Creating ITER Fiducial States

R. H. Bulmer and L. D. Pearlstein

2010-12-28 [Rev: 1.4]

Work performed under ITER Expert Contract IO/2010/ADM-107
Responsible Officer: Thomas Casper, ITER Organization

1 Introduction

Modeling ITER plasma scenarios with Corsica requires the generation of a small set of fiducial-state cases which are used as starting points or benchmarks for time-dependent analyses. Routines defined in the Corsica *fiducial-generator script* can be used to create these fiducial states by importing device configuration specifications from a set of input files and generating the necessary states by applying plasma shape and flux-linkage constraints with appropriate analytic models for plasma profiles.

This document describes routines which create an initial-magnetization state with zero plasma current and four equilibrium states: (1) low-current start-of-plasma, (2) full-current, low-beta start-of-flattop, (3) full-beta start-of-burn, and (4) and an end-of-burn state. These states represent the minimal set needed to characterize a tokamak scenario.

Section **??** gives a brief overview of the procedure. Section **??** introduces device configuration files. Section **??** introduces some of the ITER-specific diagnostic routines which are used by the fiducial generators. Section **??** describes each fiducial state in detail, and Section **??** explains how to access fiducial-state save-files and the associated data-files. Appendix **??** contains a glossary of Corsica-specific terms and Appendix **??** presents certain equilibrium quantities which have multiple definitions. Appendix **??** contains a description of the analytic plasma profile models used in the fiducial-generator script and Appendix **??** describes how flux-linkage (from the CS and PF coils to the plasma) is treated.

2 Overview

In developing fiducial states for a new device configuration, a set of input files must first be prepared containing all of the device specifications. These files are processed by routines in standard script[1] `device.bas`, described in Section **??**. The fiducial-generators standard script[2] reads the device-script and executes the top-level routine (`read_device`) automatically.

When the device-files are present in the current working directory, fiducial states can be created in a two-step (two separate Corsica sessions) process:

1. Create an initial-magnetization state by starting-up *without* a save-file, reading `generate_fiducials.bas` and executing the `make_im` routine.
2. Create all the fiducial equilibrium states by starting-up *with* an old save-file, reading `generate_fiducials.bas` and executing the `make_fiducials` routine.

In the first step we create an initial-magnetization state using default values for the run-time parameters (described in Section **??**) by executing the following:

```
caltrans generate_fiducials.bas
make_im
quit
```

The IM-state generator saves certain parameters in a PDB[3] file which are used by the equilibrium-state generators; therefore, the IM-state generator must be executed first. The `make_im` routine does not produce an equilibrium save-file since it does not represent an equilibrium state—only a static vacuum field condition. An NCGM file will be created which contains plots of the poloidal field, flux and coil parameters at the IM state.

The 2nd step in fiducials creation requires starting-up the code with an old save-file (for a prior device configuration):

```
caltrans old-save-file generate_fiducials.bas
make_fiducials
quit
```

[1] Standard script files are Corsica scripts that are available in all distributions and are located under the directory named by environment variable `CORSICA_SCRIPTS`.

[2] Pathname: `$CORSICA_SCRIPTS/ITER/generate_fiducials.bas`.

[3] See the glossary in Appendix **??** for definitions of Corsica-specific terms.

The `make_fiducials` routine calls individual fiducial-state generators for each fiducial-state and creates save-files for each with names constructed using the lowercase device name as defined in the `params.in` file (e.g., *`device`*`_sof.sav`, etc.). The save-file prefix can be changed by providing a character string argument to the `make_fiducials` routine, which may be blank if no prefix string is desired. An NCGM graphics file will also be created which includes plots of the equilibrium flux with plasma parameters, plasma profile quantities, various shape and position diagnostics and superconducting coil parameters.

There are optional arguments to all of the individual fiducial state generators, which are described in Section **??**.

3 Device Files

Project data describing the poloidal field coils, vacuum vessel, first-wall, etc., must be translated into the set of device configuration files summarized in table **??**. The

Table 1: Device configuration input files

File-Name	*File contents*
`coils.in`	Poloidal field coils and other toroidal current sources
`dgaps.in`	Diagnostic gap specifications
`limits.in`	Coil current, field and force limits
`params.in`	Nominal plasma and device parameters
`passive.in`	Passive structure specifications
`shape.in`	Target plasma boundary coordinates
`tfcoil.in`	Toroidal field coil specifications
`wall.in`	Plasma-facing first-wall, divertor and limiter geometry

preparation and/or modification of device configuration files is described in the documentation for the device-script [**?**].

