Free Boundary TRANSP Upgrade

Dick Wieland, PPPL
John Schivell
Doug McCune, PPPL
Bernard Balet, JET
Jean-Paul Jeral, JET
Denis O'Brien, JET
Pam Stubberfield, JET
Wolfgang Zwingmann, JET

July 11, 1997

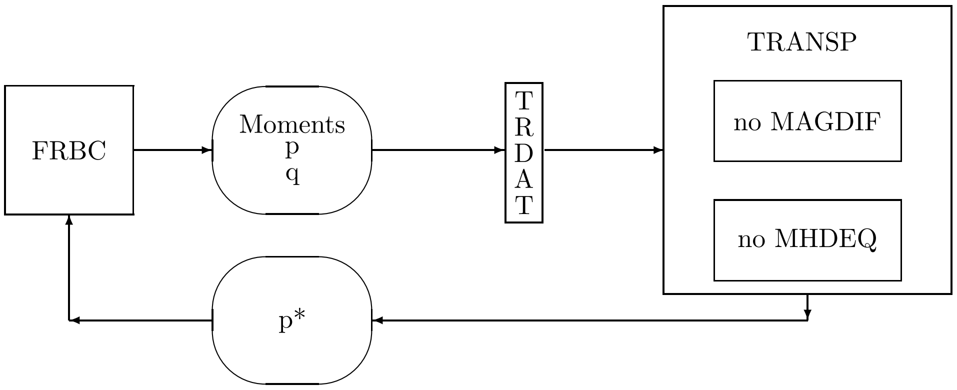
1 Introduction

1.1 Overview

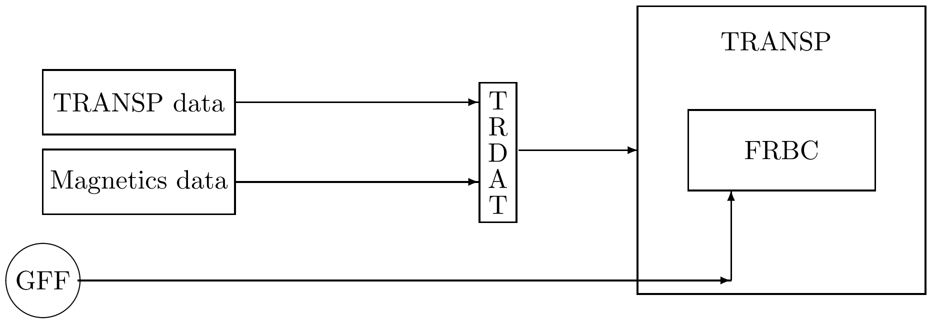
The free boundary equilibrium TRANSP upgrade (code name BRAZIL) is proceeding in two stages, as described in more detail below. In the first stage, code name BRAZIL-0, the results of a standalone free-boundary equilibrium run from either EFIT or VMEC will be used to "drive" the TRANSP run. In the second stage, code name BRAZIL-2, the EFIT or VMEC free-boundary equilibrium code (FRBC) will be incorporated wholly into TRANSP, replacing the fixed-boundary codes that are there now. At one point, we had considered an intermediate stage, code named BRAZIL-1, where TRANSP was to be gutted into the equivalent of a test "driver" for equilibrium codes. Time constraints have led us to delay or altogether eliminateits implementation.

A chronology of the steps taken in the implementation of this design can be found in the Appendix.

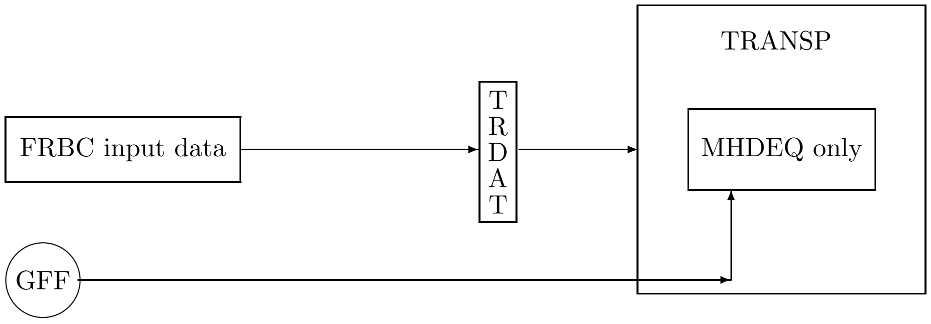
BRAZIL-0 Pass FRBC *output* time series profile data (moments, pressure and q) into TRANSP, where it will be used in lieu of calls to MAGDIF and MHDEQ. The idea here is to run TRANSP with a consistent free boundary equilibrium obtained from one of the FRBC. This means passing in the complete flux surface geometry, as well as pressure and q, as functions of the FRBC radial coordinate. The q will be fixed by the above, while the pressure will evolve in the usual way. The resulting pressure profile (p^*) can be fed back into the FRBC to continue the loop.



BRAZIL-2 In the second phase the FRBC calculation is internal to TRANSP. The *input* data now consists of switches and magnetic coil and probe data. The pressure profile is developed internally, using the ususal TRANSP methodology.



BRAZIL-1/frozen The TRANSP "Equilibrium Driver" phase , where FRBC *input* data is passed into what is a TRANSP shell, where everything *but* the equilibrium code is shut off. In this mode, the profile data that is input must conform to the profile data used by the FRBC codes in standalone. So, for VMEC, that means inputing pressure and magnetic field pitch angle as a function of major radius.



1.2 Prototype vs Production modes for BRAZIL-1 and -2

Certain programming shortcuts can be taken ("Prototype" mode) in getting a prototype version of BRAZIL-1 (and later -2) working in the shortest period of time possible. Efficiencies that would otherwise be considered in designing the code interface and the Green's Function file (GFF) interface can be sidestepped in this mode in the interests of getting started more quickly. The interfaces can be simple, and designed to optimize debugging rather than run-time execution.

Later, after the upgrade has been "validated", and shown to work properly, the interfaces can be upgraded to run more efficiently ("Production" mode).

Use of these two terms, "Prototype" and "Production", throughout the rest of this document, refer to the ideas expressed in the preceding paragraphs.

1.3 TRDAT Namelist Control

The LEVGEO namelist control will have to be extended to accomodate many new options and suboptions that will become available under the "free-boundary" mode of operation. The free boundary options need to distinguish fixed or free boundary, between VMEC or EFIT, and between up-down symmetry or asymmetry. EFIT will probably only be run in the full up-down asymmetric mode. VMEC, on the other hand, will probably be made available with both options, to accomodate the inherent speed-up available when running on TFTR cases.

A way to define this option tree is as follows:

Free Boundary Equilibrium TRDAT Namelist Extensions

scalar name	*Value*	*Description*
LEVGEO	8	BRAZIL Mode
NFRBMODE	0	Run in BRAZIL-0 Passthru Mode
	1	Run in BRAZIL-1 Test (Driver) Mode
	2	Run in BRAZIL-2 Mode
NFRBCODE	1	run with Up-Down Symmetric VMEC
	2	run with Up-Down Asymmetric VMEC
	3	run with Up-Down Asymmetric EFIT

This more logically divides the specification into “free-boundary”, what kind of “free-boundary” - TRANSP interaction, which “free-boundary” code.

As we did for LEVGEO=6, we should define internal variables, not in namelist, that break these options down even further.

1.4 Data Structures and Files

TRDAT and RPLOT will both require data structure changes in order to work in the new BRAZIL mode. A tabulation of proposed changes is given in the following sections. Changes will involve new external data files, as well as new internal data structures in TRDAT, TRANSP and RPLOT. The new file types and TRDAT items correspond more or less to elements in the existing input file set used in running VMEC and EFIT standalone at PPPL and JET, respectively. The RPLOT items correspond to the output from VMEC and EFIT that will be most interesting to view in a time history format.

The FRBC codes require new diagnostic data not currently accomodated by TRDAT. One such data file is the Green’s Function file (GFF), which is in a different format for EFIT and VMEC. In fact, for both EFIT and VMEC, the files are binary. In “Production” mode we want to preserve the ability of TRANSP to be “prepared” on one machine with TRDAT, and yet run on an entirely different machine. One way to do this is to have a complete set of GFF generators and GFF binaries available on each shared disk structure in the TRANSP machine network. The binaries would be labeled in such a way as to make them easily identifiable. The name of the particular GFF binary used in TRANSP run could be input through the TRDAT namelist, and passed to the FRBC as a line in a marker file. Or, TRDAT could make a local copy of the named GFF, assigning it a runid like name that the FRBC is hard coded to read. The GFF generator code should be maintained by the site responsible for maintaining the FRBC. The tokamak specific input files read by the GFF generator codes should be maintained by the individual tokamak labs.

1.5 FRBC - TRANSP interface in BRAZIL-2

Several alternatives are possible for constructing the actual interface between TRANSP and the FRBC. A common requirement is that each call to FRBC consists of preparation of a new input data set of "some appropriate type", followed by an invocation of the FRBC in "some way", and concluded by a retrieval of the necessary information in "some way".

Plug and Play In this option the transfer of data, both prior to the execution of the FRBC and after, is through local files that are very similar, if not exactly like, the files used in running the FRBC standalone. This will facilitate standalone debugging of a problem timeslice. The execution of the FRBC itself can be through a simple internal call mechanism, where all that is passed is the runid. The FRBC will read the locally written timeslice Namelist file, and can be modified to remember the last solution as the new starting point for the equilibrium solution iteration. The information required by TRANSP can be written to a file, and read in by TRANSP directly.

