Tokamak PF Design with Caltrans

R. H. Bulmer, LLNL
bulmer@llnl.gov

September 2005

Outline

- Introduction to Basis/Caltrans
  - Documentation for Basis, Caltrans (off-line and self-contained)
  - User interaction
- DIII-D and NSTX applications
  - Importing via EQDSK, modifying equilibria (profiles, shape, resolution), saving information.
- Constrained equilibrium analyses
- Dead-starting new configurations

Documentation

- Basis
  - http://basis.llnl.gov
  - Read tutorial, language reference and EZN graphics documents
- Caltrans
  - http://www.mfescience.org or http://web.gat.com/caltrans

University of California
Lawrence Livermore National Laboratory
U.S. Department of Energy

Basis Home Page
Documentation
Downloads
Support
More About Basis

Basis Documentation

Getting Started

There are five Basis documents.

The tutorial "Running a Basis Program" is the right place to start if you are going to be running a program written with Basis.

The authoring manual "Writing Basis Programs" is the right place to start if you are going to be writing Fortran packages to connect to an existing Basis code, or a new code.

Basis Manual Set (UCRL-MA-118543, Parts 1-5)

Please read the instructions below on how to view or download these documents.

1. Running a Basis Program
2. Basis Language Reference Manual
3. EZN User Document: The Basis Graphics Package
4. Writing Basis Programs: A Manual for Authors
5. The Basis Package Library

www.mfescience.org (Caltrans button)

- **Documentation for Basis**
  - The BASIS home page
- **Documentation for Corsica**
  1. Corsica Users' Manual - DRAFT-
  2. Corsica Users' Manual Appendix
  3. Corsica:A Comprehensive Simulation of Toroidal Magnetic-Fusion Devices Final Report (UCRL-ID-126284)
  4. DIII-D Demonstration
  5. U-file Data Collector
  6. Modelling SSPX Equilibria with Corsica
  7. Creating Tokamak Equilibria with Corsica

Caltrans Help

```
caltrans -help
setting CPU_ environment to your value, SOL
Running /mfe/theory/Caltrans/vcaltrans/bin/SOL/caltrans
Usage: caltrans [[option] [filename]]*
Options:
  -help:            Prints this message.
  -probname string  Set 'probname' variable to string. If this option is not
                        specified, 'probname' is set using the name of first input
                        file. If no input files are specified, 'probname'='problem'.

<lines deleted>

Up to 10 filenames can be specified. If the filename ends with the suffix 'sav',
caltrans will assume the file has been saved with saveq/saveqtr and will attempt
to restore the equilibrium/transport case. If the suffix is 'dat' or 'pfb',
caltrans will 'restore filename'. Otherwise, the file is assumed to be a text
file and caltrans will execute 'read filename'. Files are read In the order
specified on the command line.
```

Caltrans Help (cont.)

```
caltrans
corsica> help      # displays Basis' help message in
                   # separate window, describing
                   # "version", "news" and "list"
                   # commands.
corsica> list
list                [print the list options]
list par.Attributes
list [pkg.]functions
list Groupname
list [pkg.]groups
list idname
list macros
list packages
list [pkg.]variables
```

Caltrans Help (Cont.)

```
corsica> list packages
Priority Name             Long Name                    Status
       3 par        Basis System                       -- up --
      36 eq         eq:  calculate an equilibrium      -- up --
      28 ceq        ceq: constrained equilibria via    -- up --
      27 meq        meq: minimize eq with constraint   -- up --
      30 eqt        eqt: equilibrium temporaries       -- up --
      29 vpf        vpf: vertical stability            -- up --
      32 ctr        ctr: core transport                -- up --
      33 nbi        nbi: NB heating & CD               -- up --
      31 csc        csc: inter-package communication   -- up --
      22 mth        mth: mathematical routines         -- up --
      21 anl        Transport Analysis                 -- up --
      20 uec        Core-edge coupling package         -- up --
      19 rfl        rfl: reflectometry diagnostics     -- up --
      18 dcn        dcn: MHD global stability          -- up --
      17 o12        ONETWO 1 & 1/2 D transport         -- up --
      16 x12        for ONETWO var names also in o12   -- up --
      15 mds        mds: MDSPlus access routines       -- up --
      14 ctl        Run Controller Package             -- up --
      13 svd        Singular Value Decomposition       -- up --
      12 fit        Polynomial Fitting                 -- up --
      11 hst        History Package                    -- up --
      10 pfb        PFB Interface                      -- up --
       9 ezc        EZCURVE/NCAR Graphics              -- up --
       8 ezd        Device Package                     -- up --
       7 iso        3-D Isosurface Plotting Routine    -- up --
       6 srf        3-D Surface Plotting Routine       -- up --
       5 bes        Bessel builtins                    -- up --
       4 fft        Fast Fourier Transforms            -- up --
```

Caltrans Help (cont.)