The fiducial-generator script executes the `read_device` routine so there is no need to invoke any of the device-script routines directly.

4 ITER Script

In addition to loading the device-script, the fiducial generator script reads standard script `iter.bas` into the session. This script defines several ITER-specific routines, three of the diagnostic routines are executed by `make_fiducials` and are described in the following subsections.

4.1 Diagnostic gap evaluation

The `dgapschk` routine evaluates the distance between the last closed flux surface and the first-wall at the diagnostic gap locations as defined in the `dgaps.in` device file. Invoke with:

```
integer n = dgapschk(showplot)
```

where *n* will contain the number of gap violations. The argument is:

showplot Either zero or non-zero to make a plot of the diagnostic gap locations and list the distances to the LCFS for each gap [default: 0].

4.2 Divertor strike-line excursion

The `divchk` routine evaluates the clearance between the separatrix strike-lines and their allowable strike-zones. Invoke with:

```
real v(2,2) = divchk(showplot, make_ref, h_up, h_down)
```

which returns a 2×2 real array where the elements $v_{i,j}$ contain the minimum distances [m] to the inboard ($j = 1$) and outboard ($j = 2$), upper ($i = 1$) and lower ($i = 2$) strike-zone boundaries. The arguments are:

showplot Either zero or non-zero to create a plot of the strike-lines and the allowable strike-zones [default: 0].

make_ref Non-zero to use the present equilibrium as the reference separatrix shape creating the file *device*`_dsz.pfb` for use when `make_ref` is zero [default: 0].

h_up Strike-zone half-width [m] in the "upward" direction [default: 0.15 m].

h_down Strike-zone half-width [m] in the "downward" direction [default: 0.05 m].

4.3 SOL-wall clearance

Finally, the `wallchk` routine evaluates the clearance between the scrape-off-layer (SOL) flux surface and the first-wall. Invoke with:

```
real v(2) = wallchk(showplot, dr_sol, d_star)
```

which returns a real array v_i of length 2 containing the minimum distance between the SOL and the first-wall on the outboard $(i = 1)$ and inboard side $(i = 2)$. The arguments are:

showplot Non-zero to make a plot of the SOL-wall distance as a function of poloidal angle [default: 0].

dr_sol Defines the SOL flux surface as the radial distance ΔR_{SOL} [m] between the separatrix and the outer boundary of the scrape-off-layer as measured from the outboard edge [default: 0.04 m].

d_star Defines the minimum allowable distance between the SOL and the first-wall [default: 0.08 m], used only in graphical output.

5 Creating Fiducial States

Fiducial states based on static vacuum field or equilibrium solutions are defined for a simplified scenario model as depicted in Figure **??**. The scenario is represented as a function of only external flux, Ψ_{ext}, which links the plasma or breakdown region from the poloidal field coil system.

There are presently five fiducial states defined which can be generated by the fiducial-generator script:

IM *initial-magnetization* state: zero plasma current where a field null is created over a specified region and the coil currents are adjusted to produce a desired poloidal flux value, or to maximize it.

SOP *Start-of-plasma* state: low-current circular plasma—a starting point for time-dependent simulations.

SOF *Start-of-flattop* state: full-current, full-bore plasma prior to heating (low beta).

SOB *Start-of-burn* state: full-current, full-bore plasma after heating (full beta).

EOB *End-of-burn* state: full-current, full-bore plasma at full beta with the coil currents adjusted for maximum burn-flux production.

Each of these states requires a self-consistent determination of the poloidal flux linking the plasma from the poloidal field coils, which, along with plasma shape constraints, are satisfied by the Grad-Shafranov solver.

The procedures to create fiducial states all use the Corsica constrained equilibrium package, `ceq`, which solves multiple nonlinear equations using Michael Powell's hybrid algorithm [**?**]. Desired values for plasma parameters (β_p, ℓ_i, q_0) are realized by adjusting profile parameters. Various coil limits are satisfied by adjusting coil currents in `ceq`. These `ceq` problems involve constraint and independent variable specifications with array indexes to identify particular coils.

To make the generators independent of the coil ordering in the `coils.in` limits file, the generators determine the appropriate coil index values internally based

on coil names. Therefore, the first twelve coil names in the coil set *must be*, but in any order, six poloidal field coils (PF1–6) and six central-solenoid coils (CS1U & L, CS2U & L, CS3U & L).

5.1 Initial-magnetization state

The initial-magnetization state represents an initial steady-state condition prior to the plasma formation phase of the scenario. The poloidal flux produced by the PF coils is usually maximized and the poloidal magnetic field in the *breakdown region* is minimized. In Corsica the IM state is created by extrapolating, from two low-current equilibrium solutions, selected quantities at zero plasma current.

