The GATO Manual:
Everything You Ever Wanted to Know About GATO and a Helluva Lot More!

A.D. Turnbull

General Atomics, P.O. Box 85608,
San Diego, California 92186-9784

Abstract

The basic outline of the GATO ideal magnetohydrodynamic (MHD) stability code is described. GATO computes the linear ideal MHD stability in axisymmetric geometry using the Galerkin procedure to reduce the variational formulation [I.B. Bernstein *et al.*, Proc. Roy. Soc. A **244**, 17 (1958)] to a matrix eigenvalue problem where the eigenvalue is the square of the mode frequency and the eigenmode represents the plasma fluid displacement. The Galerkin expansion is in terms of finite hybrid elements (FHE) in the poloidal plane and a Fourier expansion in the toroidal direction. The structure of the code is explained, details explaining input options are given and instructions for compiling, loading, and running the code are provided. Special features of the code are described. This includes the convergence properties and the interpretation of the results, including the numerical destabilization of the continuum from the FHE method and it's correction. The code runs on a variety of platforms and can treat any axisymmetric toroidal geometry. It has been applied to the design and analysis of tokamaks, including DIII-D and ITER, spherical tori, spheromaks, and reversed field pinches.

1. Introduction

GATO is a linearized ideal magnetohydrodynamic (MHD) stability code based on the variational energy principle of Bernstein et al. [1]. Axisymmetric toroidal geometry is assumed. The code uses the Ritz-Galerkin method to convert the variational formulation into a matrix eigenvalue equation of the form:

$$AX = \lambda BX, \tag{1}$$

where the matrices A and B then represent the potential and kinetic energies of a displacement represented by the eigenvector X, and where the eigenvalue $\lambda = \omega^2$ is the square of the mode frequency; if all $\lambda > 0$ then ω is real and the system is stable, but if any eigenvalue $\lambda < 0$ then $\omega = i\gamma$ is imaginary and the system is unstable. γ is then the mode growth rate.

The Ritz-Galerkin expansion uses Fourier decomposition in the toroidal direction; axisymmetry then implies that the individual toroidal modes are decoupled and the code then solves for each toroidal harmonic independently. The expansion in the radial and poloidal directions uses finite hybrid elements (FHE) [2]. The FHE method usually numerically destabilizes the continuum [2] so that stability is actually the case even though, with no correction, the FHE method finds a mode with a negative eigenvalue whenever the real stable continuum extends down to marginal stability. It is usually clear if the mode found is a continuum mode, but some experience is required in figuring out the grey cases in between. Some helpful details are provided in Section 7. A numerical correction based on the analysis of the FHE approximation for localized modes is provided as an option and works extremely well. For use as a δW code, this correction is essential. The Galerkin procedure is outlined in Section 4.3 and the numerical correction procedure in Section 4.4. The following section describes the overall code structure and instructions for running it on various platforms. The sections following this describe details of the equilibrium mapping, the vacuum calculation and matrix construction, the eigenvalue solution, and output diagnostics.

The ERATO code on which the GATO code is based is described in R. Gruber *et al.*, Comput. Phys. Commun. **21**, 377, (1981). Mostly of interest here is the Finite Hybrid Element construction — the user interface for GATO is quite different. The specific features of the GATO code are described in L.C. Bernard, *et al.*, Comput. Phys. Commun.

21, 377 (1981). This has little detail but gives an overall view. The numerical correction to the destabilization is described in L. Degtyarev, *et al.*, Comput. Phys. Commun. **103**, 10 (1997).

2. Code Organization and Structure

2.1. Code structure

There are four sources called ***smap.f***, ***swnw.f***, ***seig.f***, and ***splt.f***. The first performs a mapping from the equilibrium on an (r,z) mesh to the flux coordinate mesh that GATO uses. The second (***swnw.f***) defines the wall, computes the vacuum contribution, and sets up the matrices for the eigenvalue problem (Ritz-Galerkin method). The third (***seig.f***) solves the eigenvalue problem for X and λ; as mentioned above λ is the square of the eigenfrequency and the system is unstable if λ is negative and the growth rate is then $[abs(\lambda)]^{1/2}$. The fourth package (***splt.f***) does most of the output analysis, including plots and constructing files containing various components of the displacement $\underline{\xi}$, perturbed field, δB, and perturbed vector potential δA, for interfacing with other codes.