Morph and Run In this option, the FRBC is converted into a set of subroutines with its own set of data structures, which are separated from the TRANSP data structures by a "firewall". All communication between the two takes place through the argument stack of the calling procedure, as is currently the case with LEVGEO=6.

The "Plug and "Play" option is currently under development.

1.6 Post-Processing the Results

Two kinds of post-processing can be imagined:

1. Each FRBC now has its own post-processing code suite, and if the FRBC in TRANSP writes time-slice output files with suitably unique names, they can be easily referenced by these programs and plots in similar fashion to how they are now can be processed. FRBC users are used to seeing certain graphical output for their timeslice runs, and considerable investment has been made in developing these graphics packages. Rather than try to incorporate these into RPLOT, it makes more sense (???) to preserve the capability for running them directly. These files can be processed after or even during the TRANSP run by the usual set of FRBC post-processing codes, with hopefully only minor modification. In order to limit the number of files produced, it might be wise to introduce a Namelist variable that specifies the frequency at which these output files are written. In the final "Production" mode, it would make sense to maintain a single file for output, and append to it, modifying the post-processing codes accordingly to allow them to select time slices of interest.

2. RPLOT can be used to plot certain quantities that are of interest as a function of *time*, such as convergence parameters, etc. Certain scalar information, especially the kind that describes the quality of the reconstruction, can be incorporated into TRANSP data structures (so they must form part of the data set that is transmitted back to TRANSP across the TRANSP-FRBC firewall), and be made available to RPLOT users. Early on, we thought that users might want to use RPLOT to see how well the reconstructed data agreed with measurements, hence the "error bar" plots in the next table. These and other possible data sets are listed below.

RPLOT Data Structures

Output Data Item	*Data Type*	*RPLOT Structure exists ?*
Convergence Values	$f(t)$	Yes
Total Chi Squared Values	$f(t)$	Yes
Individual Chi Squared Values	$f(t,\text{index})$	Yes
Data with Error Bars[1]	$f(t, value, index)$	New - Data w/ Error Bars
Reconstructed profile curves[2]	$f(t, R)$	New - Curves rather than points
Reconstructed vs Measured Magnetic Data[3]	$f_{MorR}(t, index)$	New - Multigraphs w/ Error Bars
Reconstructed vs Measured Profile Data[4]	$f_{MorR}(t, R)$	No - Multigraphs w/ Error Bars
Poloidal Flux Contour Plot[5]	$f(R, Z, t)$	Post-Process instead ?

Initially, we will concentrate on the "proof of principle" aspects of this project, and will rely mainly on the FRBC post-processing capability. Once that is complete, we can more fully integrate the FRBC results into RPLOT.

1. Error bars will become a new RPLOT datatype.
2. Curves will become a new RPLOT datatype. By curves, we mean data points connected by "lines" rather than represented by symbols. The equilibrium code will have to return these data on a uniform R grid, possibly as determined from the TRANSP namelist, or by default from the Green's function file.
3. These would be represented as multigraphs composed of overlapping point datatypes and error bar datatypes.
4. These would be represented as multigraphs composed of overlapping point, error bar and curve datatypes.
5. In order to retain enough flexibility to vary contour resolution, it might be best for TRANSP to dump the required data for contour post-processing (e.g., coil positions and currents, plasma current density profiles $j(R, Z)$, limiter locii) into a separate file. A seperate utility could then reconstruct the poloidal flux plots.

2 BRAZIL-0 . . . Equilibrium Pass-Thru

In this mode we pass in the complete flux surface geometry, as well as pressure and q, from a completed time sequence FRBC run. TRANSP then uses this data in lieu of running MAGDIF and MHDEQ. We use existing trigraphs, where possible.

2.1 TRDAT Input Files and data Structures

New TRDAT Input File and Data Structures

Input Diagnostic	*Data Type*	*TRDAT Structure*
Scalar Controls	Scalar	Exists
Equilibrium "Moments"[6]	$\mathcal{C}_3(t, x^7, (m * ind)^8)$	3d Ufile (MMX)
Pressure	$\mathcal{P}_2(t, x^7)$	2d Ufile (MPX)
q-value	$\mathcal{Q}_2(t, x^7)$	2d Ufile (MQX)
Toroidal Current Density	$\mathcal{I}_2(t, x)$	2d Ufile (MTC)
Poloidal Current (F)	$\mathcal{J}_2(t, x)$	2d Ufile (MPC)
Enclosed Poloidal Flux	$\mathcal{O}_1(t)$	1d Ufile (PLF)
Enclosed Toroidal Flux	$\mathcal{T}_1(t)$	1d Ufile (TRF)

$\mathcal{C}_3$ Ufile Scalar List

scalar name	*Description*	*Variable*
SYMTYPE	"SYMMETRIC" or "ASYMMETRIC"	SCLAB(2,1)
FRBCTYPE	"EFIT" or "VMEC"	SCLAB(2,2)
XTYPE	"PSINORM" or "PHINORM" (EFIT)	SCLAB(2,3)
FRBCID	the FRBC run id	SCLAB(2,4)
MMAX	max value of m	SCVAL(5)

$\mathcal{P}_2, \mathcal{Q}_2, \mathcal{I}_2, \mathcal{J}_2, \mathcal{O}_1, \mathcal{T}_1$ Ufile Scalar Lists

scalar name	*Description*	*Variable*
FRBCTYPE	"EFIT" or "VMEC"	SCLAB(2,1)
FRBCID	the FRBC run id	SCLAB(2,2)

6. For the up-down asymmetric case: $ind = 1 : R^c_m$, $ind = 2 : R^s_m$, $ind = 3 : Z^c_m$, $ind = 4 : Z^s_m$; for the up-down symmetric case: $ind = 1 : R^c_m$, $ind = 2 : Z^s_m$
7. x is the intrinsic coordinate used by the FRBC code in question. This requires passing $\Phi_0(t)$ and $\Psi_0(t)$ as 1-d Ufiles so TRANSP can convert to ξ. No conversion will be done in TRDAT.
8. A combined index, where m is the poloidal index in the Fourier expansion.

TRDAT Namelist Entries

*scalar name or trigraph**	*Description*
LEVGEO	8
NFRBMODE	0
NFRBCODE	1 (VMEC-S) or 2 (VMEC-A) or 3 (EFIT)
MMX*	$\mathcal{C}_3$ moments Ufile trigraph
MPX*	$\mathcal{P}_2$ pressure Ufile trigraph
MQX*	$\mathcal{Q}_2$ q-value Ufile trigraph
MTC*	$\mathcal{I}_2$ toroidal current Ufile trigraph
MPC*	$\mathcal{J}_2$ poloidal current Ufile trigraph
PLF*	$\mathcal{O}_1$ poloidal flux Ufile trigraph
TRF*	$\mathcal{T}_1$ toroidal flux Ufile trigraph

3 BRAZIL-1 and 2 . . . Internal Equilibrium [Driver]

In the TRANSP free boundary "Equilibrium Driver" phase (BRAZIL-1), FRBC *input* data is passed into the TRANSP driver shell, where everything *but* the equilibrium code is shut off. Some file types have to go through TRDAT so that the data can be available to TRANSP for time interpolation in preparing input data structures for the FRBC call. Others are referenced only by the FRBC itself, such as the Green's Function File (GFF), and usually have very idiosyncratic data formats that would be silly to translate into TRDAT common format and then back again. For these file types, cf. the discussion in Section 1.4.

In the TRANSP free boundary equilibrium phase (BRAZIL-2), the only difference from BRAZIL-1 is that the pressure is no longer input, and now all the TRANSP modules are fully operational.

3.1 Flux Loop Measurements in EFIT and VMEC

Some machines measure absolute poloidal flux with their flux loops, while others measure relative flux between adjacent loops ("saddle fluxes"). EFIT accomodates either method by hard coding within the program itself, while VMEC depends on the GFF to specify the configuration. We might want to modify these codes so that the configuration setup is handled in the same way by both.

3.2 FRBC specific Namelist Files

Namelist variables for the FRBC can be broken down into two categories, those that are likely to be changed often, and those that are not. The former need to be more visible, and should appear in the TRANSP namelist. The latter can be relegated to one or more additional Namelist files, read in directly by the FRBC, or by the setup routine for the FRBC. A naming convention similar to the one proposed for the GFF can be used, where a TRDAT file name entry can be used to identify the generic namelist file, and the contents copied to a file named in such a way as to guarantee its uniqueness for the run in question.

3.3 Limiter Parameterization

Both codes require a description of the limiter boundary, both for internal computation, and for graphics post-processing. In the case of VMEC, this parameterization is input by means of a separate file, whose name will be specified in the same way as described in the previous section.

3.4 Measurement "Weights"

EFIT employs the concept of measurement "weights" to allow the user to vary the degree to which the data is used in the reconstruction. A set of namelist variables will be assigned to represent these weights. Because there will be a disconnect between the location of these weight arrays in TRANSP namelist and the location of the data in the MDF file (cf. next section for details), it would be preferable to have TRDAT be able to report on these namelist "weights".