The breakdown region is defined as a torus, tangent to the first-wall, with minor radius a_{BD} and located at an elevation of Z_{BD}. The major radius of the breakdown region is determined indirectly by specifying the location of the tangency point of the breakdown region on the first-wall—either "inboard" or "outboard".

The low-current equilibrium solutions are obtained in Corsica by posing two constrained equilibrium problems using the `ceq` package. Each problem uses fuzzy-boundary points and a fixed limiter point to define a circular plasma tangent to the first-wall. The two `ceq` problems only differ in the value of the toroidal plasma current.

The usual flux constraint is to maximize, for positive plasma current directionality[4], the poloidal flux of the system by adjusting the current in each central-solenoid coil to operate at its technological limit. If the plasma current is negative, the poloidal flux will be minimized. In Corsica, this means the superconductor utilization factors for the CS coils (a subset of array `ufc`) will all be at their allowable limiting value of one.

5.1.1 IM generator

The initial-magnetization state is unique in that the code starts up *without* a save-file, using only information extracted from the device-files and a small set of parameters in the `make_im` routine. To create an IM state, execute:

```
caltrans generate_fiducials.bas
make_im
quit
```

[4] The sign of the plasma current is the sign of the *nominal plasma current* for the device, specified by the "current" entry in the `params.in` device-file.

Output files from the IM generator are discussed in Section **??**. Do not create a save-file after executing the IM generator as the final solution will be in an inconsistent state.

The `make_im` routine takes up to six optional arguments:

```
make_im(location, constraint, value, a_BD, z_BD, vs_BD)
```

where the arguments are:

`location` Character string denoting where the edge of the breakdown region is to be located, either `"inboard"` (the default) or `"outboard"`, indicating the edge of the breakdown region is tangent to the first-wall on the inboard or outboard side.

`constraint` Character string denoting the type of constraint to be applied in determining the flux-linkage from the coils, either `"ufcmax"` (the default) or `"vltsnd"`. The `"ufcmax"` constraint indicates that the CS coils are to operate at the specified value of the superconductor utilization factor, thereby producing maximum poloidal flux. The `"vltsnd"` constraint can be used to determine the coil currents for a specified value of flux-linkage.

`value` The desired value for the constraint, either a utilization factor or poloidal flux value [Wb], where the default is 1 indicating the maximum superconductor utilization factor is to be at its upper limit.

`a_BD` Minor radius of the breakdown region, with a default value of 1.6 m.

`z_BD` Axial position of the breakdown region, with a default value of 0.6ṁ.

`vs_BD` Flux consumed during plasma formation, $\Delta\Psi_{BD}$, with a default value of 8 Wb.

The last optional argument, flux consumed during the plasma formation phase, $\Delta\Psi_{BD}$, is provided by an independent analysis [**?**] of the plasma formation phase of the scenario.

5.1.2 Output files

The IM generator writes graphical output to a Corsica NCGM file, which includes Ψ_{ext} and $|B_{pol}|$ contour plots, out-of-plane forces on the TF coils and superconductor utilization factors.

A PDB file will also be created, with the name *`device`*`_im.pfb` containing extrapolated quantities at IM including coil currents and poloidal flux, where *`device`* is the lowercase device name defined in the `params.in` device-file. The PDB file is used by the equilibrium fiducial state generators, described in the next section.

5.2 Equilibrium fiducial states

The presently defined four equilibrium fiducial states are generated with a single routine, `make_fiducials`. Whereas the IM generator required only information in the device-files, the equilibrium states require much more information, which will be inherited from an old save-file.

To create the fiducial equilibrium states, start-up with an old save-file, as follows.

```
caltrans old.sav generate_fiducials.bas
make_fiducials(prefix)
quit
```

where *prefix* is an optional prefix string for naming output files, with names of the form *prefix*`-`*state*`.sav`, where *state* is the lowercase fiducial state acronym. The default *prefix* string is the lowercase representation of the device name as defined in the `params.in` device-file. If *prefix* is an empty or blank string[5], the output files will have names of the form *state*`.sav`.

The `make_fiducials` routine calls individual fiducial-state generator routines which are described in Sections **??** through **??**. Each generator routine accepts optional arguments, so one could write a custom wrapper routine providing argument values to the underlying generators. However, the default values of all arguments can be specified directly by the user, as described in Section **??**, and this is the recommended way to pass argument values.