The four packages ***smap.f***, ***swnw.f***, ***seig.f***, and ***splt.f*** are linked through binary disk files as shown in Fig. 1.

2.2. Input and Output

The input required is in three ASCII files. The equilibrium is assumed to be in a file called "eqgta". The code accepts both direct equilibria in the standard EFIT format specified by $\psi(r,z)$ and inverse equilibria specified by $r(\psi,\chi)$ and $z(\psi,\chi)$, where χ is a uniformly space equal-arc poloidal angle (TOQ format). For EFIT equilibria, the codes require a 'square' matrix for the flux on a uniform r,z mesh - $N_r \times N_z = 65 \times 65$ or 129×129, or 257×257, for example; $N_r = 2^n + 1$ is usual but not necessary. For TOQ inverse equilibria no such restriction exists on the size. However, the input equilibrium poloidal angle is currently restricted to a uniform equal arclength in that case.

The Namelist input in a file "ingta" controls the run. This generally requires little change from run to run. The third file called "inwgta" defines the wall position if needed. To run with a conformal or self-similar wall, a wall on the plasma surface, or a wall at infinity, however, this file is not required. Otherwise, this file is expected to contain a Namelist specifying either a set of wall positions $(r(i), i = 1,N, z(i), i = 1,N)$ of the wall or harmonic coefficients of an expansion in the wall positions. The option taken and number of points or coefficients N is specified in the "ingta" file. The "eqgta" equilibrium is read only by the mapping and the wall file "inwgta" is read and used only by the vacuum and wall

code ***swnw.f***. All four packages, however, read the same namelist input file. The following sections give brief descriptions of some of the key variables. A summary is also provided in Appendix A.

Each of the four packages produces an ASCII output file called "o*k*gta" ($k = 1,4$). These contain the basic run data reproduced from the standard output along with more detailed diagnostics, including more data in the event of an error. The amount of data provided can be controlled for each file using a corresponding parameter in the namelist input. Two default plotting options are available. Both are built on the TV80 package, which, in turn, is a driver for NCARGKS. The TV80 library is available with the GATO code. PGPLOT is a free package and a separate driver can be loaded which translates the TV80 calls in GATO to PGPLOT. With the TV80 option, ***smap.f*** and ***splt.f*** produce CGM files called "gato1.cgm" and "gato4.cgm", which can be translated to POSTSCRIPT files "gato1.ps" and "gato4.ps". The PGPLOT option produces the POSTSCRIPT files ("gato1.ps" and "gato4.ps") directly. In addition, the final code ***splt.f*** produces several comprehensive ASCII output files that can be used as input for other codes or for stand-alone graphics packages. A code also exists for reading one of these files and writing the data to the MDSplus database. Other files are produced during the run that are explained in the notes below. Some of these are useful for restarting the code after a crash and will be discussed later.

There are about a dozen Namelist input variables that are important to a run. These control the physical variables (toroidal mode number, compressibility, etc.), the mesh parameters, including the mesh sizes N_ψ and N_χ, mesh type, and symmetry and packing options, and specifications for the wall and vacuum.

The toroidal mode number is set by *ntor*. Compressibility or incompressibility is enforced by the variable *ncase*, with *ncase=0* the full compressibility case and *ncase=1* forcing incompressibility. The numerical destabilization correction is imposed by *ncorr = 0* (no correction) or *ncorr* $\neq 0$ with *corfac = 1.0* for various correction approximations. The two parameters *qxin* and *btdes*, if not zero, can be used to reset the vacuum toroidal field and scale the q profile so that either the axis q value, *QAXE,* is set to *qxin*, or the vacuum toroidal field at the nominal R_o, *BTOR*, is set to *btdes*. If both are zero, the equilibrium file input values of *QAXE* and *BTOR* are used; if both are nonzero, *qxin* takes precedence and *btdes* is ignored. The parameter *idnsty* controls the specification of the density profile used in the kinetic energy. For *idnsty = 0* a constant density, equal to $2\mu_0 BTOR^2$, so that the growth rate is automatically normalized to an Alfven time, is used.

