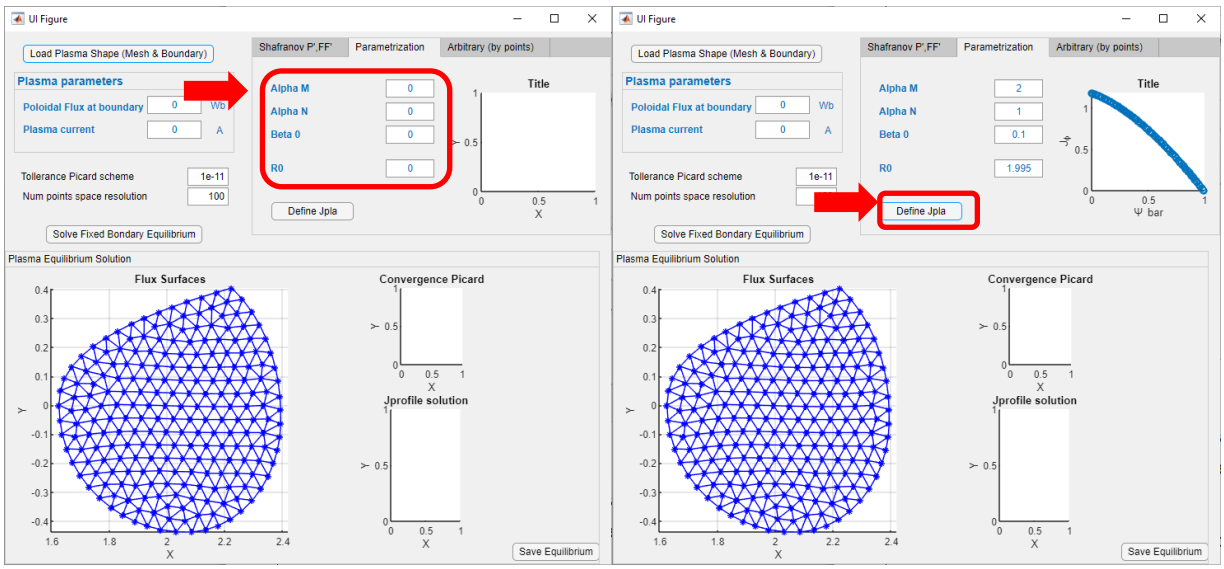


Fig. 9

After that, the plasma current density profile can be defined in different ways. As example we present the definition via parametrization (FIG 9).

Then it is needed to specify the total poloidal magnetic flux at boundary and the total plasma current. After that the “Solve fixed boundary equilibrium” button has to be pushed.

The solution will be represented in term of poloidal magnetic flux surfaces, plasma current density patch map and picard convergence plot (fig. 11).