3.5 TRDAT Input Files and Data Structures

New TRDAT Input File and Data Structures

Input Diagnostic	*Data Type*	*TRDAT Structure*
Scalar Controls[9]	Scalar	Exists
PF Coils Green's Function File	Proprietary	TRDAT
Composite Magnetics Data File (MDF)[10]	$\mathcal{M}_3(t, index, 1:2^{11})$	Indexed 3d Ufile
FRBC Namelist File	Proprietary	TRDAT
Pressure (NFRBMODE=1 only))	$\mathcal{P}_3(t, R, 1:2^{11})$	New - 3d Ufile
MSE Pitch Angle	$\mathcal{Q}_3(t, R, 1:2^{11})$	New - 3d Ufile
Interferometry	$\mathcal{I}_3(t, R, 1:2^{11})$	New - 3d Ufile

TRDAT Namelist Entries

*scalar name or trigraph**	*Description*
LEVGEO	8
NFRBMODE	1 (Driver) or 2 (let 'er rip!)
NFRBCODE	1 (VMEC-S) or 2 (VMEC-A) or 3 (EFIT)
FRBCGFF	name of Green's Function File (Set)
FRBCNMLF	name of the FRBC template Namelist File
EFIT**	cf Section 4.2.3 for EFIT Namelist variables
VMEC**	cf Section 4.1.? for VMEC Namelist variables
CUR*	Plasma Current trigraph[12]
DFL*	existing Diamagnetic Flux trigraph[12]
MDF*	$\mathcal{M}_3$ composite magnetics data file trigraph
MPR*	$\mathcal{P}_3$ pressure Ufile trigraph
MQR*	$\mathcal{Q}_3$ magnetic field pitch angle Ufile trigraph
MFR*	$\mathcal{I}_3$ interferometer Ufile trigraph

9. Cf. Sect 3.5.2 for VMEC entries and Sect 3.5.3 for EFIT entries.
10. Contains set of magnetic measurements required by VMEC and EFIT to perform reconstruction with free boundary equilibrium. Cf. sect 3.2.1 .
11. Index "1" points to "measured values", index "2" to "sigmas".
12. Ufile scalar SIGMA can be used to specify the time independent uncertainty.

3.5.1 $\mathcal{M}_3$ Composite Magnetics Data Ufile (MDF)

$$\text{Composite Magnetics Data Ufile: } \mathcal{M}_3(t^{13}, index, 1:2)$$
$$\text{Measurement} = \mathcal{M}_3(t, index, 1)$$
$$\sigma = \mathcal{M}_3(t, index, 2)$$

Input Diagnostic	*Index, Number*[14]	*Data*	*Index*[15]	*Units*
Flux Loops	IFL, NFL	Φ_{FL}	$i_{FL}, \ldots, i_{FL} + n_{FL}$	Webers
Reference Flux Loop	ISRF, NSRF	Φ_{FLR}	$i_{FLR}(n_{FLR} = 1)$	Webers
Saddle Coils	ISADDL, NSADDL	Φ_{FL}	$i_{SDL}, \ldots, i_{SDL} + n_{SDL}$	Webers
Magnetic Probes	IMPROBE, NMPROBE	B	$i_{MP}, \ldots, i_{MP} + n_{MP}$	Tesla
PF Ohmic Coil[16]	IIOHMIC, NIOHMIC	I_{Ohmic}	$i_{I_{Ohmic}}, \ldots, i_{I_{Ohmic}} + n_{I_{Ohmic}}$	Amps
PF Shaping Coil[17]	IISHAPE, NISHAPE	I_{Shape}	$i_{I_{Shape}}, \ldots, i_{I_{Shape}} + n_{I_{Shape}}$	Amps
Current Center	IRELIP, NRELIP	R_{Ellip}	$i_{RLP}(n_{RLP} = 1)$	Meters
Current Center	IZELIP, NZELIP	Z_{Ellip}	$i_{ZLP}(n_{ZLP} = 1)$	Meters

Other diagnostic information can be present, as may be the case if other applications make use of these Ufiles. TRDAT will ignore it.

In the case of TFTR, the B_R, B_Z loops are stored as "Magnetic Probes", the Mirnov coils are stored as "Flux Loops", and the Poloidal Field Coil Currents (OH,EF,VC,HF) are stored as "PF Shaping Coils".

13. Each $\mathcal{M}_3$ Ufile will be built on a single time base. If data is present on more than one time base, then additional $\mathcal{M}_3$ Ufiles will be present, one per time base. A naming convention like "*Mnnnnn.MDFi*" can be used to link them together, with $i = 1, \ldots, n$ covering the n different time base data sets.
14. These parameters are hidden in the MDF. The FRBC knows how to map a given index to a complete specification of the measurement geometry as given by the GFF.
15. Each "named" index pair (i, n) will appear as separate scalars in the ufile. Not every pair has to be present, since not every data type need be present.
16. A useful EFIT terminology labels those PF coils as "E", or "Ohmic" if they are not included in the reconstruction fit.
17. A useful EFIT terminology labels those PF coils as "F", or "Shaping" if they are included in the reconstruction fit.

$\mathcal{M}_3$ Ufile Scalar List appearing in the MDF

scalar name	*Description*
FRBCID	the FRBC run id
MCONFIG	name of "shot-independent" magnetics configuration file
SEQNO	time base sequence number in ufile series
IFL	index value for Flux Loop class of measurements
NFL	number of measurements appearing in the Flux Loop class
IIOHMIC	index value for I_{Ohmic} class of measurements
NIOHMIC	number of measurements appearing in the I_{Ohmic} class
IISHAPE	index value for I_{Shape} class of measurements
NISHAPE	number of measurements appearing in the I_{Shape} class
IMBNDRY	index value for plasma boundary parameters
NMBNDRY	number of plasma boundary parameters
IMFLUX	index value for enclosed plasma flux measurements
NMFLUX	number of enclosed plasma flux measurements
IMPROBE	index value for Magnetic Probe class of measurements
NMPROBE	number of measurements appearing in the Magnetic Probe class
ISADDL	index value for Saddle Coil class of measurements
NSADDL	number of measurements appearing in the Saddle Coil class
ISRF	index value for Reference Flux Loop (one value)
NSRF	= 1; a single Reference Flux Loop measurement
IX3	unknown
NX3	unknown
IRELIP	index value for I_{RLP} class of parameters
NRELIP	number of parameters appearing in the I_{RLP} class
IZELIP	index value for I_{ZLP} class of parameters
NZELIP	number of parameters appearing in the I_{ZLP} class

In the case of TFTR, the magnetically determined $R_{C_0}, Z_{C_0}, a_C, b_C$ parameters describing the plasma boundary are stored in the MBNDRY positions, and the magnetically determined toroidal and poloidal fluxes are stored in the MFLUX positions.

TRCOM Non-Namelist Common Block Variables

Variable	*Description*
integer NFEROM	Number of Interferometry Measurements
integer NFL	Number of Flux Loops
integer NIOHMIC	Number of Ohmic PF Coils
integer NISHAPE	Number of Shaping PF Coils
integer NITEREQ	number of FRBC iterations
integer NMPROBE	Number of Magnetic Probes
integer NRELIP	Number of Current Centers
integer NZELIP	Number of Current Centers
integer NSDL	Number of Saddle Coils
integer NSRF	Number of Ref Flux Loops (Obsolete?)
real*4 BMP(1:NMPROBE)	Magnetic Probe Measurements
real*4 BMPEQ(1:NMPROBE)	FRBC Magnetic Probe Values
real*4 BMPSG(1:NMPROBE)	Magnetic Probe Sigmas
real*4 CHISEQ	Reconstruction chi-squared pdf
real*4 CHISEQP	Reconstruction CHISQ for pressure profile
real*4 CHISEQMS	Reconstruction CHISQ for MSE
real*4 CHISEQFL	Reconstruction CHISQ for flux loops
real*4 CHISEQMP	Reconstruction CHISQ for magnetic probes
real*4 CHISEQSD	Reconstruction CHISQ for saddle coils
real*4 CHISEQDI	Reconstruction CHISQ for diamagnetic flux
real*4 CHISEQIP	Reconstruction CHISQ for toroidal current
real*4 CHISQN	Reconstruction CHISQ/N
real*4 CHISQPN	Reconstruction CHISQ/N for pressure profile
real*4 CHISQMSN	Reconstruction CHISQ/N for MSE
real*4 CHISQFLN	Reconstruction CHISQ/N for flux loops
real*4 CHISQMPN	Reconstruction CHISQ/N for magnetic probes
real*4 CHISQSDN	Reconstruction CHISQ/N for saddle coils
real*4 CHISQDIN	Reconstruction CHISQ/N for diamagnetic flux
real*4 CHISQIPN	Reconstruction CHISQ/N for toroidal current
real*4 CMBNDRY(MBNDRY)	Magnetic reconstruction of Plasma Boundary
real*4 CMTFLUX	Magnetic reconstruction of Toroidal Flux
real*4 CMPFLUX	Magnetic reconstruction of Poloidal Flux
real*4 CONVEQ	Equilibrium convergence
real*4 CPFOH(1:NIOHMIC)	Ohmic PF Coil Currents
real*4 CPFOHEQ(1:NIOHMIC)	FRBC Ohmic PF Coil Currents
real*4 CPFOHSG(1:NIOHMIC)	Ohmic PF Coil Current Sigmas
real*4 CPFSH(1:NISHAPE)	Shaping PF Coil Currents
real*4 CPFSHEQ(1:NISHAPE)	FRBC Shaping PF Coil Currents
real*4 CPFSHSG(1:NISHAPE)	Shaping PF Coil Current Sigmas
continued next page	