For *idnsty > 0* the density is assumed to be read in from the input equilibrium file. For *idnsty < 0*, the density profile is constructed as a function of the poloidal flux *PSI* using the input namelist variables *ndnxp0*, *ndnxp1*, and *ndnxp2* as parameters. These can be used to encorce a density profile that vanishes at the axis, or the edge, or both. The parameter *gamma* is the ratio of specific heats, normally set at 5/3, but can be set to 1 for the isothermal case. Neither *idnsty* nor *gamma* affects the stability index - they only modify the growth rate. The parameter *qsurf* is only a diagnostic and does not modify the physical case; the code calculates the q_0 value required to obtain $q_{\lim} = qsurf$.

The parameter *ncase* controls the matrix pattern and storage, with *ncase = 0* allowing incompressibility of the plasma and *ncase = 1* forcing the code to ignore the toroidal component in the kinetic energy matrix and enforcing $\nabla \cdot \xi = 0$ in the potential energy. The δW norm that can be imposed as an alternative to the full kinetic energy norm is controlled using the parameter *norm*, with *norm = 0* for the full KE and *norm > 0* for the alternatives using only the normal displacement component. It should be noted that it is important to numerically restabilize the continuum modes by setting *ncorr = 1* when *norm > 0* to avoid serious spectral pollution.

The two parameters *ncase* and *norm* operate differently and interact in several ways. While *ncase* controls the actual matrix storage and does affect the resultant kinetic energy norm indirectly, *norm* controls the kinetic energy normalization without modifying the matrix pattern or storage. For consistency, it is recommended that one use *ncase = 0* and *norm = 0* to enforce true compressibility, or *ncase = 1* and *norm > 1* to enforce true incompressibility; if *ncase = 0* and *norm = 0* all components of the displacement are included in the kinetic energy norm and all are used in constructing the matrix, whereas if *ncase = 1* and *norm > 1* the equilibrium factors for the additional components in the kinetic energy normalization are not calculated and are consistently ignored in constructing the matrix. The other two possibilities are in principle inconsistent. If *ncase = 0* and *norm > 1* the full matrix is used but the kinetic energy normalization will only include the normal displacement. It is not clear what this corresponds to physically but the option exists. If *ncase = 1* and *norm = 0* the equilibrium factors for the additional components in the kinetic energy normalization are calculated and stored but not used in constructing the kinetic energy matrix and $\nabla \cdot \xi = 0$ is enforced in the potential energy. This should therefore revert to the incompressible case. However, this has never been fully tested so is not recommended.

Other physics parameters are *nmod*, *nlt*, and *nlim*. These parameters modify the physics problem in various ways, and set the ideal wall or line tying boundary conditions, and limiter boundary conditions respectively, but are rarely used. The option *nmod = 1* forces zero displacement for the region where q < 1 but does not work well. The parameter *nmod = 2* sets a floating boundary condition for the normal component of the displacement. This is used to check the growth rate against that for an infinite vacuum and requires *ivac = 0*. The parameter *nlim* $\neq 0$ sets a limiter boundary condition on the *nlim'th* poloidal angle and *nlt* $\neq 0$ sets line tying boundary conditions in the vacuum at the *nlt'th* poloidal angle and the surface where the pressure vanishes. However, this option works only with up-down symmetry. For ideal boundary conditions set *nlim = 0* and *nlt = 0*.