5.2.1 Start-of-plasma state generator

The start-of-plasma state represents a fiducial state with a low-current, wall-limited, circular plasma, which can be used as a starting point for time-dependent simulations.

The SOP state in Corsica uses the same location ("inboard" or "outboard") as used for the IM state, generating a set of fuzzy boundary points for a circular plasma shape of minor radius a limited by either the inboard or outboard first-wall.

The flux constraint is programmed for

$$\langle \Psi_{ext} \rangle = \Psi_{init} - L_p I_p - \mu_0 C_E I_p R_0$$

[5] An oddity in Basis codes is that specifying a blank string requires the syntax `"␣"` (here, ␣ represents the space character); the conventional expression `""` will result in a syntax error. One may also use the built-in variable `blanks`, e.g., `make_fiducials(blanks)`.

as described in Appendix **??**, where the Ejima coefficient C_E is specified in code variable `cejima` and the initial flux value, $\Psi_{init} = \Psi_{BD} = \Psi_{IM} - \Delta\Psi_{BD}$, in code variable `vltf`.

Simple parabolic profiles are used for pressure and ohmic current (see Appendix **??**):

$$p(x) \propto (1-x)^{a_p}$$
$$J_O(x) \propto (1-x)^{a_J}$$

with $a_p = 2$ and $a_J \approx 1$.

The SOP state is created with the `make_sop` routine, which accepts up to five optional arguments:

```
make_sop(i_p, a_p, c_e, beta_p, l_i)
```

where we use values from a plasma initiation analysis [**?**] at 1.6 s from the initial magnetization state (breakdown occurs at 1.1 s):

i_p Plasma current, I_p, default value 0.433 MA.

a_p Minor radius, a, default value 1.6 m.

c_e Ejima coefficient, default value 0.30.

beta_p Poloidal beta, β_p, default value 0.2.

l_i Internal inductance, ℓ_i, default value 0.85.

5.2.2 Start-of-flattop state generator

The start-of-flattop is the fiducial state at the end of a conventional (ohmically-heated) plasma ramp-up phase. The plasma shape conforms to the reference shape as prescribed in the `shape.in` device-file.

The flux constraint at SOF is identical to that at the SOP state:

$$\Psi_{ext} = \Psi_{init} - L_p I_p - \mu_0 C_E I_p R_0$$

where the Ejima coefficient C_E is held in variable `cejima` and the initial flux, $\Psi_{init} = \Psi_{IM} - \Delta\Psi_{BD}$, in variable `vltf`.

The plasma pressure profile, with `ipp` $= 3$, now includes a small edge-pedestal (see Appendix **??**):

$$p(x) \propto \left(1 - x^{b_p}\right)^{a_p} + \varepsilon_p C_p (1-x) x^{\eta_p}$$

with $a_p = 2$, $b_p = 1$, $\varepsilon_p = 0.02$, $\eta_p = 20$ and $\beta_J \approx 0.05$.

The ohmic current profile (with `ipf` $= 4$) has a parabolic profile form with a region of sawtooth activity:

$$\begin{aligned} J_O(x) &\propto 1 && 0 < x \le x_s \\ J_O(x) &\propto \left[1 - (x - x_s)^{b_J}\right]^{a_J} && x_s < x \le 1 \end{aligned}$$

with $a_J \approx 1$, $b_J = 1$ and $x_s \approx 0.1$.

The SOF state is generated with the `make_sof` routine, which accepts up to four optional arguments:

```
make_sof(c_e, beta_p, l_i, q_0)
```

where the arguments are:

- ***c_e*** Ejima coefficient, default value 0.30.
- ***beta_p*** Poloidal beta, β_p, default value 0.1.
- ***l_i*** Normalized internal inductance, ℓ_i, default value 0.85.
- ***q_0*** Safety factor on axis, q_0, default value 0.95.

The SOF generator solves a constrained equilibrium problem where a_J, β_J and x_s are independent variables varied to achieve the desired values of ℓ_i, β_p and q_0.

Additional shape constraints are also included in the SOF state generator (as well as SOB and EOB state generators). These consist of four "hard boundary" points which are guaranteed to lie on the separatrix surface. Two fixed boundary points are installed which are adjusted by `ceq` to achieve the desired values of the nominal plasma size, R and a. Although these constraints are very weak, they insure the fiducial equilibria will conform exactly to the reference plasma shape. Two other hard points are employed to fix the strike-points at their nominal positions to maintain separatrix conformity in the divertor channels. These coordinates are obtained from the terminal ends of the target separatrix.

