Use CORSICA to Generate MHD Equilibrium and how to do it in OMFIT

BOUT++
Boundary Plasma Turbulence Code

Guoqiang Li [1,2]

[1]Institute of Plasma Physics, Chinese Academy of Sciences, Hefei, China.

[2]Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

Mini workshop on BOUT++ and OMFIT

Oct. 7 2014, Livermore

Fusion Energy Sciences

Introduction

- Equilibrium is an important basis for BOUT++ simulation
- BOUT++ accepts three kinks of equilibrium: g-file, t-file and .eqin file. It has some IDL/python tools to convert those equilibrium files to grid-file
- CORSICA can generate and modify g-file and t-file
- In OMFIT, a simple CALTRANS module has been developed. It actually wraps the BASIS scripts

What is CORSICA

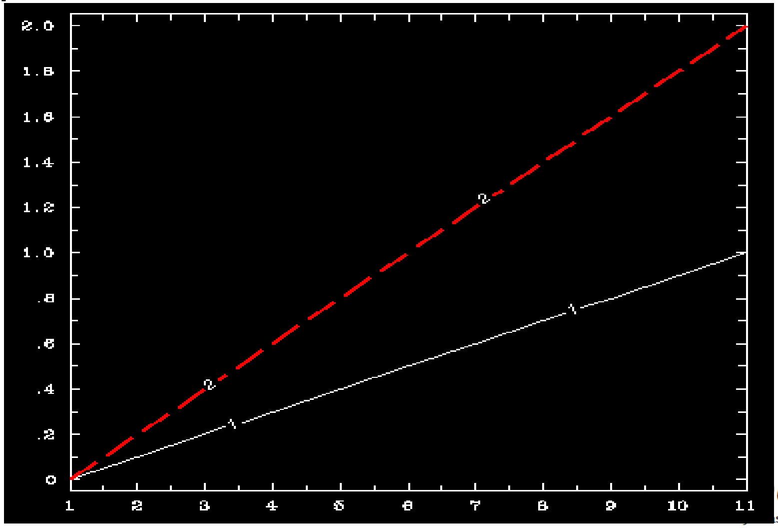
- CORSICA is an integrated modeling tool for equilibrium, MHD stability, transport/heating/CD
- Developed at LLNL
- Corsica runs on GA clusters and hopper on nersc
- http://wormhole.ucllnl.org/caltrans/: download and documentation
- This lecture will only talk about equilibrium part of CORSICA

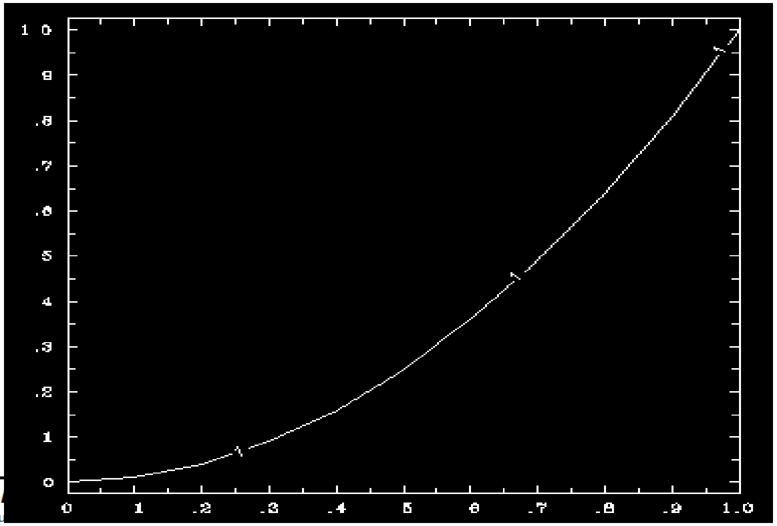
What is BASIS

- BASIS is an interactive script language, like python, IDL, Matlab …
- CORSICA is wrapped by BASIS https://wci.llnl.gov/codes/basis/
- Expressions are similar to FORTRAN
- Declare variables
  - Variables must be declared before using
  - integer, real, double, character, logical
- ‘list’ is very useful to get help information on names
- Has vector operations like F90
- Build-in plot functions and math functions

Exercise for BASIS

```
$ caltrans
> real xx = (0:1.:11)     # declare xx and set vaules
> xx
> list xx
> real yy=xx**2           # vector operation
> yy
> win                     # start a window frame
> plot xx
> plot xx*2., color=red, thick=5, style=dashed  # overlay the xx**2
> nf                      # New framework
> plot yy,xx              # plot yy vs xx
> quit
```