Example queries of package eq (primary equilibrium package)…

```
corsica> list eq.groups          # Display group names
corsica> list eq.functions       # Display user-callable
                                 #   routines
corsica> list eq.variables       # Display user-
                                 #   accessible variables
corsica> list cc                 # Display variable
                                 #   descriptor for a
                                 #   particular variable
corsica> list eq.cc              # Ditto, with qualifier
```

Caltrans Start-up Variants

Start from previous equilibrium

```
caltrans [-probname pname] iter.sav
```

Load from EQDSK

```
caltrans [-probname pname] d3.bas
d3("g106795.01725")
```

Dead-start

```
caltrans [-probname pname] tokamak.bas
ds("tokamak.inp")
```

Scripts and Paths

Caltrans consists of Fortran and C++ compiled modules, along with *script* modules. Variable `scripts` contains the pathnames of standard Caltrans script modules which have been loaded.

Personal scripts will be read from your current working directory, but it is often convenient to put them in a common place, say: `$HOME/Caltrans/scripts/`. Inform Basis about this location with:

```
pathadd("$HOME/Caltrans/scripts")
```

Variable `path` contains the current list of directories in which Basis will look for files. By default, `pathadd` will add its directories to the top of this search-path.

Start-up Scripts

Caltrans will, by default, automatically read two private script files on start-up: `$HOME/.basis` and `$HOME/.caltrans`, (also, `$HOME/.corsica`).

If you run other Basis codes in addition to Caltrans, put common customizations in the `.basis` file, then put your Caltrans customizations in the `.caltrans` file. An obvious entry in the start-up file is something like:

```
pathadd("$HOME/Caltrans/scripts")
```

See the Basis documentation for other search path related routines.

Session Termination

- Type `quit` or `end` (or `quit(1)` to signal an error exit to a controlling process).
- Session log file: *`pname`*`.log`
- Graphics: *`pname.nnn`*`.ncgm`
- Error messages: *`pname.nnn`*`.err`

User Interaction

- Basis provides a Fortran 90-like user interface to Caltrans.
- Commands may be entered interactively or via a script file, or a combination of both.
- Variables may be created on-the-fly to, for example, collect output.
- Basis routines provide both text and binary I/O facilities.

Variable Declarations

- Variable types: real, double, integer, character, complex, logical and chameleon
- Some examples…
  - `integer n, m(5)`
  - `real some_data=[1.2, 3.8, 7]`
  - `character*3=[“abc”,”def”]`
  - `chameleon s=“some string”`
  - `chameleon a=[1,2,3,4,5,6,7]`

Text I/O

Read a file containing a table of time and current…

```
integer n=1000
real buf(2,n)
integer io=basopen(“data.dat”,”r”)
io >> buf
call basclose(io)
real time(n)=buf(1,)
real current(n)=buf(2,)
```

Basis stream input operator: “>>” and stream output operator: “<<“.

Text I/O (cont.)

Write a file containing a table of time and current…

```
character*1 tab=char(9)
integer io=basopen("data.dat","w")
do $i=1,n
    io << format(time($i),10,3,2) \
    << tab \
    << format(current($i),16,8,2)
enddo
call basclose(io)
```

Basis built-in chameleon variables: $a,$b,…$z

Binary I/O (PFB)

Write a file containing a table of time and current…

```
integer n=10000
real time(0), current(0)
do $i=1,n
    <change one or more parameters>
    run # Execute the G-S solver
    time := shotTime
    current := placur
enddo
```

PFB: Portable-Files-from-Basis are easy to create…

```
create data.pfb
write time, current
close
```

Binary I/O (cont.)

Read the file containing time and current data.

PFB files are even easier to read:

```
restore data.pfb
win # Open NCAR X-window
plot current,time
```

To see contents of a PFB file:

```
open data.pfb
ls
```

See PFB package in Basis Package Library doc.