Up-down symmetry is imposed using *isym*; *isym = 0* for up-down asymmetry and *isym = 1* for a symmetric calculation, irrespective of the actual symmetry of the equilibrium. If the symmetry switch is set to *isym = 1* but the equilibrium is not explicitly symmetric, a warning is issued but the calculation is still performed. The two parameters *nmap* and *neqtyp* define the equilibrium type expected in the file "eqgta" – either a direct equilibrium (*nmap = 0*) or an inverse equilibrium of several types (*nmap* $\neq 0$) specified by *nmap* and *neqtyp*. Two parameters should be set to control the type of equilibrium treated – *ndoublt* and *ndivert*: To enforce Dee shape coding options set *ndoublt = 0* and set *ndoublt* = 1 to enforce a doublet equilibrium calculation. The parameter *ndivert* controls assumptions about the plasma edge in some diagnostic calculations. For a limiter plasma edge one should set *ndivert = 0*. However, for a diverted plasma edge one should set *ndivert = 1* to account for the logarithmic singularity in the calculation of the toroidal flux and plasma volume.

The basic coordinate system size is set by *jpsi* = N_ψ and *itht* = N_χ. For the flux mesh, the code actually does the mapping twice, once with a mesh equally spaced in ψ^α, where it finds the q profile. The parameter α is set with *cspak* = α, which is defaulted at $\alpha = 0.5$. Then the mapping is repeated with an automatically 'optimized' flux mesh with packing of surfaces at rational q and other specified surfaces. The packing is determined by a number of input Namelist parameters, *nmesh*, *cspak*, *pkfrac*, *qpfrac*, *nrat*, *npak*, *mpak*, *plpak*, *pspak*, *nedge*, *sedge0*, and *sedge1*. *nmesh* $\neq 0$ is the switch for invoked the grid repacking. With repacking, the default fractions of mesh points reserved for distributing equally in q or ψ^α or at rational surfaces is then controlled by *pkfrac*, *qpfrac*, and *nrat*. The default packing is generally optimal but can be changed relatively easily by using *npak* and *mpak* and the associated arrays *plpak* and *pspak* to add extra packing at

specified q or ψ^α surfaces, or by using *nedge = 2* and *sedge0* and *sedge1* non zero to add extra packing at the edge. These will be described in more detail in Section 3.

There are two choices for the poloidal angle determined by the input variable *igrid*. *igrid = 0* uses an equal arclength and *igrid = 1* uses the "PEST" straight field line coordinate [defined by Jacobian $= r^2 q/(rB_\phi)$]. The option *igrid = 0* is superior in almost all cases. The "PEST" coordinate is almost always numerically destabilizing. In some cases with large axis shift the destabilization is large. Also, the "PEST" coordinate is not valid in the case with a finite wall and vacuum. Note that the *igrid* parameter determines only the grid used for the calculation. The final analysis of the output can be done in the "PEST" poloidal angle, or even with a more general poloidal angle. It turns out that the "PEST" angle is almost always more suitable for the final mode analysis, especially for the Fourier analysis of the eigenmode. This option is chosen using *mshchi*, with *mshchi* = 3 for the "PEST" straight field line coordinate. A similar feature is available for the flux coordinate used for the final mode analysis; this can be done in several coordinates, set by the Namelist parameter *mshpsi*, independently of the physical poloidal flux coordinate used for the actual calculation.

To change the maximum dimensions for the input equilibrium in the sources, only the mapping ***smap.f*** needs to be modified. For direct equilibria the dimension for the input $\psi(r,z)$ is specified in a parameter statement:

parameter (nxx=129) or *parameter (nxx=65)*

This sets the maximum dimension for the array *PSI* as *PSI(nxx,nxx)*. To modify this one needs only to make a global change such as: "$nxx = 129$" $\rightarrow$ "$nxx = 65$". Note that the input array needs to be square with the two dimensions equal.

For inverse equilibria the dimensions for the input $r(\psi,\chi), z(\psi,\chi)$ and associated arrays are specified in a parameter statement:

*parameter (npp=513,ntt=2*npp-1)*

This sets the maximum dimension for the arrays *SEQRPS* and *SEQZPS* and associated arrays as *SEQRPS(npp.ntt)* and *SEQZPS(npp.ntt), etc*. To modify this one needs only to make a global change such as: "$npp = 513$" $\rightarrow$ "$npp = 1025$". Note that the input arrays do not need to be square. Also, one can always change the default "$ntt = 2 * npp - 1$" as needed; there are no real restrictions.