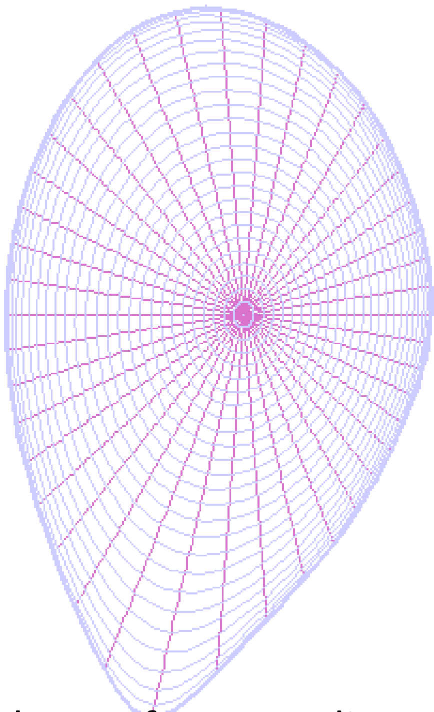


Equilibrium solver in CORSICA

- Grad-Shafranov equation solver, ψ poloidal flux

$$\Delta^*\psi = -\mu_0 R J_\phi \ , \quad J_\phi = RP'(\psi) + \frac{\mu_0 FF'(\psi)}{4\pi^2 R}$$

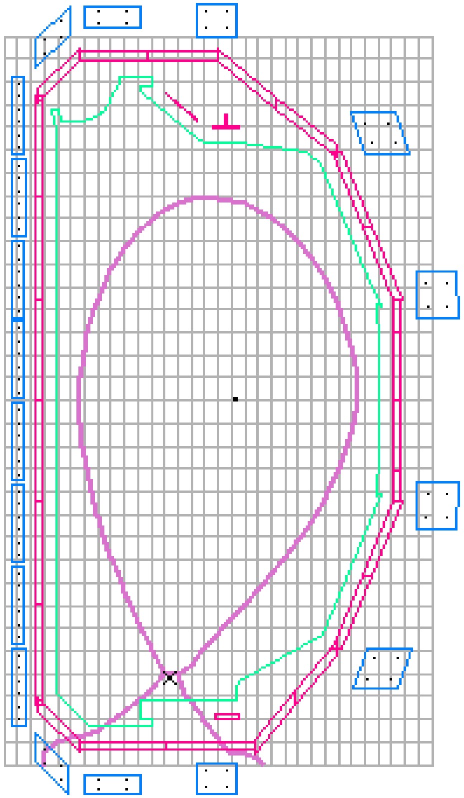
- CORSICA has both direct and inverse solver
  - **Inverse solver**: (ψ, θ) coordinate, solve for R, Z
    - prescribed-boundary: the Grad-Shafranov eq'n is solved inside a region specified by two arrays (input) and the R,Z points around the boundary



Flux surface coordinate

Equilibrium solver in CORSICA (Cont.)

- **Direct solver**: (R, z) coordinate, solve for ψ
  - free boundary: the separatrix (or limiter plasma/vacuum boundary) is found as part of the solution (i.e., output); but there are input parameters to exert substantial control over the boundary.
  - Use pprime and Fprime to evaluate Jphi on R Z grid
  - run inverse solver to get J, then interpolate to R, Z
  - Solve on infinite domain with coils