Variable	*Description*
continued from previous pg	
real*4 DFLUX	Diamagnetic Flux
real*4 DFLUXEQ	FRBC Diamagnetic Flux
real*4 DFLUXMSG	Diamagnetic Flux Sigma (M???)
real*4 FEROM(1:NFEROM)	Interferometry Measurements
real*4 FEROMSG(1:NFEROM)	Interferometry Sigmas
real*4 FEQWGTF(1:300)	FRBC Wgts for Flux Loops (Obsolete?)
real*4 FEQWGTI(1:300)	FRBC Wgts for Shaping Coil Currents (Obsolete?)
real*4 FEQWGTM(1:300)	FRBC Wgts for Magnetic Probes (Obsolete?)
real*4 FRBCWGTF(1:300)	FRBC Wgts for Flux Loops
real*4 FRBCWGTI(1:300)	FRBC Wgts for Shaping Coil Currents
real*4 FRBCWGTM(1:300)	FRBC Wgts for Magnetic Probes
real*4 FRBCWGTR(1:300)	FRBC Wgts for Interferometer Data
real*4 FRBCWGTS(1:300)	FRBC Wgts for MSE Data
real*4 FRBITMPI(1:300)	Minimum magnetic probe signal
real*4 PCUR	Plasma Current
real*4 PCUREQ	FRBC Plasma Current
real*4 PCURSG	Plasma Current Sigma
real*4 PHIFL(1:NFL)	Flux Loop Measurements
real*4 PHIFLEQ(1:NFL)	FRBC Flux Loop Values
real*4 PHIFLSG(1:NFL)	Flux Loop Sigmas
real*4 PHISDL(1:NSDL)	Saddle Coil Measurements
real*4 PHISDLEQ(1:NSDL)	FRBC Saddle Coil Measurements
real*4 PHISDLSG(1:NSDL)	Saddle Coil Sigmas
real*4 SIREF	Reference Flux Loop Measurement

3.5.2 VMEC Namelist entries

VMEC Namelist Entries appearing in TRDAT file

Namelist entry	*Description*
NVM8IRAX	optimize wrt magnetic axis position
NVM8ITER	maximum number of iterations allowed
NVM8MPHI	fix toroidal flux
NVM8SIN	initial radial grid size
NVM8STEP	output status line every *nstep* lines
NVM8VSKP	iterations between updates of vacuum solution
VM8FTOL	convergence criteria for MHD force residual
VM8TENSI	tension in iota splines
VM8TENSP	tension in pressure splines

3.5.3 EFIT Namelist entries

EFIT Namelist Entries appearing in TRDAT file

Namelist entry	*Description*
EBITFAR	Minimum polarimetry angle
EBITSAD	Minimum magnetic saddle signal
EERROR	Convergence criterion
EFCURBD	1=0 ffprime at boundary, 0= float
EFWTBP	Wgt for ffprime and pprime proportional
EFWTCUR	Wgt on current measurement
EFWTDLC	Wgt on diamagnetic flux
EFWTFAR	Wgt for polarimetry
EFWTQA	Wgt on q0 measurement
EPCURBD	related to edge pressure
ESERROR	Standard deviation for fitting
ESVDTOL	LSVD tolerance
ERELIP	initial guess for R of current centre
EZELIP	initial guess for Z of current centre
FRBITMPI(MP2)	minimum magnetic coil signals
FRBCWGTF(SIL)	weight on the flux loop measurements
FRBCWGTS(MSE)	weight for mse pitch-angle measurments
FRBCWGTM(MP2)	weight on the pick-up coil measurements
FRBCWGTI(FCOIL)	weight on the f-coil measurements
FRBCWGTL(SDL)	weight on the saddle measurements
FRBCWGTR(IFEROM)	weight on the saddle measurements
NDOKINE	Kinetic fitting on
NICONVRE	fitting options
NICURRT	type of expansion
NIECOILE	Plotting switch
NIECURRE	1=include E-coils in calc.
NIEXCALE	1=plot magnetic signals
NIFITVSE	1= fit vessel segments
NKFFCURE	No. of terms in ffprime
NKPPCURE	No. of terms in pprime
NKPRFITE	Pressure fitting on
NKWRIPRE	Output switch 1
NMXITERE	Max. no of outer iterations
NXITERE	Max. no of inner iterations

3.5.4 EFIT Namelist entries

EFIT Namelist Entries appearing in ESNAP2 file

Namelist entry	*Description*
ICNTOUR	???
IPRES	???
KEQDSK	???
KBOUND	???
KDOSCRUN	0 = regular SCRUNCHER on all contours
	1 = like 0 but with Zakharov representation
	2 = Lao-Hirshman JET parameterization (65)
	3 = Lao-Hirshman JET parameterization (21)
	5 = Lao-Hirshman JET parameterization (bdy)
	6 = SCRUNCHER on outermost boundary
MPOLSCRUNCH	9 = return $m = 0, ..., 8$ moments to EFREAD
QVFIT	???
ITEK	???
IERCHK	???
ICFLUX	???
LIMITR	number of limiter pts
XLIM	limiter positions
YLIM	limiter positions
IFITVS	???
KCGAMA	???
CGAMA	???
XGAMA	???
KCALPA	???
CALPA	???
IRON	???
IRONS	???
FWTPER	???
GAINP	???
BITFC	???
PSIBIT	???
IFCURR	???
KDZCUR	???
ICONVR	???
XALPA	???

4 Calling the FRBC Code from TRANSP

The subroutine MHDEQ contains the TRANSP code that calls whatever equilibrium package is specified by the LEVGEO Namelist parameter. Up until LEVGEO=8, the connection has always been made by a subroutine call; i.e., the equilibrium package has been subsumed into TRANSP through a callable interface which additionally serves as a firewall between the TRANSP common and the equilibrium code common.

With LEVGEO=8, this paradigm will change. The equilibrium code will be a separate process, invoked by a system call from MHDEQ, and information will be passed back and forth using namelist reads and writes through "logical" files (i.e., pipes). In the beginning this IO will be in ASCII format. Eventually, the IO may become binary.

The advantages of such a calling architecture are

1. Alleviate the necessity of possibly massive code reorganization to convert the FRBC code into a "callable" entity.

2. Use the exisiting "restart" capability of the FRBC codes

3. Minimum conversion costs when the FRBC needs to be updated

This separation, however, does bear some cost:

1. A "restart" mechanism has to be established for the FRBC. While this would have to be done in any case, it does increase the amount of logical file IO that has to go on.

2. A mechanism has to be established so that the FRBC can signal MHDEQ if and when there is a crash in the FRBC code.

4.1 Calling VMEC from MHDEQ

[To be filled in ...]

4.2 Calling EFIT from MHDEQ

4.2.1 Required Files

Files required for TRANSP - EFIT operation

File Name	*Description*
for TRANSP ...	
./esnap2	K-File template namelist read in EFITRIDE
	contains vbls not in TRDAT namelist
./efit_beg	Unix script used to setup EFIT links
./efitgrunt	Unix script used to startup EFIT in STPIPS
	as well as make the runid_eftdir work directory
for the SED filter ...	
*runid*TE.SCR	sed input translation script for TRANSP to EFIT pipe
*runid*ET.SCR	sed input translation script for EFIT to TRANSP pipe
for EFIT ...	
./brazil	EFIT input script referenced in EFITGRUNT
EC3333	Green's Function file
EP3333	Green's Function file
RE3333	Green's Function file
DPROBE	Green's Function file
ANGLE	Green's Function file
TAREAF	Green's Function file
RFCOIL	Green's Function file

Files generated in the course of TRANSP - EFIT operation

File Name	*Description*
by TRANSP ...	
./fort.45	a copy of the IN1 namelist being sent to EFIT
by EFIT ...	
./EQDSK	the EQDSK file
./EQDSKA	the EQDSKA file ???
./err	stderr for EFIT
./fort.6	stdout for EFIT
./fort.40	??
./fort.72	??
./fort.73	a copy of the BNDMOM namelist being sent to TRANSP
./fort.74	??
./fort.96	??

4.2.2 Program Flow

In the flow description, below, "T*n*" signifies the *n*-th step in the TRANSP code, and "E*n*" signifies the *n*-th step in the EFIT code. "S" signifies an operation by the SED filters, of which there are two.

T1. Call MKPIPS ... only once. The order of the following operations is *not* significant ...

- Create pipe *runid*TS.PIP for writing from TRANSP to the filter
- Create pipe *runid*SE.PIP for writing from the filter to EFIT
- Create pipe *runid*ES.PIP for writing from EFIT to the filter
- Create pipe *runid*ST.PIP for writing from the filter back to TRANSP

T2. Call STPIPS. The order of the following operations *is* significant ...

- Put the back filter (TRANSP $\Leftarrow$ EFIT) process in place
- Open T $\Leftarrow$ S pipe in read mode
- Start EFIT with the command

  ```
  ./efitgrunt runid &
  ```

  [Cf. E1 for consequence]
- Put the forward filter (TRANSP $\Rightarrow$ EFIT) process in place
- Open T $\Rightarrow$ S pipe in write mode

E1. EFIT wakes up ...