To change the maximum dimensions for the stability mesh in the sources, all four main sources ***smap.f, swnw.f, seig.f,*** and ***splt.f*** need to be modified. The dimensions for the needed arrays are specified in a parameter statement:

*parameter (npx=100,ncx=2*npx-1)*

To modify this, one needs only to make a global change such as: "$npx = 100$" → "$npx = 200$". Note that the input arrays for the stability mesh do not need to be square. Also, just as for the inverse equilibrium input mesh one can always change the default "$npx = 2 * npx - 1$" as needed. The dimensions for an input direct equilibrium are the only dimensions that have restrictions.

2.3. Compilation and running

Presently the code sources are set up with maximum mesh sizes N_ψ and N_χ. The sources are labeled as smaph*.xxx*.f, swnwh*.xxx*.f, seigh*.xxx*.f, and splth*.xxx*.f, where *xxx* is a label used to distinguish various versions. By default, *xxx* is the maximum number of flux surfaces, N_ψ, the source allows. Different versions are generally provided with different values for this.

The code can be complied in Fortran 90, 95, or Fortran 77. For Fortran 90 on UNIX, use:

set fort = "f90"
set cee = "cc" and
$fort -c -$options -o smaph*.xxx*.o smaph*.xxx*.f
$fort -c -$options -o swnwh*.xxx*.o swnwh*.xxx*.f
$fort -c -$options -o seigh*.xxx*.o seigh*.xxx*.f
$fort -c -$options -o splth*.xxx*.o splth*.xxx*.f

where options is a platform dependent, compilation switch. For most UNIX workstations (*e.g.* DEC ALPHA) use:

set options = "-O -64 -r8 -i8".

If the optimization fails it may be necessary to use instead "–Opt:Olimit=0 -64 -r8 -i8".

For LINUX use:

set fort = "lf95"
set cee = "gcc –O2" and
set options = "-O --dbl --long --tpp --prefetch 2"

A set of I/0 routines also needs to be compiled. Four versions are provided called dmroutines_*ppp*.y where

$$ppp.y = \begin{cases} cio.c \\ cray.f \\ fortran.f \\ fortalt.f \end{cases}$$

These are needed for swnwh.*xxx*.f and seig.*xxx*.f, and are compiled using

```
        $cee  -o dmroutines_cio.o  dmroutines_cio.c
or      $fort -o dmroutines_xxx.o  dmroutines_xxx.f
```

Only the first two routines work correctly and efficiently on the machines tested so far. The first, (cio.c) is a set of C routines that works well on all UNIX and LINUX machines. The second is only available on the NERSC CRAYs but even there, the first (cio.c) is faster. The last two are Fortran implementations that can be used in principle but have so far not worked well on any machine. However, the efficiency of these I/0 routines is, in principle, platform dependent and they should all be tested on any new platform.

Also required are plotting routines for smap.*xxx*.f and splth.*xxx*.f. On most of the GA workstations these are in a library TV80, which only needs to be loaded. Sources for these also exist which utilize NCAR graphics. Alternatively, there is a driver pgplot_driver.f, which translates the calls to PGPLOT routines. To compile this use

```
        $fort -o pgplot–driver.o   pgplot_driver.f
```

This, however, has a problem in that the usual implementation of PGPLOT on most machines is 32 bit whereas GATO requires 64 bit compilation. The PGPLOT driver converts integers to 32 bit but not real arrays and a 64 bit real PGPLOT library needs to be specially created and loaded. This is called pgplot-i4-r8.a in the following. Even then, the interface is currently poor.