Handy Features

User interaction at the command-line is interpreted by the GNU readline facility (a la Emacs)

Interrupt `caltrans` with CTRL-C; resume (from debug state) with “cont”

Execute system commands…

```
!ls -l
if (basisexe(“ls foo”) <> 0) then
    << “No file foo”
endif
```

Graphics Routines

Caltrans is installed with some pre-defined graphics routines, with optional ones in the script file "graphics.bas"

```
caltrans iter.sav
win; layout; profiles; pb

caltrans iter.sav graphics.bas
graphics # To display help message
win; layout; profiles;pb; …etc…
```

Basis' EZN routines may be used to write your own routines for graphical output.

Importing via EQDSK Files

Capability available for DIII-D, C-Mod and NSTX

```
caltrans d3.bas
d3(“help”)
d3(g-file_name)

caltrans nstx.bas
nstx(“help”)
nstx(g-file_name)
```

EQDSK a-files are read automatically if they are present

Changing Grid Resolution

```
caltrans d3.bas
d3(“g106535.02110”)
gridup # To double resolution
run # Execute the G-S solver
saveq(“some_name.sav”) # Save to disk

caltrans some_name.sav
griddown # To halve resolution
```

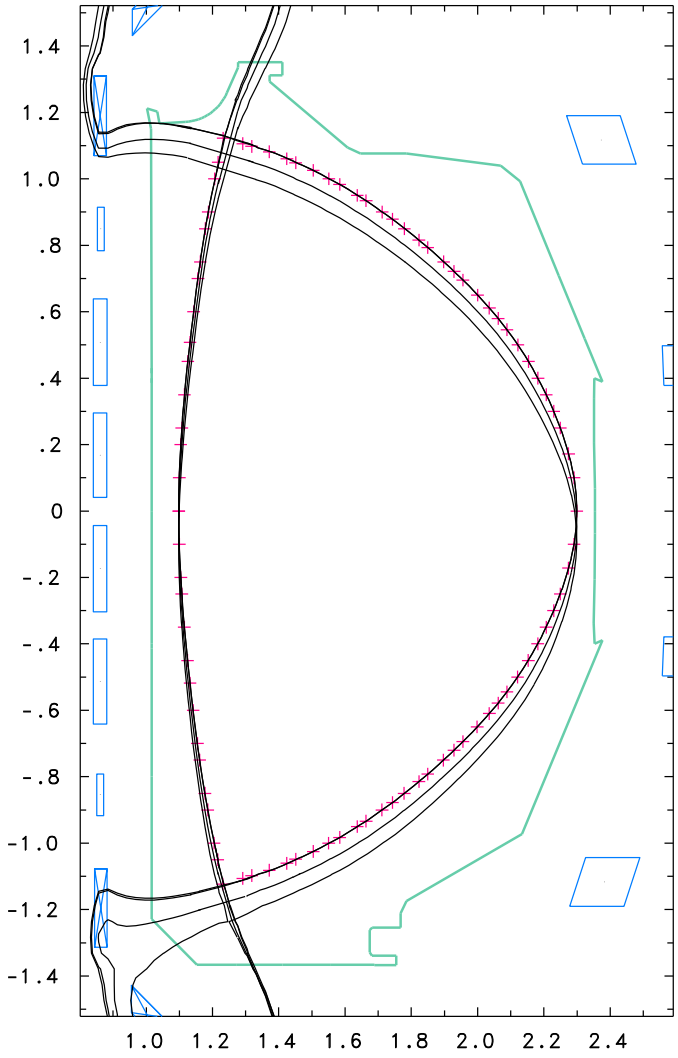
Exporting via EQDSK

```
caltrans d3.bas
d3(“g106535.02110”)
gridup # To double resolution
run # Execute the G-S solver
weqdsk(“help”)
weqdsk(“g”,”_hi_res”)
```

Changing an Equilibrium

Shifting the plasma vertically when its boundary is controlled by “fuzzy” markers:

```
zfbd=zfbd-5; run; pb
zfbd=zfbd-5; run; pb
```



Equilibrium Constraints

- **Shape**
  - Hard markers: `rbd(i),zbd(i), i=1,nbd`
  - Fuzzy markers: `rfbd,zfbd,alfbd(i), i=1,nfbd`
- **Flux**
  - External flux linkage: `vltf` and `cejima`
  - Separatrix separation: `irl`, `rl`
  - Flux at coils: `pfps0(i), i=1,nc`
- **Coil currents**
  - Currents (MA-t): `cc(1:nc)`
  - Circuit indexes: `ic(1:nc)`
  - Regularization schemes: `ircwt` selector
  - Fixed current: `ic(`k`)=0`
  - Connect circuits: `ic(`k_2`)=ic(`k_1`)`
  - Complex constraints via `ceq` package