- Pipes activity triggered by non-blank arg[1] in command line of efitgrunt invocation
- in DATA, call EFPIPS ... open EFIT input and output pipes
  - Open S $\Leftarrow$ E pipe (used for WRITEing back to TRANSP)
  - Open S $\Rightarrow$ E pipe (used for READing from TRANSP)
- in DATA, wait for TRANSP by issuing a read for the IN1 namelist

T3. Make a System call to *sleep* for 3 seconds ... required on the DEC Alpha to prevent an apparent "race" condition

T4. Call FBEGAT ... interpolate MDF data onto ZT2 time slice and update the TRCOM arrays

T5. Call EFITRIDE ... generate K File for EFIT from TRCOM

- Read the ESNAP K-File template and then assemble the data arrays
- Write the IN1 namelist to fort.45 for debug purposes
- Write the IN1 namelist to the pipe [Cf. S1 for consequence]

T6. Call HWRPIP ... signal EFIT that the K File is ready with "GO EFIT"

S1. SED reads the TS pipe, filters it into the SE pipe [Cf. E2 for consequence]

E2. EFIT digests the namelist ...

- in DATA EFIT completes the read of the namelist data
- still in DATA, use HRDPIP to read "GO EFIT" to confirm the TRANSP transmission of data is complete

E3. EFIT calculates an equilibrium ...

E4. in BETALI EFIT calls SCRUNCHER to parameterize the boundary flux surface

E5. in BETALI write the BNDMOM namelist to fort.73 for debug purposes

E6. in BETALI write the BNDMOM namelist back into the pipe [Cf. T6 for consequence]

E7. in BETALI use HWRPIP to signal completion with "GO TRANSP" [Cf. T7 for consequence]

S2. SED reads the ES pipe, filters it into the ST pipe

T7. Call EFREAD ... wait for EFIT to finish by issuing a read for the returning namelist

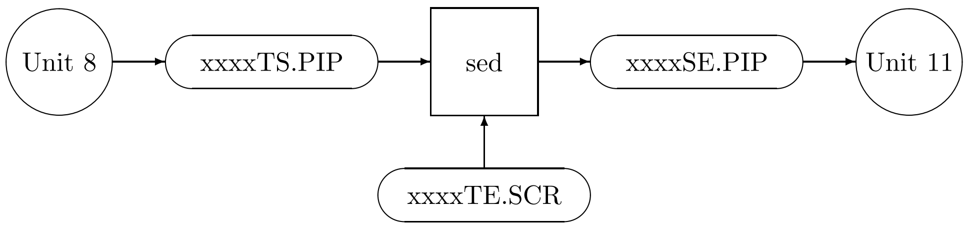
- read the BNDMOM namelist from the S $\Rightarrow$ T pipe
- interpolate the data into the TRCOM arrays

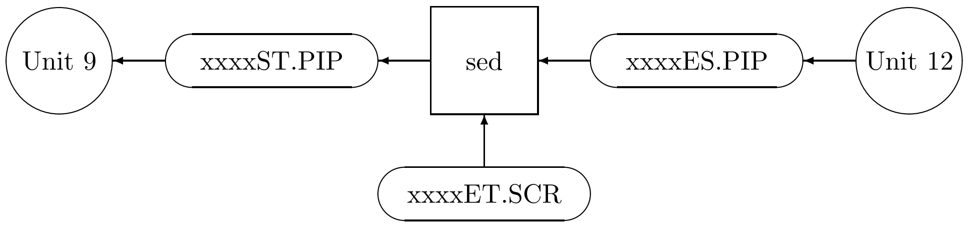
T8. Call HRDPIP ... wait for EFIT to send "GO TRANSP"

T9. Call VMEC6INI ...

- run the fixed boundary equilibrium code to map the interior of the free-boundary solution into moment space
- use as many EFIT results as possible

T10. Manually invoke *kill-brazil runid* at the end of the run to clear out all the pipes





4.2.3 Parameter Passing to EFIT through the K File

TRCOM TRANSP Vbl	*EFIT Namelist Vbl*	*Description*
integer NDOKINE	NDOKIN	Kinetic fitting on
integer NFEROM(nu?)	NDEN	Number of Interferometry Measurements
integer NICONVRE(nu?)	???	fitting options
integer NICURRT(nu?)	???	type of expansion
integer NIECOILE	IECOIL	Plotting switch
integer NIECURRE	IECURR	1=include E-coils in calc.
integer NIEXCALE	IEXCAL	1=plot magnetic signals
integer NIFITVSE(nu?)	IFITVS	1= fit vessel segments
integer NKFFCURE	KFFCUR	No. of terms in ffprime
integer NKPPCURE	KPPCUR	No. of terms in pprime
integer NKPRFITE	KPRFIT	Pressure fitting on
integer NKWRIPRE	KWRIPRE	Output switch 1
integer NMXITERE	MXITER	Max. no of outer iterations
integer NPRESS	NPRESS	Number of pressure points
integer NXITERE(nu?)	NXITER	Max. no of inner iterations
real EBITFAR(nu?)	BITFAR	Minimum polarimetry angle
real EBITSADA	SADBIT	Minimum magnetic saddle signal
real EERROR	ERROR	Convergence criterion
real EFCURBD	FCURBD	1=0 ffprime at boundary, 0= float
real EFWTBP	FWTBP	Wgt for ffprime and pprime proportional
real EFWTCUR	FWTCUR	Wgt on current measurement
real EFWTDLC	FWTDLC	Wgt on diamagnetic flux
real EFWTFAR	FWTFAR	Wgt for polarimetry
real EFWTQA	FWTQA	Wgt on q0 measurement
real EPCURBD	PCURBD	related to edge pressure
real ESERROR	SERROR	Standard deviation for fitting
real ESVDTOL	SVDTOL	LSVD tolerance
real EZELIP	ZELIP	Initial guess for current centre
real PRESSR	PRESSR	Pressure measurement
real RPRESS	RPRESS	Radial position of pressure measurement
real SIGPRE	SIGPRE	Pressure sigma
continued next page		

TRCOM TRANSP Variable	*EFIT Namelist Variable*	*Description*
continued from previous pg		
real*4 BMP(1:NMPROBE)	EXPMP2	Magnetic Probe Measurements
real*4 BMPSG(1:NMPROBE)	N/A	Magnetic Probe Sigmas
real*4 BZXR/296.0	BTOR	Vacuum Toroidal Magnetic Field
real*4 CPFOH(1:NIOHMIC)	ECURRT	Ohmic PF Coil Currents
real*4 CPFOHSG(1:NIOHMIC)	N/A	Ohmic PF Coil Current Sigmas
real*4 CPFSH(1:NISHAPE)	BRSP	Shaping PF Coil Currents
real*4 CPFSHSG(1:NISHAPE)	N/A	Shaping PF Coil Current Sigmas
real*4 DFLUX	DFLUX	Diamagnetic Flux
real*4 DFLUXSG	SIGDLC	Diamagnetic Flux Sigma
real*4 FEROM(1:NFEROM)	DN2KG1	Interferometry Measurements
real*4 FEROMSG(1:NFEROM)	N/A	Interferometry Sigmas
real*4 FRBCWGTF(1:NFL)	FWTSI	FRBC Wgts for Flux Loops
real*4 FRBCWGTI(1:NISHAPE)	FWTFC	FRBC Wgts for Shaping Coil Currents
real*4 FRBCWGTL(1:NSDL)	FWTSAD	FRBC Wgts for Saddle Coils
real*4 FRBCWGTM(1:NMPROBE)	FWTMP2	FRBC Wgts for Magnetic Probes
real*4 FRBCWGTR(1:NFEROM)	FWTKG1	FRBC Wgts for Interferometer Data
real*4 FRBCWGTS(1:300)	FWTTGA ???	FRBC Wgts for MSE Data
real*4 FRBITMPI(1:NMPROBE)	BITMPI	Minimum Magnetic Probe Signals
real*4 PCUR	PLASMA	Plasma Current
real*4 PCURSG	N/A	Plasma Current Sigma
real*4 PHIFL(1:NFL)	COILS	Flux Loop Measurements
real*4 PHIFLSG(1:NFL)	N/A	Flux Loop Sigmas
real*4 PHISDL(1:NSDL)	SADLS	Saddle Coil Measurements
real*4 PHISDLSG(1:NSDL)	N/A	Saddle Coil Sigmas
real*4 SIREF	SIREF	Reference Flux Loop Measurement
real*4 ZPRESS	ZPRESS	Z position of pressure measurement

4.2.4 Parameter Passing back to MHDEQ

EFIT Namelist Variable	*TRCOM TRANSP Variable*	*Description*
NSURTR	integer MSURTR	Number of EFIT flux surfaces returned
FPOLTR	real(1:MSURTR) ZFPOLTR	Poloidal Current array (F)
PRESTR	real(1:MSURTR) ZPRESTR	Pressure array
PHIN	real(1:MSURTR) ZPHIN	Normalized toroidal flux array
PHI	real(1:MSURTR) ZPHI	Toroidal flux array
NMBOUT1	integer MMBOUT	not used
RMBOUT	real ZRMBOUT(0:8,2,65)	Scrunched flux surface moments for R
ZMBOUT	real ZZMBOUT(0:8,2,65)	Scrunched flux surface moments for Z
not yet implemented ...		
TSAISQ	real*4 CHISEQ	Reconstruction chi-squared pdf
ERRORM	real*4 CONVEQ	Equilibrium convergence
IXT	integer NITEREQ	number of FRBC iterations
CPASMA	real*4 PCUREQ	FRBC Plasma Current
CDFLUX	real*4 DFLUXEQ	FRBC Diamagnetic Flux
???	real*4 PHIFLEQ(1:NFL)	FRBC Flux Loop Values
???	real*4 BMPEQ(1:NMPROBE)	FRBC Magnetic Probe Values
???	real*4 CPFOHEQ(1:NIOHMIC)	FRBC Ohmic PF Coil Currents
???	real*4 CPFSHEQ(1:NISHAPE)	FRBC Shaping PF Coil Currents
???	real ZVMEITOR	Toroidal Current

4.3 EFIT - TRANSP interface issues

Cartesian to Moments Mapping It was found that the *SCRUNCHER* code was unable to do a satisfactory job of mapping the inidividual flux contours back into a suitable moments representation. The process introduced radial "noise" into the moment profiles, returned high-curvature theta representations at the edge, and was time consuming. What is done instead now is to use the *VMEC6* fixed boundary code to generate the moment maps, using the boundary and pressure and q profiles supplied by EFIT. It turns out to be faster than using the SCRUNCHER code.