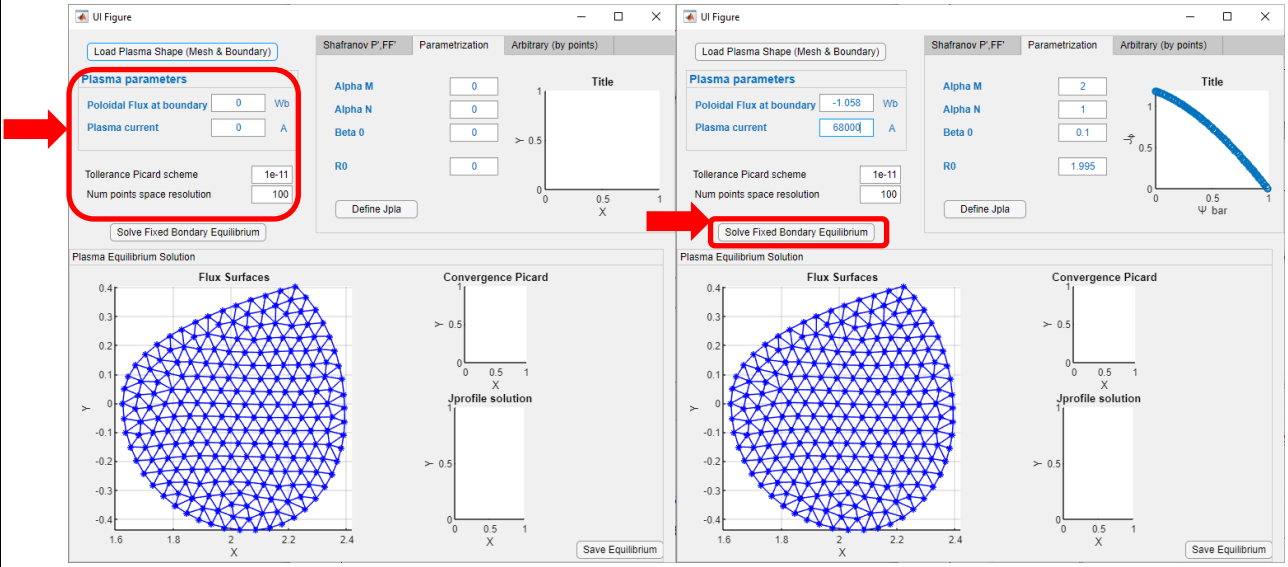


Fig. 10

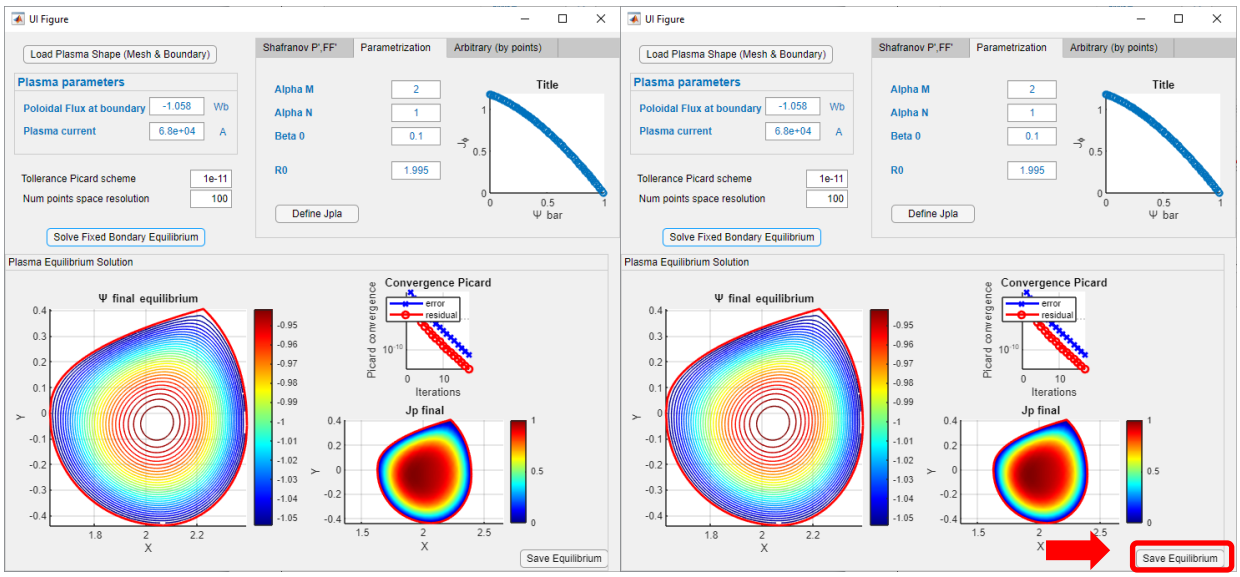

UI Figure
Load Plasma Shape (Mesh & Boundary)
Plasma parameters
Poloidal Flux at boundary
-1.058
Wb
Plasma current
6.8e+04
A
Tollerance Picard scheme
1e-11
Num points space resolution
100
Solve Fixed Bondary Equilibrium
Shafranov P',FF'
Parametrization
Arbitrary (by points)
Alpha M
2
Alpha N
1
Beta 0
0.1
R0
1.995
Define Jpla
Title
Ψ bar
Plasma Equilibrium Solution
Ψ final equilibrium
Convergence Picard
error
residual
Picard convergence
Iterations
Jp final
Save Equilibrium

Fig. 11

6 IET OPTIMIZATION TOOL

6.1 Input and Output quantities

The first Input MAT file should include the structure variable called *OutputEquil,* previously generated with IET equilibrium tool.