5.2.3 Start-of-burn state generator

The start-of-burn state is created in the same way as the SOF state except the initial flux value accounts for the resistive loss during the heating phase and beta is increased to its full value. The pressure and current profiles are also somewhat different.

The initial flux value is now:

$$\Psi_{init} = \Psi_{IM} - \Delta\Psi_{BD} - \Delta\Psi_{heat}$$

The pressure profile (with `ipp` $= 3$) is similar to the SOF state except for the edge-pedestal parameters:

$$p(x) \propto \left(1 - x^{b_p}\right)^{a_p} + \varepsilon_p C_p (1-x) x^{\eta_p}$$

with $a_p = 2$, $b_p = 1$, $\varepsilon_p = 0.07$, $\eta_p = 10$ and $beta_J \approx 0.3$.

The ohmic current profile (with `ipf` $= 4$) now has a significant edge-pedestal:

$$\begin{aligned} J_O(x) &\propto 1 && 0 \le x \le x_s \\ J_O(x) &\propto \left(1 - x^{b_J}\right)^{a_J} + \varepsilon_J C_J (1-x) x^{\eta_J} && x_s < x \le 1 \end{aligned}$$

with $a_J \approx 4$, $b_J = 1$, $\varepsilon_J = 0.25$, $\eta_J = 20$ and $x_s \approx 0.2$.

The SOB state is generated with the `make_sob` routine, which accepts up to four optional arguments:

```
make_sob(vs_heat, beta_p, l_i, q_0)
```

where the arguments are:

vs_heat Resistive flux consumption during burn phase, default value 3 Wb.

beta_p Poloidal beta, β_p, default value 0.7.

l_i Normalized internal inductance, ℓ_i, default value 0.85.

q_0 Safety factor on axis, q_0, default value 0.95.

The SOB generator, like the SOF generator, solves a constrained equilibrium problem where a_J, β_J and x_s are varied to achieve the desired values of ℓ_i, β_p and q_0. It also adjusts two hard boundary points to hold the plasma at its reference size, while holding the strike-points in the divertor channels at their nominal positions.

5.2.4 End-of-burn state generator

The end-of-burn state is unique among the equilibrium fiducial states in that the flux-linkage from the coils, as in the IM state, is determined by coil limits and not by inductive and resistive consumption. We maximize the flux-linkage from the coils by initializing the Corsica flux constraint, `vltf`, to the value of Ψ_{ext} obtained

for the SOB state and set $C_E = 0$. Additionally, the coil circuit index values are modified to operate the CS2U and CS2L coils at the same current as the CS1 pair which is desirable for maximum flux production.

The pressure and current profile models at EOB are the same at SOB.

The EOB state is created with the `make_eob` routine, which accepts up to four optional arguments:

```
make_eob(dsep_min, beta_p, l_i, q_0)
```

where the arguments are:

dsep_min Minimum allowable separatrix separation, ΔR_{sep}, default value 0.04 m.

beta_p Poloidal beta, β_p, default value 0.7.

l_i Normalized internal inductance, ℓ_i, default value 0.85.

q_0 Safety factor on axis, q_0, default value 0.95.

The `ceq` problem has the same constrains as that for the SOB state, varying a_J, β_J and x_s to achieve the desired values of ℓ_i, β_p and q_0, along with the constraints on R_0 and a and constraining the nominal strike-points in the divertor channels.

We also add two additional constraints to the `ceq` problem: the maximum value of the superconductor utilization factor, `max(ufc)` $= 1$, is achieved by varying the flux constraint `vltf`, and the maximum allowable repulsion force on the CS coils, in variable `csfz_rep`, is constrained to its allowable value, obtained from the `limits.in` device-file, by adjusting the current in the CS3U coil.

The constraint on ΔR_{sep} is not performed by `ceq`, but directly by Corsica, as follows. An initial solution with the separation unconstrained is found, then, only if the resulting separation is less than the value specified in argument `dsep_min`, the EOB generator will recompute the solution with the separation constrained by setting selector `irl` to 1 and `rl` to the desired value.

5.2.5 Output files

The `make_fiducials` routine writes graphical output to an NCGM file which includes equilibrium summary plots, plasma profiles, various plasma shape diagnostics, superconductor utilization and overlays of the separatrix and out-of-plane forces on the TF coils.

Four PDB files are are also created, using the (lowercase) device name as specified in the `params.in` file:

***device*`_id.pfb`** containing device informational quantities,
***device*`_dgaps.pfb`** containing diagnostic gap locations and limits,
***device*`_tfc.pfb`** containing toroidal field coil specifications, and
***device*`_dsz.pfb`** containing divertor strike-zone definitions.