Profile Interpolation Care must be taken in mapping EFIT profiles back into TRANSP and on into VMEC6. It was found that, especially at the edge, too many interpolation stages can lead to spikes in the toroidal current profile.

SCR pipe filters and Fortran Namelist Fortran Namelist writes are done differently on the DEC Alphas and the Suns. The SCR sed scripts cannot depend on any of the following "features" of a namelist write:

1. upper or lower case of the variable names

2. occurance and positioning of “blanks” in the output line
3. positioning of “equal sign” in the output line

4.4 Initializing the Environment

1. Setup GFF’s in proper directory (cf *efitrun*)
2. Setup your path correctly
3. Setup links and work subdirectory

```
setupefit \{normal|debug\} 65 runid
```

4.4.1 Debugging the Pipes

1. *setupefit debug 65 runid*
2. Start up TRANSP in or out of the debugger
3. Set a breakpoint somewhere in TRANSP
4. An EFIT dbx window will appear ...

4.4.2 Debugging EFIT off-line

1. Put fort.45 through the filter

   ```
   cat fort.45 | sed -f X706TE.SCR > fort.efit
   ```

2. Debug EFIT in “no-pipes” mode

   ```
   dbx ./efit14.exe
   2
   0 3 0.1
   junk
   junk
   0
   1
   fort.efit
   ```

A Status Report on TRANSP Free Boundary Project

To review our progress to date in implementing a free-boundary equilibrium analysis in TRANSP:

Fall 1994
: Begin discussions at APS with Lang Lao (EFIT) and Steve Hirshman(VMEC) on feasibility of using their codes in a free-boundary TRANSP.

Spring 1995
: Begin collaboration between PPL and JET, with JET working on the EFIT implementation and PPL working on the VMEC implementation; begin writing a specification for the interfaces

Summer 1995
: Elaborate on the interface specification; code the interface into the respective codes. Wieland and McCune spend two weeks and one week, respectively, at JET.

December 1995
: O'Brien and Stubberfield spend two weeks at PPL. 1st EFIT iteration runs in TRANSP in a preliminary set-up using an early version of the JET EFIT code on a 33 x 33 grid.

Spring 1996
: Set up more recent version of JET EFIT to easily work on either a 33 x 33 grid or a 65 x 65 grid. Interfaces between the two codes tidied up and the start-up phase of TRANSP made to work.

May 1996
: 1st VMEC iteration runs in TRANSP

Summer 1996
: Problems encountered in mapping EFIT equilibrium of a JET shot back into TRANSP. It had been noticed early on that the set of moments provided by the SCRUNCHER called in EFIT caused difficulties back in TRANSP when derivatives were calculated for geometric coefficients. This was particularly noticeable in the current density profile at the edge. A slight kink was seen in limiter configuration which became much more pronounced once the X-point formed. Using the newer EFIT with a better grid resolution did not solve this problem. Different methods were then tried to provide a better set of moments - eg: a more up-to-date SCRUNCHER, with and without Zahkarov option, Lao-Hirshman representation, all with varying number of moments up to max 0:8. All methods were tested in a stand-alone code on points provided by the contouring

algorithm in EFIT for the outermost 2 flux surfaces at a time in the middle of a run where the X-point has formed. Some options gave better fits than others but overall there was very little improvement when these were used in TRANSP.

Fall 1996
VMEC mode begins final testing phase on a variety of shots; work continuing on resolving the EFIT to TRANSP mapping problem. The outermost boundary from EFIT was pulled in slightly by differing amounts, particularly to get rid of the most pronounced X-point shape. This had the main effect of shifting the discontinuity in the j profile radially. It was noticed that the contouring algorithm in EFIT CNTOUR was failing on the outermost boundary in X-point configuration giving fewer points from the SURFAC subroutine. CNTOUR has now been replaced by the most up-to date routines from GA and handles X-points without difficulty. However the problem in TRANSP remained.

Late Fall 1996
Because of the poor performance of the SCRUNCHER in mapping the EFIT interior flux surfaces back into moment space, we have begun experimenting with using VMEC6 fixed-boundary as a way to do the mapping. It turns out to be a very good solution. The current profile spike at the edge was greatly reduced, but now the edge current goes negative.

January 1997
Wieland spends 10 days at JET. EFIT-VMEC6 interface interpolation problems are identified and fixed. Plans are made for further improvements.

B Mapping EFIT back into TRANSP

This summarizes the scheme that was determined to be unsuitable:

- The EFIT equilibrium is analyzed and contours in R,Z space are generated on a specified flux coordinate grid.
- These contours are individually Fourier analyzed by the SCRUNCHER code to extract the Fourier moments needed by the TRANSP code

The problem that develops is in the edge radial behavior of these moment profiles. The moment profiles develop rapid spike like changes in the edge region. These reflect the change in geometry near the edge, and particularly the change in theta representation near the edge. These spikes lead to unrecoverable errors in TRANSP when calculating flux surface metrics.

We have gone back and analyzed LEVGEO=6 fixed boundary runs of similar JET runs, using SCRUNCHER on the resulting flux surface contours and comparing the moments thus obtained with the moments from the VMEC fixed boundary analysis. The results are strikingly different, emphasizing the indeterminacy of fixing the theta representation of the surfaces without additional constraints, such as the force balance equation used in VMEC.

To date the following steps have been taken in attacking this problem:

EFIT contouring
: The contouring algorithm in EFIT has been examined to see if there is any "noise" in the R,Z description of the contours produced. The contouring subroutines have been updated to reflect the most up-to-date versions available from GA. We are satisfied that this part of the code is working correctly.

SCRUNCHER moments analysis
: SCRUNCHER has been modified to work in a mode where it uses the previous solution (i.e., the results of the analysis of a neighboring contour). This was done because there is concern that this analysis does not provide any radial smoothness constraints in going from contour to another; they are treated independently. There is noise in the radial distribution of the resulting moments, but that does not seem to be the cause of the problem. We have tried smoothing these profiles, to no avail.

Moments interpolation
: The interpolation of moments from EFIT to TRANSP radial grids has been examined, and improvements have been made in how the interpolations are being done and where. But no improvement in the final flux surface metrics is seen.

In telephone discussions between the participants, and at a meeting held at PPL on Friday, November 22 (Balet, McCune, Wieland and Zarnstorff), a short term plan of attack has been formulated:

- Take the EFIT boundary in the current implementation and discard the internal solution; run VMEC fixed boundary code to, in effect, obtain a smooth moments representation in the interior, using the EFIT boundary and the EFIT pressure and q profiles. It may turn out to be faster than using SCRUNCHER to analyze the individual contours.A byproduct of this approach is an easy comparison of the internal solutions provided by the two codes.

- Investigate the possibility of using a different flux surface representation for the flux surfaces, one in which the theta representation changes less drastically near the plasma edge in x-point plasmas. Suggestions include using a straight polar representation, or using a heuristic method (as in the BLOAT algorithm) that begins at the plasma boundary and moves inward to the magnetic axis.

- Reinvestigate the effect of shrinking the effective plasma boundary on the radial moments distribution.
- Port the JET TRANSP-EFIT code over to the PPL computers and run there, to improve the debugging accessibility for PPL folks.
- Test the interface at PPL by plugging in a TFTR version of EFIT and running on a TFTR shot to see if any spiking behavior appears.