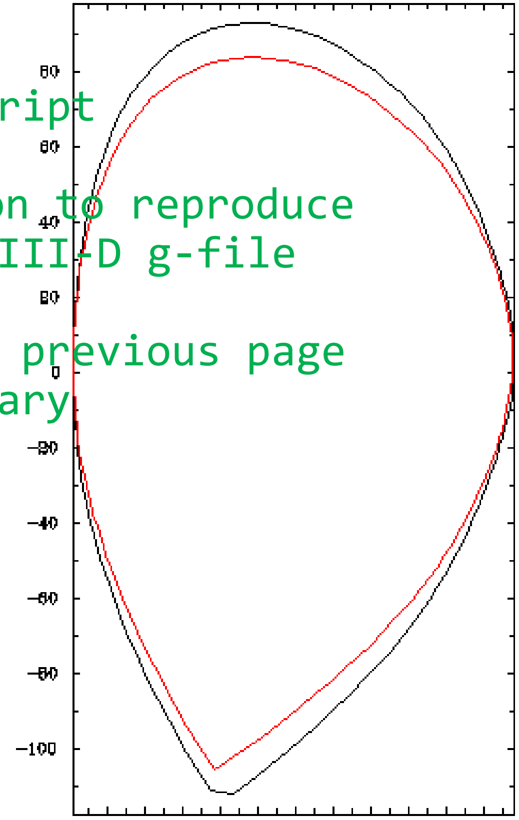


(R,z) coordinate

Change the plasma shape

- Do this work with CORSICA direct solver
- First read a g-file equilibrium, then change the shape
  - g-file is a community standard equilibrium file format, originally from EFIT

```
$ cd corsica-ex
$ caltrans
> read d3.bas         # read in the predefined d3 sript
> d3("g098128.02500",0) # read in the g-file
                        # d3 is an predefined function to reproduce
                             direct equilibrium from DIII-D g-file
> win on
> layout(0,0)         # generate a plot like one on previous page
> plot zls,rls,scale=equal    # plot actual boundary
> zfbd = zfbd*0.9     # modify requested z boundary
> run                 # execute direct solver
> plot zls,rls,color=red
> weqdsk("g")         # write out the new g-file
```



Create a circular plasma from "dead start"

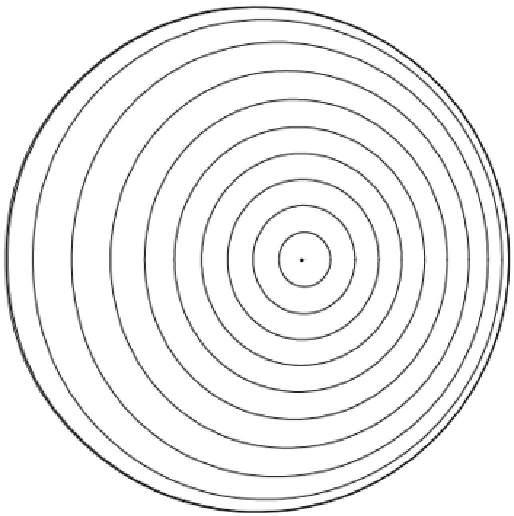
- "Dead start": create an equilibrium from a small set of parameters given in a text file.

```
$ caltrans
> read tokamak.bas            # read in predefined script
> ds("circ.inp")              # dead start with circ.inp file
> win
> layout(0,0)
> start_inv                   # convert to inverse equilibrium
> saveq("circ_inv.sav")       # save the inverse equilibrium to a file
```

```
"circ"

Plasma...
      1.00 MA       plasma current
      2.00 m        major radius
      0.50 m        minor radius
      0.00 m        Zaxis
      1.00          95% elongation
      0.00          95% triangularity
      0.00 m        Dsep (DN)
      0.80          poloidal beta
      0.90          li
      0.00 Wb       External flux linkage

Toroidal field...
2.00 T @ R = 2.00 m
```



UT++
Plasma Turbulence Code

Change the profiles

- Two profiles(P, [FF', q, Jpar, Jt]) are required to determine an equilibrium
- teq_inv(inv_k, inv_p): # command to run inverse solver, inv_k: select which two 1d arrays, inv_p select which scalar sets scale

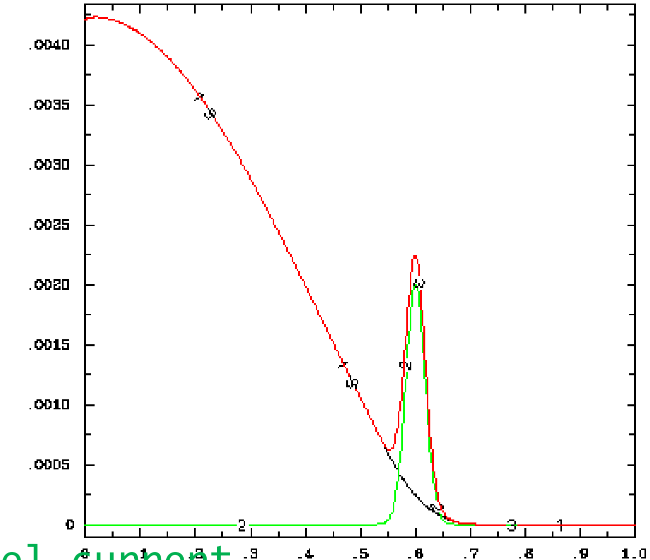
Table 1: teq_inv constrained profile options via inv_k.

inv_k	*constrains...*
0	$p \mapsto$ psave and $q \mapsto$ qsave
1	$p \mapsto$ psave and $FF' \mapsto$ frsrf*fpsrf
2	$p \mapsto$ psave and $\langle \mathbf{J} \cdot \mathbf{B} \rangle / \langle F/R^2 \rangle \mapsto$ jtsave
3	$p \mapsto$ psave and $\langle \mathbf{J} \cdot \mathbf{B} \rangle / \langle B^2 \rangle \mapsto$ jparsave