The first three PDB files are created directly by the device-script and the last PDB file is created by the fiducials-script, using the SOB solution as the reference case for determining the divertor strike-zones.

5.3 Changing default argument values

The recommended method for passing arguments to the individual initial magnetization and equilibrium fiducial state generators is to redefine the default values of the arguments. Arguments have associated default values which are contained in global variables with names of the form `d_`*`state`*`_`*`arg`*, where *`state`* is the fiducial state acronym and *`arg`* is the argument name as given in the function prototypes listed in the previous sections.

To change the default values for the SOF state and burn phase arguments `beta_p` to 0.05 and 0.5, for example, enter:

```
d_sof_beta_p = 0.05
d_sob_beta_p = 0.5
d_eob_beta_p = d_sob_beta_p
```

Argument default settings can be placed in a text file in the current working directory with the name `fiducial_defaults.bas`; it will be loaded by the fiducial-generator script when it is read.

6 Using Fiducial Files

The equilibrium save-files and associated PDB data files created by the device and fiducial generator scripts comprise a set of "fiducial files" for a particular device configuration. It is desirable to be able to open these files in a Corsica session from any place in a hierarchical file-system by name, without having to qualify each file name with its path.

One way to accomplish this is to place the fiducial files in a single location and add

a Basis search path entry to identify their location. Say, for example, your copy of the fiducial files are placed in directory[6]:

```
$HOME/ITER/v3.4
```

Append this directory name to your Basis search path by executing the statement:

```
pathadd("$HOME/ITER/v3.4")
```

or better yet, place the `pathadd` statement in your `$HOME/.basis` file which will be automatically read into each Corsica session when the code is launched. This path entry, of course, should be updated with each new issue of the configuration.

Refer to the Basis documentation for additional information on path management or other general information.

[6] `v3.4` is a synonym for the 2010-12-14 ITER device configuration.

A Glossary

Definitions of common Corsica acronyms used in this document:

PDB *Portable Database File*, a binary file format used by Corsica which is defined by the Portable Application Code Toolkit (PACT). This file format is used by Corsica for equilibrium save-files (which contain a specific set of equilibrium quantities) and for general purpose usage where arbitrary data can be saved from or restored into a session.

PFB *Portable Files from Basis*. The Basis Code Development System, the framework used by Corsica, provides an interface to the PACT PDB library. PDB files written and read by Basis codes like Corsica, by convention, have a `pfb` file extension.

NCGM *NCAR Computer Graphics Meta-file*. Corsica uses a graphics library from the National Center for Atmospheric Research (NCAR). Graphical output files written from Corsica have an `ncgm` file-name extension. NCAR provides utility routines to view and translate `ncgm` files into other file formats.

B Definitions

The following definitions are functions of the poloidal magnetic field, B_p, and plasma current, I_p, volume, V and pressure, p. The notation $\langle x \rangle$ indicates a volume averaged value of quantity x, and $\langle\langle x \rangle\rangle$ indicates a flux-surface-averaged value of quantity x.

The quantities listed below have multiple definitions in Corsica; those noted with the † symbol are the definitions used by the ITER Organization.

B.1 Poloidal beta

There are three evaluations of *poloidal beta*, β_p, which are returned in Corsica array `betap(1:3)`:

$$\beta^{\dagger}_{p_1} = \frac{2\mu_0 \langle p \rangle}{\langle\langle B_p^2 \rangle\rangle}, \quad \beta_{p_2} = \frac{4V \langle p \rangle}{\mu_0 I_p^2 R_{axis}} \quad \text{and} \quad \beta_{p_3} = \frac{4V \langle p \rangle}{\mu_0 I_p^2 R_0}.$$

B.2 Normalized internal inductance

Similarly, there are three definitions of *normalized internal inductance*, ℓ_i, evaluated and returned in array `li(1:3)`:

$$\ell_{i_1} = \frac{\langle B_p^2 \rangle}{\langle\langle B_p^2 \rangle\rangle}, \quad \ell_{i_2} = \frac{2V \langle B_p^2 \rangle}{(\mu_0 I_p)^2 R_{axis}} \quad \text{and} \quad \ell^{\dagger}_{i_3} = \frac{2V \langle B_p^2 \rangle}{(\mu_0 I_p)^2 R_0}.$$

B.3 External flux

External flux, Ψ_{ext}, is the poloidal flux linking the plasma from the poloidal field coils. It is returned in array `vltsnd(1:3)` which contains values for the following definitions:

1. Plasma-current-weighted external flux, $\langle \Psi_{ext} \rangle$, linking the plasma from the external field coils, `vltsnd(1)`,
2. External flux linking the circular loop defined by the magnetic axis, `vltsnd(2)`, and
3. External flux linking the circular loop defined by the geometric radius at the elevation of the magnetic axis, `vltsnd(3)`.