The second Input MAT file that defines the machine geometry should include a structure variable called *coils* constituted by the following fields:

- *coils .R: [1×56 double]*
- *coils .Z: [1×56 double]*
- *coils .DR: [1×56 double]*
- *coils .DZ: [1×56 double]*
- *coils .turns: [1×56 double]*
- *coils .kon: [14×56 double]*

The second Input MAT file that defines the machine geometry should include a structure variable called *EngLimits* constituted by the following fields:

- *EngLimits .UB: [1×12 double]*
- *EngLimits .LB: [1×12 double]*

The Output file will include the struct variable called *OptimDataOut* which includes the following fields:

- OptimDataOut. SOF_solution: [1×1 struct]
    - OptimDataOut .SOF_solution .currents: [14×1 double]
    - OptimDataOut .SOF_solution . Psi_LS: [232×1 double]
    - OptimDataOut .SOF_solution . err_patch_LS: [232×1 double]

6.2 GUI description

Firstly the equilibrium and machine geometry needs to be loaded (Fig. 12). Then, with prepare data button, the poloidal flux contribution of the active coil currents will be computed and displayed (Fig.13). If the green circles do not fit the magenta asterisk than there is a problem.

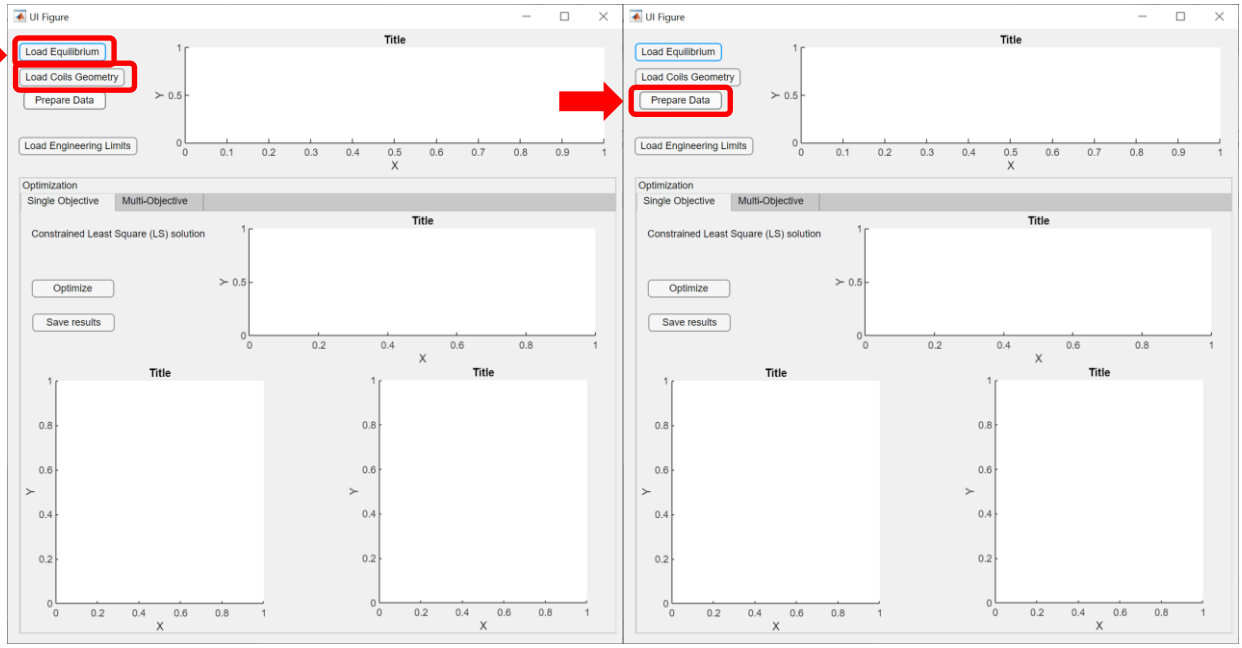


Fig. 12

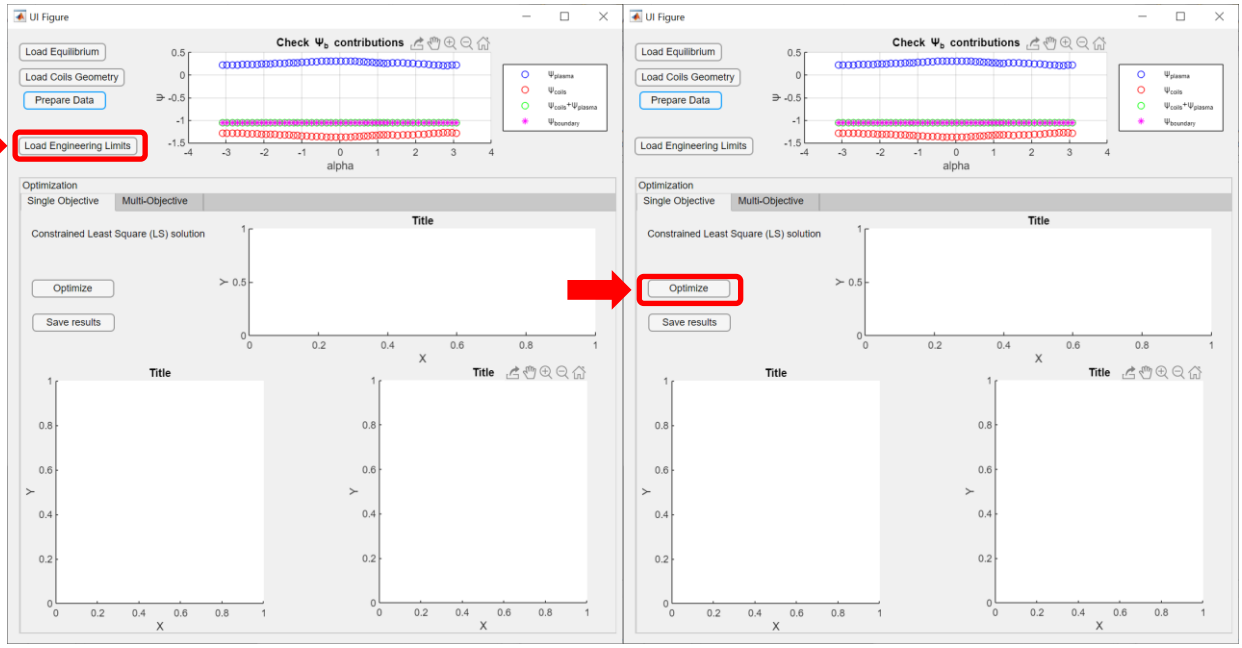


Fig. 13 (a) **(b)**

The engineering limits in terms of maximum current for each coil has to be loaded (13 a).

Clicking the optimize button will solve the constrained least square problem (Fig. 13 b). The results are represented in terms of poloidal magnetic flux surfaces, relative percentage error map between LS solution and direct equilibrium one, active coil currents distribution of currents (i.e. LS solution) with engineering limits (Fig. 14 a). The results can be saved by clicking save results button (Fig. 14b).