Table 2: teq_inv constraints $\Delta\psi_p$, F_{edge} or I_p via inv_p

inv_p	inv_k=0,3	inv_k=1,2
		$F^2_{edge} = F^2_{wall}$
$<$ 0	$\Delta\psi_p$	$\Delta\psi_p$ and profiles scaled
$=$ 0	F_{edge}	profiles not scaled
$>$ 0	I_p	I_p and profiles scaled

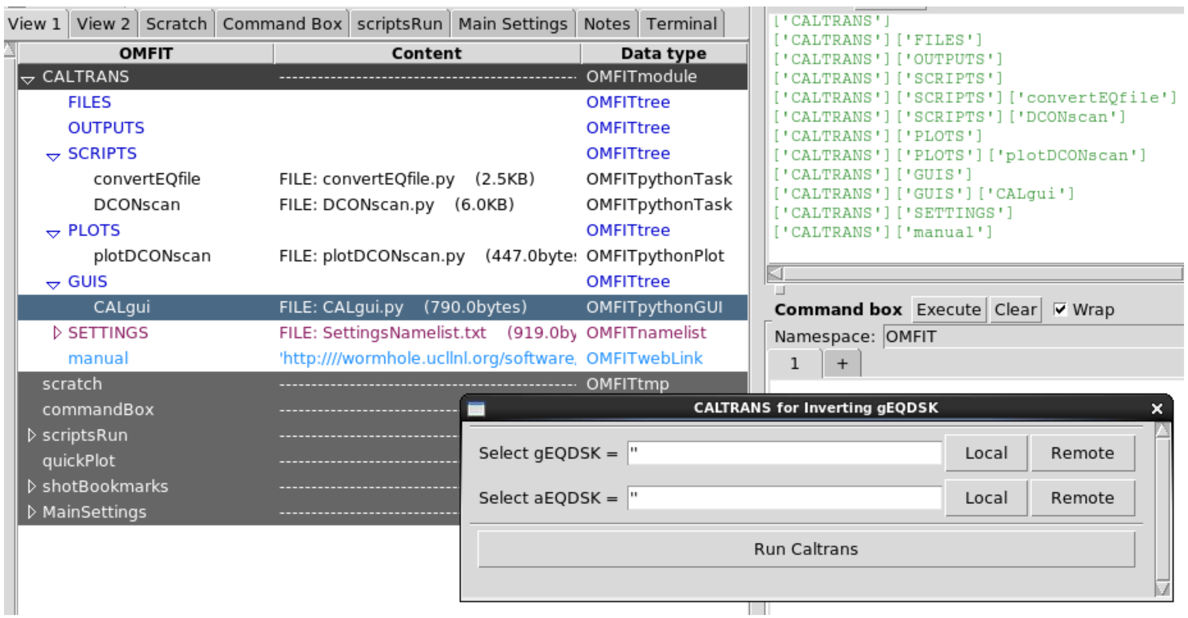
Exercise of changing profiles




```
$ caltrans cbm18_dens8_inv.sav  # start CORSICA and restore an equilibrium
> win on
> plot psave, psibar        # plot pressure vs psi
> psave = psave * 1.2
> teq_inv(0,0)              # excute inverse solver
> plot psave, psibar, color=red
> nf
> plot jparsave, psibar
> real jedge = 0.002*exp(-((psibar-0.6)/0.025)**2)
                            # define an edge current
> plot jedge, psibar, color=green
> jparsave = jparsave + jedge  # change the parallel current
> teq_inv(3,0)
> plot jparsave, psibar, color=red
> shotName = “10000”
> shotTime = 1.0
> weqdsk(“t”)               # Write the equilibrium to “t-file” (dskgato file)
```

Simple CALTRANS module in OMFIT has been developed (by Meneghini)

- At present, CALTRANS in OMFIT has two functions
  - Read in g-file and convert it to new g-file, t-file, i-file and .sav files
  - Scan a sequence of equilibria by scaling the pressure profile and running DCON to check the stability boundary (beta limit)



Some considerations

- CORSICA is wrapped with BASIS. If we use python to wrap the BASIS, it will lose some convenience and functions.
- But BOUT++ does not need so many functions
- What BOUT++ needs in OMFIT:
  - Refine and modify the equilibrium (increase the grid size, change the shape and profiles)
  - Visualize and compare the modified equilibria (develop a tool to visualize the t-file or grid-file), give some equilibrium global parameters
  - Convert the equilibrium to grid-file

Thank you

BOUT++
Boundary Plasma Turbulence Code