C Analytic Profile Models

Analytic profile models in Corsica are used to prescribe the $p(\psi)$ and $F(\psi)$ profiles for fiducial state equilibria. These models use normalized poloidal flux:

$$x = \frac{\psi - \psi_{axis}}{\psi_{edge} - \psi_{axis}}$$

as the independent variable. We specify the profile forms by first setting the selector `ipj` to 2 which interprets the $F(\psi)$ profile model as specifying the ohmic current profile, $J_O(\psi)$. The $F(\psi)$ profile is then determined indirectly.

C.1 Parabolic forms

There are two selectors which determine the specific analytic models to be used: `ipp` for pressure and `ipf`, with `ipj`$= 2$, for J_O. These two selectors will set to 3 for simple parabolic forms or perhaps 4 to specify a region undergoing sawtooth oscillations. If `ipp` and/or `ipf` = 4, then the region

$$0 \leq x \leq x_s$$

will be flattened to mimic the effects of sawtooth oscillations, where the sawtooth radius (x_s) is specified in code variable `aap`.

The ohmic current form with `ipf` $= 3$ is:

$$J_O(x) \propto \left(1 - x^{b_J}\right)^{a_J}$$

and the parabolic pressure profile form for `ipp`$= 3$ ia:

$$p(x) \propto \left(1 - x^{b_p}\right)^{a_p}$$

where the exponents are specified by the user.

To introduce a sawtooth region, set `ipf` $= 4$ and the ohmic current model is:

$$\begin{aligned} J_O(x) &\propto 1 && 0 < x \leq x_s \\ J_O(x) &\propto \left[1-(x-x_s)^{b_J}\right]^{a_J} && x_s < x \leq 1 \end{aligned}$$

and similarly, with `ipp`$= 4$, the pressure profile is:

$$\begin{aligned} p(x) &\propto 1 && 0 < x \leq x_s \\ p(x) &\propto \left[1-(x-x_s)^{b_p}\right]^{a_p} && x_s < x \leq 1 \end{aligned}$$

Similar superposition may be applied to the sawtooth forms of these profile models.

C.2 Edge-pedestal terms

An edge-pedestal may be superimposed on the parabolic forms by specifying nonzero values of additional parameters ε and η as follows. With `ipf` $= 3$, the ohmic current form is:

$$J_O(x) \propto \left(1-x^{b_J}\right)^{a_J} + \varepsilon_J C_J\,(1-x)\,x^{\eta_J}$$

where

$$C_J = \frac{(1+\eta_J)^{1+\eta_J}}{\eta_J^{\eta_J}}$$

and the pressure profile form with `ipp` $= 3$ is:

$$p(x) \propto \left(1-x^{b_p}\right)^{a_p} + \varepsilon_p C_p\,(1-x)\,x^{\eta_p}$$

where

$$C_p = \frac{(1+\eta_p)^{1+\eta_p}}{\eta_p^{\eta_p}}$$

C.3 Profile summary

The Grad-Shafranov solver (with `ipscl` set to 0) will scale the solution to produce the specified value of toroidal current, I_p, and parameter β_J determines how much of the toroidal current is produced by the pressure gradient.

Table **??** summarizes the relevant parameters for the analytic profile models. Note

Table 2: Analytic profile input parameters

Quantity	*Symbol*	*Variable*	*Value*
I_p scale selector	—	`ipscl`	0
J_O profile selector	—	`ipj`	2
Sawtooth extent	x_s	`aap`	$0 \leq x_s \ll 1$
F profile form selector	—	`ipf`	3 or 4
Outer $J_O(x)$ exponent	a_J	`alfa(0)`	≥ 1
Inner $J_O(x)$ exponent	b_J	`betp(0)`	≥ 1
$J_O(x)$ edge-pedestal amplitude	ε_J	`epf`	≥ 0
$J_O(x)$ edge-pedestal exponent	η_J	`npf`	≤ 0
Pressure scale factor	β_J	`betaj`	> 0
$p(x)$ profile form selector	—	`ipp`	3 or 4
Outer $p(x)$ exponent	a_p	`alfa(1)`	≥ 1
Inner $p(x)$ exponent	b_p	`betp(1)`	≥ 1
$p(x)$ edge-pedestal amplitude	ε_p	`epp`	> 0
$p(x)$ edge-pedestal exponent	η_p	`npp`	≤ 0

that both `npf` and `npp`, if non-zero, must be *negative real numbers*[7] on input. Positive values for these exponents invoke a different profile form—having *finite* p and F (or J_O) values at the edge—not-compatible with the Corsica Grad-Shafranov solver which assumes zero pressure and current on the open field-lines.