C Plan of Action January, 1997

1. Sort out CNTOUR1,2 problem in EFIT (RMW,PS; done)
2. Upturn in j profile at edge (RMW,PS,DMc)
3. Change EFIT to not read Green's Functions every call (PS,WZ,JPJ)
4. Tidy up what switches to put in TRANSP namelist and which to put in the ESNAP2 file (RMW,PS,DOB,WZ)
5. CVS
    - move JET iron-core version into the repository (WZ,JPJ)
    - discuss tokamak specific modifications with Doug (DOB)
    - DOB to contact Lang
6. Timing tests should be run (NB X737 LEVGEO=6 run to 12s cf with X746)
7. Tie up discrepancies in Ip and Bz BZXRCMP,IPCMP (RMW)
8. Sort out use of NMOMS in VMEC6, put in Namelist? (RMW)
9. PPF or EQDSK output from TRANSP/EFIT version at JET; wait until we get production version - use facilities from JAC version
10. Extra output from RPLOT to facilitate mhd code comparisons(RMW)
11. IDL - multiplot - discuss with Doug
12. Modify TRANSP to enable running NLMDIF=T option alongside MHD for comparison (RMW,DMc)
13. Fitting q0 - Ufile input (RMW,PS)

14. Input magnetic axis from soft X-ray generalise switching from one mode to another
15. Fitting to j profile
16. Move later version of EFIT to SUN (RAL,JPJ)
17. Effect of rotation

D EFIT Installation and Operation Notes for JET

```
Structure of the efit_jet area in the transp.jet.uk computer
============================================================

(Names followed by "/" are directories)
(The programs used to produce to Green's functions files are sometimes
 collectively referred to as "efund")

.*    (hidden files, some are links to efit/codesys/SetUp files)
REPO/ (CVS repository)
bin/  (contains shell scripts not under CVS)
own/  (on the side things, e.g. email)

efit/ (main directory)
  codesys/ (contains all code under CVS)
    SetUp/   (environment files e.g. .login, settrn-)
    doc/     (documentation)
      TODO
      dir.struct.doc (this file)
      efit.inputs.doc
      efit.notes.doc
      greenf.notes.doc
    csh/     (shell scripts)
      greenrun (script to run the Green's functions programs)
      efitdeblink (links to FORTRAN sources for debugging)
      efitrun  (script to run efit)
      testrun  (script to run the e--rgfc test programs)
    source/
      greenf/ (FORTRAN sources for Green's Functions - also makefile)
      testgf/ (FORTRAN sources for test of Green's Functions - also makefile)
      efit/   (FORTRAN sources for efit - also makefile)
  exe/ (contains executable binaries)
  log/ (contains temporary logs, e.g. from linking)
  work/
    makefile    (to produce Green's functions data output files)
    33/
      inputefun/  (input files for efund 33x33, renaming of "jac" efund96/input)
      outputfrx/  (output of e--frx4)
      outputfdn/  (output of e--fdn3)
      outputpdx/  (output of e--pdx2)
      outputgfc/  (output of e--gfc4)
      inputefit/  (input files for efit 33x33, renaming of "jac" efit96/input)
      _eftdir or <runid>_eftdir (work/output dir of efit 33x33 if run locally)
    65/
      inputefun/  (input files for efund 65x65, edited from 33)
      outputfrx/  (output of e--frx4)
      outputfdn/  (output of e--fdn3)
      outputpdx/  (output of e--pdx2)
      outputgfc/  (output of e--gfc4)
      inputefit/  (input files for efit 65x65, edited from 33)
      _eftdir or <runid>_eftdir (work/output dir of efit 65x65 if run locally)
Efit input directories, files and links ( -- = 33 or 65) (<- or -> = symlink)

pdx $HOME/efit/work/--/outputpdx <runid>_eftdir
--- ---------------------------- --------------
<       MHDIN   -> ~/efit/work/--/inputefun/#171195F
 >      MHDOUT
 >      ECONTO
 >      RFCOIL
 >      EPLASM ---------------------------------------- <-
 >      RECOIL ---------------------------------------- <-
 >      RVESEL  (in fact not created)
 >      RACOIL  (in fact not created)
 >      fort.6  (" >" in command line)
 >      err     ("2>" in command line)

frx $HOME/efit/work/--/outputfrx
--- ----------------------------
<       MHDIN   -> ~/efit/work/--/inputefun/#171195F
 >      MHDOUT
 >      FARADAY
 >      fort.6  (" >" in command line)
 >      err     ("2>" in command line)

fdn $HOME/efit/work/--/outputfdn
--- ----------------------------
<       MHDIN   -> ~/efit/work/--/inputefun/#TD3
```

```
 >      fort.16 (New. NO OPEN STATEMENT!,same var. name)
 >      ECONTO  (Finally Empty!)
 >      EPLASM  (Finally Empty!)
 >      RFCOIL  (Finally Empty!)
 >      RECOIL  (Finally Empty!)
 >      EDPLAS --------------------------------------- <-
 >      fort.6  (" >" in command line)
 >      err     ("2>" in command line)

gfc $HOME/efit/work/--/outputgfc
--- ---------------------------
<       MHDIN   -> ~/efit/work/--/inputefun/#171195F
<       ECONTO  -> ~/efit/work/--/outputpdx/ECONTO
<       RFCOIL  -> ~/efit/work/--/outputpdx/RFCOIL
<       FARADAY -> ~/efit/work/--/outputfrx/FARADAY
 >      ECONTO2 --------------------------------------- <-
 >      RFCOIL2 --------------------------------------- <-
 >      FARAD2  --------------------------------------- <-
 >      fort.6  (" >" in command line)
 >      err     ("2>" in command line)

efit $HOME/efit/work/--/inputefit
---- ---------------------------
<       ANGLE    -------------------------------------- <-
<       AUTO  ----------(or brazil)----------------- <- fort.5
<       CTURNS  -------------------------------------- <-
<       DPROBE  -------------------------------------- <-
<       EFITSNAP -------------------------------------- <-
<       FILIMT   -------------------------------------- <-
<       FITWT    -------------------------------------- <-
<       IRON     -------------------------------------- <-
<       PPFCURR  -------------------------------------- <-
<       SLICE    -------------------------------------- <- fort.17
<       SUBPART  -------------------------------------- <- fort.10
<       ZTEST    -------------------------------------- <-
<       fort.81  -------------------------------------- <-
<       fort.91  -------------------------------------- <-
<       t001     -------------------------------------- <-
t001.README
<       t002     -------------------------------------- <-
 NOTES FOR THE USER OF EFIT ON THE 'transp' Sun UNIX SYSTEM
 ==========================================================

 Note: the notation "--" in filenames is used to mean either 33 or 65,
 depending of the grid used.

 Green's described in separate note "greenf.notes.doc".



 The "home" uid for EFIT sources & input data is ~efit_jet.
 The "home" uid for TRANSP is any one of the TRANSP uids, e.g. ~pstubber,
 however at this moment only ~pstubber has been used.

 The EFIT executables are ~efit_jet/efit/exe/e--efit

 The TRANSP executable is ~pstubber/transp/work/JET/<runid>TR.EXE where
 <runid> is the run identifier, e.g. X716 (Uppercase where letters are used).
 I.e. it is in the ~pstubber's $WORKDIR/JET directory.

 Currently there is only one executable per grid size for EFIT while
 any TRANSP run has its own executable. This should be reviewed (esp. for
 the use of programs like ps).




 1) Running EFIT alone
 =====================

 1.a) Running EFIT in ~efit_jet

 Use the script efitrun (in ~efit_jet/efit/codesys/csh directory).

 Usage:
  efitrun  <function> <context> <mode> <grid> [<runid>]
  function: full   | init   | run   | clean
  context : alone  | pipes
  mode    : normal | debug
  grid    : 33     | 65
  runid   : shot number or any run identifier.

 1.a.1) nondebug mode

 While we work in efit_jet we seldom use a run or shot identifier, e.g. a
 normal run could be (we will always assume in this description a 33x33 grid),
 enter:
  efitrun full alone normal 33 [runid]
```

The other options are used e.g. for establishing the initial environment only,
for debugging, or for cleanup if the operation was incomplete.

The operation start with creating a subdirectory "_eftdir" (in whole
generality "<runid>_eftdir") in the directory ~efit_jet/efit/work/--.

As a next step, symbolic links are created in this subdirectory to the
input data files used by efit, i.e. files in directories

```
     ~efit_jet/efit/work/--/inputefit (nb: contains AUTO, t001.kb0, t001.kb1)
     ~efit_jet/efit/work/--/output(*) where (*) stands for pdx, fdn and gfc.
```

These (*) files have been produced by the Green's functions programs.

You might have to edit AUTO beforehand to point to the correct version of t001,
i.e. t001.kb0 for kbound=0 and t001.kb1 for kbound=1.

Note that kbound is also in EFITSNAP (directory --/inputefit).

Next step, the running, will invoke the executable with input from
fort.5 (which is a symbolic link to ~efit_jet/efit/work/--/inputefit/AUTO),
and output to fort.6 (which will be created).

Expect a CPU running time ~ 2 mins for the 33x33 grid.

The next and final step, the cleanup, will remove all the symlinks
from the _eftdir.

1.a.2) debug mode

Enter

```
 efitrun full alone debug 33 [runid]
```

There are differences with the normal mode:

- In debug mode, additionnal symbolic links are created inside _eftdir
  to the FORTRAN sources.

- The debugger will invoke the executable with terminal input & output,
  i.e. with fort.5 being ignored, stdin/stdout will come/go to the terminal.