Loading then takes the form:

On UNIX workstations with NCAR and TV80:

```
set fort          = "f90"
set load          = " "
set TV80_DIR  = "-L/d/tv80/libs/alpha_dux40"
set NCAR_DIR= "-L/d/osf/ncar/lib"
```

set X11_DIR = "-L/usr/bin/X11"
set TV80_LIB = "-ltv80"
set NCAR_LIB = "-lncarg -lncarg_gks -lncarg_c"
set X11_LIB = "-lX11"
set PLOTLIB = "$TV80_DIR $TV80_LIB $NCAR_DIR $NCAR_LIB $X11_DIR $X11_LIB"

On LINUX workstations with NCAR and TV80:
set fort = "f90"
set load = "--staticlink "
set TV80_DIR = "-L/d/tv80/libs/linux"
set NCAR_DIR = "-L/d/linux/ncarg-4.3.0/lib"
set X11_DIR = "-L/usr/X11R6/lib"
set TV80_LIB = "-ltv80"
set NCAR_LIB = "-lncarg -lncarg_gks -lncarg_c"
set X11_LIB = "-lX11 –ldl"
set PLOTLIB = "$TV80_DIR $TV80_LIB $NCAR_DIR $NCAR_LIB $X11_DIR $X11_LIB"

On UNIX workstations with PGPLOT, use instead of using:
"$TV80_DIR $TV80_LIB $NCAR_DIR $NCAR_LIB", use:
set PGPL_DIR = "-L/usr/local/pgplot-i4-r8 -L/usr/lib64"
set PGPL_LIB = "pgplot_driver.o -lpgplot"
set PLOTLIB = "$PGPL_DIR $PGPL_LIB $X11_DIR $X11_LIB"

On LINUX workstations with PGPLOT, use:
set PGPL_DIR = "-L/usr/local/pgplot-i4-r8"
set PGPL_LIB = " pgplot_driver.o -lpgplot"
set PLOTLIB = "$PGPL_DIR $PGPL_LIB $X11_DIR $X11_LIB"

$fort	-o smaph*.xxx*.e	smaph*.xxx*.o	$PLOTLIB
$fort	-o swnwh*.xxx*.e	swnwh*.xxx*.o	dmroutines_*ppp*.o
$fort	-o seigh*.xxx*.e	seigh*.xxx*.o	dmroutines_*ppp*.o
$fort	-o splith*.xxx*.e	splth*.xxx*.o	$PLOTLIB

There are four methods for running GATO. The separate codes can be run sequentially either using a set of scripts or by running from the FusionGrid using the interface provided. In using scripts, two different versions are available which differ only in the manner they manage the data files during a run. The code can also be run interactively, by running the four codes in sequence.

The FusionGrid option is adequate for most routine use and is described in: http://web.gat.com/comp/analysis/grid/gato/. The standard set of scripts is described on the GATO web page at http://fusion.gat.com/THEORY/gato/Gato_summary.html A set of makefiles is also provided that can be used to compile and load the code interactively.

For the other set of scripts that can be used to run GATO, there are two scripts that are described here. These are useful to start with. These were written for the NERSC CRAYs but are useful on most UNIX systems or any LINUX system since they allow multiple runs to be organized in different directories according to different projects and also allow the large temporary files to be placed on a separate scratch or temporary storage area. These will actually do all the compiling and loading if desired, as well as cleaning up files after a run. The first script is called runGATO_*job_label*.batch. The second is called runGATO_*platform*.script. The runGATO_*platform*.script file should not have to be changed at all except when porting to a new platform. The runGATO_*job_label*.batch script, however, has all the run-dependent variables in it. It is commented at the beginning and should be self-explanatory. These are set within the script in the form of statements:

set "option = value"