D Flux Constraints

The flux linking the plasma from the poloidal field coils can be specified in several ways in Corsica, or not specified at all. Of course there must be enough degrees of freedom (more unconstrained coil currents than shape constraints) in order to accommodate a constraint on flux-linkage.

D.1 Indirect flux constraint

In representing the fiducial states of a scenario, the most useful way to specify the flux is indirectly, by computing the inductive and resistive flux consumption relative to an *initiation point* for the scenario. In this method we know the inductive consumption since we know the plasma current distribution:

$$\Delta\Psi_{ind} = L_p I_p$$

The resistive consumption must be estimated and in Corsica we use Ejima scaling where the resistive flux consumption is found with

$$\Delta\Psi_{res} = \mu_0 C_E I_p R_0$$

[7] Internally, absolute values of `npf` and `npp` are used to evaluate the edge-pedestal profile forms.

and C_E is the Ejima coefficient. We must also be given flux losses during the plasma formation, or breakdown phase, $\Delta\Psi_{BD}$, and during the heating phase, $\Delta\Psi_{heat}$, from independent analyses.

We can then specify the current-weighted flux-linkage, $\langle\Psi_{ext}\rangle$, at any fiducial state by setting an *initial* flux-linkage, Ψ_{init}, in variable `vltf` and the Ejima coefficient, in variable `cejima`.

$$\Psi_{init} = \Psi_{BD} = \Psi_{IM} - \Delta\Psi_{BD}$$

At any point during the current ramp-up phase the flux-linkage is

$$\langle\Psi_{ext}\rangle = \Psi_{init} - \Delta\Psi_{ind} - \Delta\Psi_{res} \quad \text{or}$$
$$\langle\Psi_{ext}\rangle = \Psi_{init} - L_p I_p - \mu_0 C_E I_p R_0$$

At the end of the heating phase we must change our definition for Ψ_{init} by the resistive flux consumed during heating, so `vltf` becomes:

$$\Psi_{init} = \Psi_{IM} - \Delta\Psi_{BD} - \Delta\Psi_{heat}$$

Summarizing, in Corsica we set the variable `vltf` to an initial flux value (either Ψ_{BD} or $\Psi_{BD} - \Delta\Psi_{heat}$) and variable `cejima` to the Ejima coefficient and coil currents will be adjusted to yield the self-consistent value of flux-linkage, $\langle\Psi_{ext}\rangle$, returned in variable `vltsnd(1)`.

D.2 Direct flux constraint

In some cases we may want to constrain the flux-linkage directly, as when we generate an initial-magnetization state or an end-of-burn state. In these two states, we seek solutions where the flux-linkage is usually either maximized or minimized, subject to coil limitations. In these applications we vary $\langle\Psi_{ext}\rangle$ until coil limits are reached. In Corsica, this is accomplished by setting C_E to zero, then the code interprets `vltf` as $\langle\Psi_{ext}\rangle$ instead of Ψ_{init} as the desired value of the flux-linkage.

D.3 No flux constraint

Finally, if we do not want any constraint on flux-linkage in Corsica, we set Ψ_{init} (`vltf`) and C_E (`cejima`) to zero.

References

[1] R. H. Bulmer and L. D. Pearlstein, Managing Device Configurations in Corsica, 2010-12-23 [Rev: 1.2] (Unpublished).

[2] M. J. D. Powell, "A Hybrid Method for Nonlinear Equations", In *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, Editor (Gordon and Breach, 1970).

[3] A. A. Kavin, K. M. Lobanov and A. B. Mineev, Final Report, *Study of plasma initiation using TRANSMAK code: ITER design "Baseline 2010"*, 23 September 2010

Figure 1: Scenario model for the ITER fiducial state generator, shown with positive Ψ_{IM} and positive plasma current directionality (external flux decreasing to the right). Plasma initiation occurs at $\Psi_{BD} = \Psi_{IM} - \Delta\Psi_{BD}$ where the value of $\Delta\Psi_{BD}$ is provided by an independent plasma initiation analysis (as well as the resistive flux consumed during the heating phase, $\Delta\Psi_{heat}$). Resistive flux consumption during the current ramp-up must also be provided by an independent analysis and expressed in terms of an Ejima coefficient.