Changing Profiles

The script routine “eprofiles” (experimental profiles) is available to map arbitrary *F* and *p* profiles to the Caltrans flux surface arrays:

```
call eprofiles(x_exp,f_exp,p_exp,delta_psi)
```

Where `x_exp` is the vector defining normalized Ψ, `f_exp` and `p_exp` are the arrays that contain the desired values. The `delta_psi` argument is the plasma flux: $(\Psi_{edge} - \Psi_{axis})/2\pi$.

There are also several analytic profile models (do: “`list ipp`”).

Inverse Solver

- The inverse solver in Caltrans (derived from the POLAR1 code) may also be used to change profiles and shape.
- See Section 6 (“Altering profiles and shape”) in the document “DIII-D Equilibrium and Stability Modeling with Caltrans“

Constrained Equilibrium Solver

- Package `ceq` provides an interface to Michael Powell's non-linear solver, HYBRID
- User defines:
  - constraint variable names or expressions (`vo`)
  - target values (`vo0`)
  - independent variable names (`vi`)
  - initial values (`x0`)
- For an example, do:
  - `caltrans iter.sav`

Constrained Equil. Solver (cont.)

Example: vary three profile parameters and one shape constraint to control the internal inductance, poloidal beta, peak/ave. pressure and X-point radius:

```
package ceq
nctot=4 # No. of constraints
vo=[“li(3)”,”betap(1)”,”prsrf(1)/presva”,”rxpt”]
vo0=[0.9,0.7,3,124]
vi=[“betp(0)”,”betaj”,”alfa(1)”,”rbd(1)”]
x0=[betp(0),betaj,alfa(1),rbd(1)]
ihy=40 # Iteration limit
run # Run the HYBRID solver
```

Caltrans Variable Names

- Numerous variable names in `caltrans`
- Use the Basis `list` facility to become familiar with them:
  - `list eq.groups # ~40 groups`
  - `list Srcx # Profile parameter group`
  - `list alfa, betp`

Dead Start Procedure

There are several save-files available with each Caltrans installation. To get a current list of them, use the Basis `getenv` routine:

```
call basisexe("ls -l "// \
trim(getenv("CORSICA_PFB"))//"*/*.sav")
```

Tokamak save-files are: `cmod.sav`, `d3d.sav`, `fire.sav`, `ignitor.sav`, `iter.sav`, `kstar.sav`, `nstx.sav`

Only use the dead start procedure when a save-file is not available.