A new xterm window appears and here is the proper dialogue you enter in when
back to the "(dbx)" prompt:

WARNING: in this first prompt enter "run" NOT "run something" !

```
|--------------------------------------------------
|(dbx) run
|Running: /home/tr2/efit_jet/efit/exe/e33efit
|
|          EFITD  33 x 33 Version  03/11/91
|
|
| type mode (2=file, 3=snap, 4=time, 5=input, 6=com file, -=laser):
|2
|
| i need  nft2: type of fit
|      npkg1: no of params
|        ene: where fit=0
| use 0 3 .01 for moment
|
| which ppf dda for output ? efit etc..
|xxx
|
| which uid for ppf output ?
|xxx
|
| are we writing a ppf 1=yes,0=no
|0
|
| number of time slices?
|1
|
| type input file names:
| #
|t001.kb0    (that is for kbound=0, for kbound=1 type t001.kb1)
|
...
(lots of output happen here)
...
|
|
|(dbx) quit
|--------------------------------------------------
```

```
1.b) running EFIT alone from a TRANSP uid ( for the moment ~pstubber).

This is a preparation for the run with TRANSP. TRANSP will be run
```

in $WORKDIR/JET. Note that a few TRANSP runs can take place at the same time because TRANSP identifiers for files (data and executables) do contain the runid. But efit has only (in the current set-up) one executable. We will run it into a directory of $WORKDIR/JET named with the run number. This directory will be named [<runid>]_eftdir.
(Often in tests it will be only _eftdir, however for the use with TRANSP there will always be a run number).

There is a symbolic link to the efit_jet script "efitrun" in the script directory of ~pstubber (referred to as ${SC}). Therefore efitrun is available.

1.b.1) nondebug mode

While in $WORKDIR/JET, enter:

```
 efitrun full alone normal 33 [runid]
```

This is similar to running in efit_jet, with these differences:

- The <runid>_eftdir is created inside the $WORKDIR/JET directory.

- The terminal input will come from $WORKDIR/JET/brazil and term output will go to [<runid>]_eftdir/fort.6, with error output to [<runid>]_eftdir/err. Edit brazil to ensure the use of the desired t001.kb0 or t001.kb1.

1.b.2) debug mode

While in $WORKDIR/JET, enter:

```
 efitrun full alone debug 33
```

For an example of debug terminal manual input see the previous case 1.a.2).

2) Running EFIT with TRANSP. (i.e. using pipes, not files)
============================

Preliminary remark:

The run of efit will be initiated by TRANSP (which can be seen as the master program). Therefore the user has no more the possibility to call the script efitrun, this will be done by TRANSP. The user has to select in advance the options with which to call efit, and set them in the system. This is done via the script setupefit in the TRANSP user script library. It HAS to be executed from the relevant tokamak subdirectory of $WORKDIR. Its usage is:

```
 setupefit <mode> <grid> <runid>
 mode    : normal | debug
 grid    : 33     | 65
 runid   : run identifier.
```

This will create two small files, <runid>.efitmode and <runid>.efitgrid
which will be used by transp (via the scripts efit_beg and efitgrunt) to
prepare (once) and initiate (often more than once) efit.

The scripts efit_beg and efitgrunt are in $WORKDIR/JET for the moment.
(This is not good practice & will need cleanup). They call efitrun with
the proper options. The preparation is similar to running without transp,
the <runid>_eftdir directory is cleaned or created and populated with
links to efit input files. Here it also create symlinks to pipes
which will be used when running with TRANSP.

Note: the current version of the pipes library has been modified so
that a copy of what goes to EFIT is copied into /tmp/junk. This provides
(at least for the moment) yet another method of spying on the run.

Two programs will be running, TRANSP and EFIT. Each can be independently
run in debug mode. A program running in debug mode receives a X-window xterm.
A program not in debug mode does not need a xterm.

NOTE THAT *TRANSP* WILL GET SOME INPUT FROM $WORKDIR/esnap2 and this can
include kbound!

We assume a JET run, that our runid is X716, and that we want a 33x33 grid.

2.a) Setting-up the options for EFIT.
Go to $WORKDIR/JET

2.a.1) Set-up for nondebug EFIT.
Enter:

```
 setupefit normal 33 X716
```

2.a.2) Set-up for debug EFIT.
Enter:

```
 setupefit debug 33 X716
```

2.b) Next, we want to run TRANSP - this one will call EFIT.

2.b.1) Running TRANSP in normal mode.
Assuming you have already run pretr, enter:

```
 runtr X716
```

```
Alternaltively, assuming that you already have a debug executable that
you want to use, albeit in non-debug mode, enter:
 X716db.dbx X716
You don't need a screen. You can spy on what happen via e.g.
 tail -f $WORKDIR/JET/X716tr.log

2.b.2) To debug run TRANSP itself, enter in $WORKDIR/JET:
 uplink X716db debug   ( to get a TRANSP debug executable )
and
 debug X716db.dbx

You current xterm becomes the TRANSP debug screen.
In this TRANSP debug screen, the start of the dialogue should be:
|--------------------------------------------------
|(dbx) run X716
|Running: X716db.dbx X716
|
| %TRANSP DEBUG RUN ID = X716
| OPTIONS:
|  C -- READ RUN CRASH DATA
|  R -- READ RESTART RECORD, RESTART RUN
|  S -- START RUN FROM VERY BEGINNING
|  Q -- QUIT
|
|MASTER_DEBUG:  ENTER CONTROL OPTION:
|R
|
|  READ RESTART RECORD   TA =  1.07000E+01
|
...
(lots of output happen here)
...
|
|Subroutine efitride called
|Hwrpip> writing : GO EFIT
|--------------------------------------------------

If you have also specified EFIT debugging, a pause will occur here
and you will have to enter data into an EFIT debug screen (see next
sections). When the control comes back proceed as for a normal
TRANSP debug session. You will have the possiblity
of specifying a flux surfaces plot.

2.c) The EFIT running.
```

```
 This will start when initiated by TRANSP.

 2.c.1) EFIT Nondebug running.
 Does not need a screen. You can spy on what happens via e.g.
  tail -f $WORKDIR/JET/X716_eftdir/fort.6

 2.c.2) EFIT Debug running
 As far as EFIT is concerned, debug will happen as before, except:
  i) the initial answer to the prompt "(dbx) " should be :
 |--------------------------------------------------
 |(dbx) run X716
 |--------------------------------------------------
  WARNING: THIS TIME NOT just "run" !

  ii) the answer to the prompt
 |--------------------------------------------------
 | type input file names:
 | #
 |--------------------------------------------------
  becomes irrelevant.

  iii) EFIT may be called more than one, producing each time a
  new xterm window. You will have to quit them all.


 2.d) End of run

 In all cases esp. if abnormal termination check for stray leftover
 processes and kill them.
 NB: a small script has been written (in ~pstubber/$WORKDIR/JET) "killit"
     which kill all processes with .PIP in their ps listing.
Notes on greenf

NAGINT, FLUX, HPINDX and SECOND have been made proper independently compiled
code nagint.f flux.f hpindx.f second.f

datim.f datimx.f errtra.f fileinf.f were unused and are also present in ../efit
sources. Removed from greenf. xuflow.f is also in ../efit. Removed from greenf.

Source still contains many commons defined in the code (instead of via include).

Probably makefile could be shortened using abbreviations.
At that moment we could maybe move the .o files to a specific directory $(OBJ).
```

```
From now on (20-May-1996) all data files names are UPPERCASE.

An useful example of a command line to retrieve I/Os:

egrep '(nin|NIN|nout|NOUT)' e--fdn3.f | \
egrep '(read|READ|write|WRITE|open|OPEN|close|CLOSE|=)'|more

Files I/O Output directory and files ( -- = 33 or 65)
===========
pdx $HOME/efit/work/--/outputpdx
--- ---------------------------
old nin 11 <  MHDIN -> ~/efit/work/--/inputefun/#171195F
new nout 10  > MHDOUT
new ncontr 35  > ECONTO
new nrspfc 26  > RFCOIL
new nrsppc 25  > EPLASM
new nrsppc 25  > RECOIL
new nrsppc 25  > RVESEL  (in fact not created)
new nrsppc 25  > RACOIL  (in fact not created)
new 6  > fort.6  (" >" in command line)
new -  >      err     ("2>" in command line)

frx $HOME/efit/work/--/outputfrx
--- ---------------------------
old nin 11 <  MHDIN -> ~/efit/work/--/inputefun/#171195F
new nout 10  > MHDOUT
new nffile 16  > FARADAY
new 6  > fort.6  (" >" in command line)
new -  >      err     ("2>" in command line)

fdn $HOME/efit/work/--/outputfdn
--- ---------------------------
old nin 5 <  MHDIN -> ~/efit/work/--/inputefun/#TD3
? nout 16  > fort.16 (New. NO OPEN STATEMENT!,same var. name)
old=0 nout 36  > ECONTO  (Finally Empty!)
old=0 nout 36  > EPLASM (Finally Empty!)
old=0 nout 36  > RFCOIL (Finally Empty!)
old=0 nout 36  > RECOIL (Finally Empty!)
old=0 nout 36  > EDPLAS
new 6  > fort.6  (" >" in command line)
new -  >      err     ("2>" in command line)

gfc $HOME/efit/work/--/outputgfc
--- ---------------------------
```

```
old nin 35 <  MHDIN -> ~/efit/work/--/inputefun/#171195F
old nin 35 <  ECONTO -> ~/efit/work/--/outputpdx/ECONTO
old nin 35 <  RFCOIL -> ~/efit/work/--/outputpdx/RFCOIL
old nin 35 <  FARADAY -> ~/efit/work/--/outputfrx/FARADAY
old=0 nout 36  > ECONTO2
old=0 nout 36  >     RFCOIL2
old=0 nout 36  >     FARAD2
new 6  > fort.6  (" >" in command line)
new -  >     err     ("2>" in command line)
```