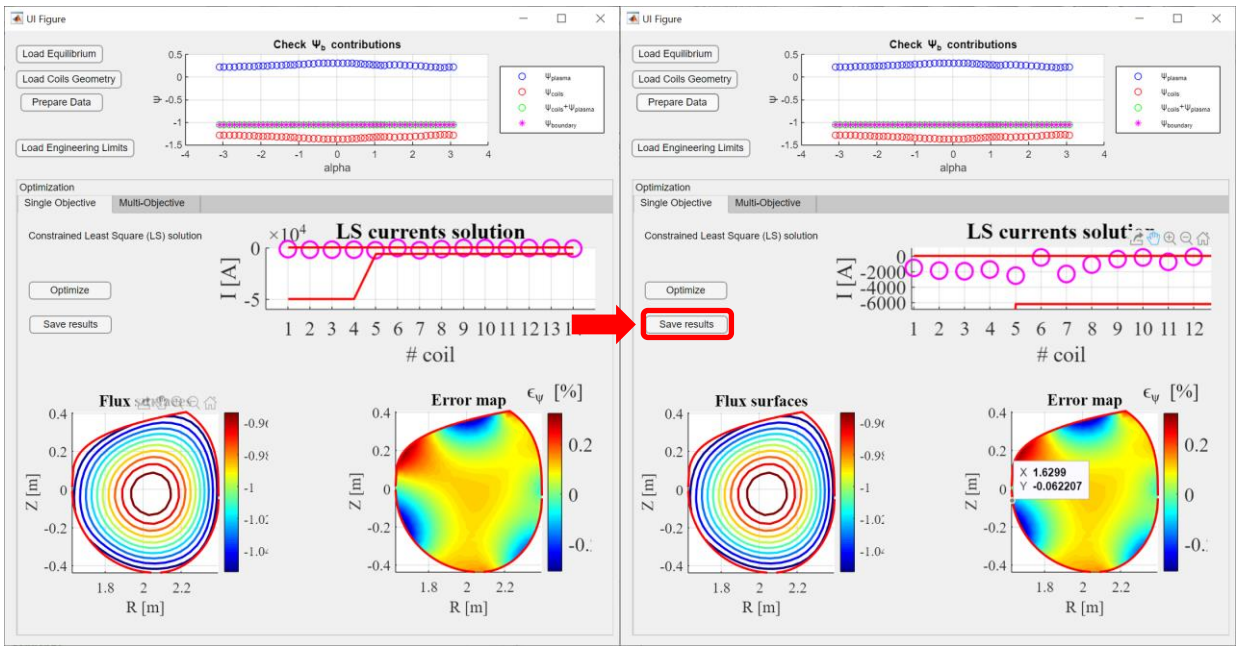

UI Figure
Load Equilibrium
Load Coils Geometry
Prepare Data
Load Engineering Limits
Check Ψ_b contributions
alpha
Optimization
Single Objective
Multi-Objective
Constrained Least Square (LS) solution
Optimize
Save results
LS currents solution
I [A]
coil
Flux surfaces
Error map
ϵ_Ψ [%]
Z [m]
R [m]
X 1.6299
Y -0.062207

Fig. 14

7 EXAMPLES

The example files for RFX-mod and ITER-like test cases are collected in the folder “Examples\RFX_Example” and “Examples\ITER_Example”. Inside each folder there are three folders named: “INPUT_EquilTool”, “INPUT_OptimizationTool” and “INPUT_ShapingTool”. Inside each of these folders there are the input file necessary to each of the IET tool to be tested.

RFX example

- \INPUT_ShapingTool:
  - RFX_USN_withFW.mat : contains the reference boundary with first wall (*geometry* structure in Sect. 4.1)
  - RFX_USN_withoutFW.mat : contains the reference boundary without first wall (*geometry* structure in Sect. 4.1)
- \INPUT_EquilTool:
  - The Jprofile is defined via parametrization (Fig. 9).
- \INPUT_OptimizationTool:
  - RFX_active_coils.mat : contains the geometry of active coils (*coils* structure in Sect. 6.1*)*
  - RFX_eng_limits.mat : contains the engineering limits on currents for each active coil (*EngLimits coils* structure in Sect. 6.1*)*

ITER example

- \INPUT_ShapingTool:
  - INPUT_ITER_shape.mat: contains the reference boundary with first wall
- \INPUT_EquilTool:
  - Psibar.mat: contains the normalized poloidal flux related to the profiles of p and ff’ (*Jprofile* structure in Sect. 5.1*)*
  - Pprime_profile.mat: contains the pressure gradient profile (*Jprofile* structure in Sect. 5.1*)*
  - FFprime_profile.mat: contains the FF’ profile (*Jprofile* structure in Sect. 5.1*)*
  - Jprofile_arbitrary.mat: contains the plasma current density profile by points(*JprofileArbitrary* structure in Sect. 5.1*)*
- \INPUT_OptimizationTool:
  - ITER_geometry_active_coils.mat: contains the geometry of active coils (*coils* structure in Sect. 6.1*)*
  - ITER_eng_limits.mat: contains the engineering limits on currents for each active coil (*EngLimits coils* structure in Sect. 6.1*)*

Each tool needs the input of the previous one as already described in Sect. 3.