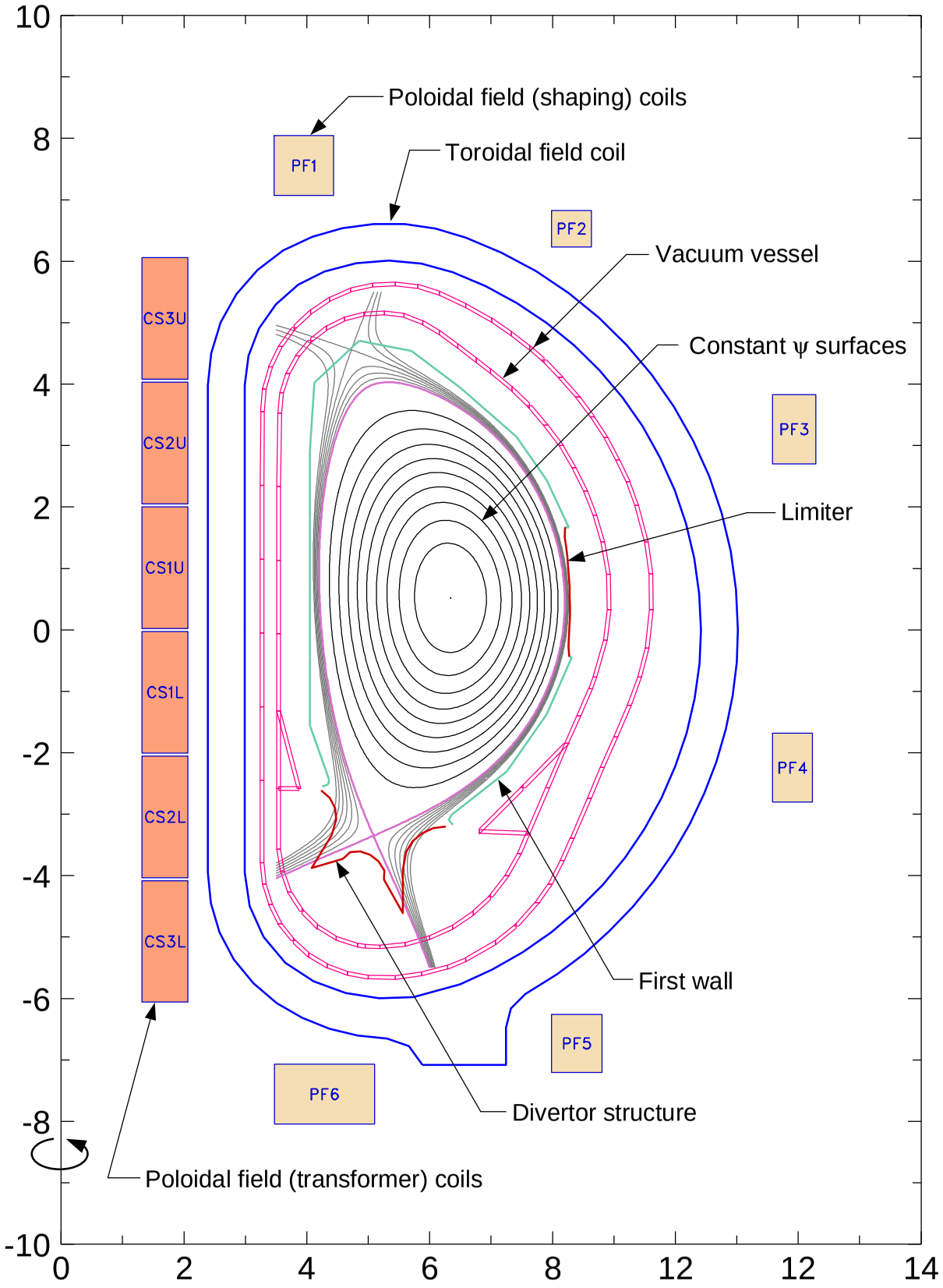
Dead Start Procedure (cont.)

The dead start procedure is defined in a script file:

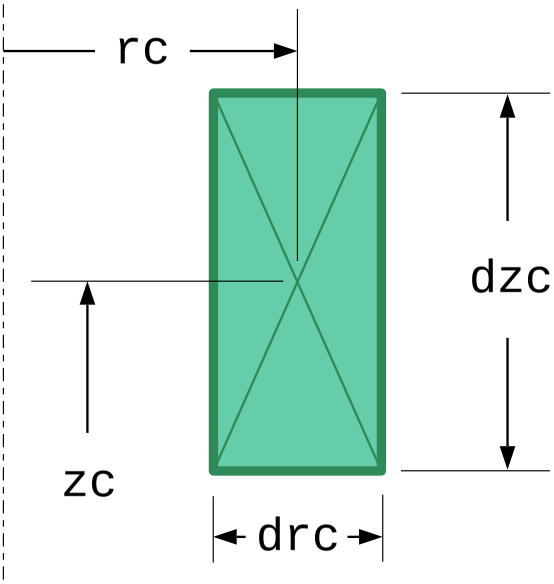
```
caltrans tokamak.bas
ds("help")
```

The dead start routine, `ds`, accepts one or two arguments: file names for the plasma and (optional) coil specifications:

```
ds("plasma.inp"[,"pfcoil.inp"])
```

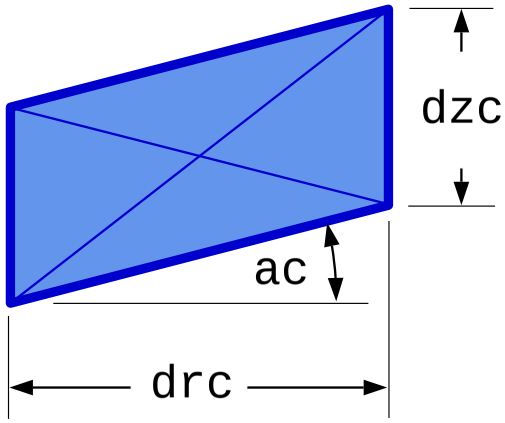

Poloidal field (shaping) coils
Toroidal field coil
Vacuum vessel
Constant ψ surfaces
Limiter
First wall
Divertor structure
Poloidal field (transformer) coils
PF1
PF2
PF3
PF4
PF5
PF6
CS3U
CS2U
CS1U
CS1L
CS2L
CS3L
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
2
4
6
8
12
12
14

Corsica coil geometric specifications



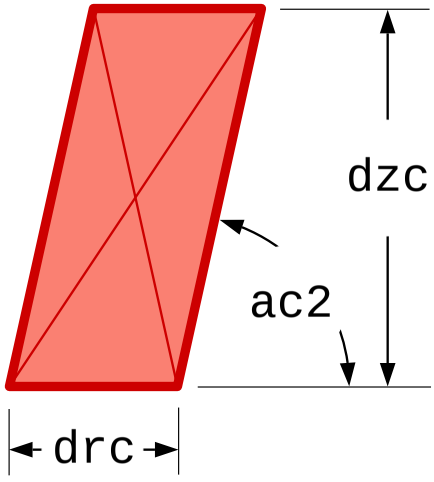
rectangular
ac=ac2=0

$Area = drc \times dzc$
(all three types)



type 1 parallelogram
ac2=0

$thk = dzc\ cos(ac)$



type 2 parallelogram
ac=0

$thk = drc\ sin(ac2)$

Dead Start Procedure (cont.)

First, create a file containing plasma parameters:

```
"KSTAR/DN"

Plasma...
      2.00 MA       plasma current
      1.80 m        major radius
      0.50 m        minor radius
      0.00 m        Zaxis
      1.90          95% elongation
      0.30          95% triangularity
      0.00 m        Dsep (DN)
      1.00          poloidal beta
      0.73          li
     -4.00 Wb       External flux linkage

Toroidal field...
3.50 T @ R = 1.80 m

Computational grid...
      1.00 m        Rmin
      2.60 m        Rmax
     -1.50 m        Zmin
      1.50 m        Zmax
 33 x 65            No. grid points (Nr x Nz)

Plot Scales...
      0.00 m        Rmin
      4.00 m        Rmax
     -2.50 m        Zmin
      2.50 m        Zmax
      2.00 MA/m^2   Current density for drawing coil cross-sections
```

Dead Start Procedure (cont.)

Optionally, create a file containing coil specifications:

```
"KSTAR" PF coil set of 05/01/99 from Kim
 14 coils
name    Rc [m]   Zc [m]  DRc [m]  DZc [m]   n_turn    NI_cap  B_cap
"PF1U"  0.5610   0.2470   0.2135   0.4764        1      1       1
"PF2U"  0.5610   0.6932   0.2135   0.3808        1      1       1
"PF3U"  0.5610   0.9960   0.2135   0.1896        1      1       1
"PF4U"  0.5610   1.2510   0.2135   0.2852        1      1       1
"PF5U"  1.0850   2.2960   0.3330   0.3808        1      1       1
"PF6U"  3.0900   1.9200   0.1896   0.3808        1      1       1
"PF7U"  3.7300   0.9600   0.1418   0.2852        1      1       1
"PF1L"  0.5610  -0.2470   0.2135   0.4764        1      1       1
"PF2L"  0.5610  -0.6932   0.2135   0.3808        1      1       1
"PF3L"  0.5610  -0.9960   0.2135   0.1896        1      1       1
"PF4L"  0.5610  -1.2510   0.2135   0.2852        1      1       1
"PF5L"  1.0850  -2.2960   0.3330   0.3808        1      1       1
"PF6L"  3.0900  -1.9200   0.1896   0.3808        1      1       1
"PF7L"  3.7300  -0.9600   0.1418   0.2852        1      1       1
```

Dead Start Procedure (cont.)

Execute the dead start routine:

```
ds(“plasma.inp”,”pfcoil.inp”)
```

which will create a save-file (in this case `kstar_dn.sav`).

Thereafter, start-up with the save-file:

```
caltrans kstar_dn.sav
```

Dead Start Procedure (cont.)

Auxiliary routines are available to import first-wall/divertor, TF coil geometry and passive structure specifications:

```
read_fwall(“fwall.inp”)
read_passive(“passive.inp”)
read_tfcoil(“tfcoil.inp”)
```

See the document “Creating Tokamak Equilibria with Corsica” for details.