Usually only a few lines need to be changed from run to run. One of the options for example is *$scriptfl*, which should be set to "runGATO_*platform*.script" and rarely changed. Essentially the runGATO_*job_label*.batch script allows one to run sources from any directory, with input from another directory and to run in yet another directory. It sets up a unique subdirectory to run in to avoid overwriting previous output. The script takes any input names set in *$eqfile, $infile,* and *$inwfile* for the three input files needed to run GATO and copies them to three new files called "eqgta", "ingta", and "inwgta" while running and renames the copies as "inpt.eqgta", "inpt.ingta", and "inpt.inwgta", respectively, on completion. To run with a wall on the plasma or infinity, however, the "inwgta" file is not required and one should put "none" in the runGATO_*job_label*.batch script for the variable *$inwfile*. The script then runs the runGATO_*platform*.script specified as *$scriptfl* in runGATO_*job_label*.batch. In runGATO_*job_label*.batch, *$size* is the label used by the script to identify sources and executables as smaph"*$size*", etc (i.e. *$size* = *xxx* = "100" for example). Usually this is taken to be the maximum number of flux surfaces which must be at least the number of flux surfaces requested for the run (called "jpsi" in the code and "ingta" Namelist input file). Usually, $N_\chi = 2 \times N_\psi$ — also largely historical but convenient for doing studies where the mesh is progressively halved. In the sources, the maximum N_χ is set at twice the maximum N_ψ by default but

this is not a restriction. The sources with maximum N_ψ=*xxx* and maximum N_χ=2 × *xxx* will work for any smaller N_ψ and N_χ.

On completion of the run, the ASCII output files "o*k*gta" ($k = 1,4$) are also renamed as "outpt.out*k*". The output CGM graphics files (from TV80) "gato1.cgm" and "gato4.cgm" are renamed as "outpt.cgm1" and "outpt.cgm4" and the output POSTSCRIPT graphics files (from PGPLOT) "gato1.ps" and "gato4.ps" are renamed as "outpt.ps1" and "outpt.ps4". There are also corresponding postscript files "outpt.ps1" and "outpt.ps4" created by the script from "outpt.cgm1" and "outpt.cgm4" if the printer option in runGATO_*job_label*.batch is set to something other than "none".

The runGATO_*job_label*.batch script was originally developed for the CRAY J90 at NERSC to run either interactively or in batch using the QSUB batch facility. If the latter facility is available, one can use: "qsub runGATO_*job_label*.batch". The QSUB batch facility will then read the #QSUB statements at the top to set the batch control variables such as the shell (csh), time limit, memory, and queue priority. When running the script interactively as "runGATO_*job_label*.batch >& logfile", under either the c or t shell, the shell will ignore the QSUB directives as comments.

The code can also be run manually by running ***smap.f, swnw.f, seig.f,*** and ***splt.f*** sequentially with the "eqgta" file set up for ***smap.f*** , the "inwgta" file set up for ***swnw.f*** if needed, and and the "ingta" file set up for each. The ***splt.f*** code can also be rerun by itself to obtain a different set of plots by keeping the "cgta" and "tgta" files and setting up a new "ingta" file as needed.

2.4. Timing and Memory Requirements

The standard mesh size is $N_\psi \times N_\chi$=100 × 200. The 100×200 mesh cases with up-down asymmetry and full compressibility take about half an hour on the DIII-D cluster DEC ALPHA machines. On a single processor LINUX machine in the DIII-D cluster, this is reduced a factor three or more. But the timing on any single platform is variable and depends on how close the initial eigenvalue guess is. A mesh of 60×120 takes less than a minute on almost all machines with up to a half dozen eigenvalue guess iterations. With up-down symmetry the times are reduced by an additional factor four and with incompressibility set by *ncase = 1*, this is reduced by a further factor of two. An up-down symmetric, incompressible, 100×200 mesh case takes less than 30 seconds on the single processor LINUX cluster machines.

2.5. Benchmarks

Regarding the reliability of the results; the results agree very well with observations from DIII–D as well as with PEST, KNIX, DCON, MARS, and ERATO. But like any of these large codes, it has its 'inelegant' features. There are a few known code failure modes in the mapping with work-around solutions. Currently, the most common mapping failure is when the input direct equilibrium is either non-square, or has a different dimension from the parameter declaration. The code then aborts with an error. A second less common failure mode occurs in mapping the plasma boundary from a direct equilibrium with an X point, either on or near the boundary; the mapping sometimes fails to close the boundary. This is almost always correctly detected by the code and several attempts are made automatically to adjust parameters and recomputed the surface. The last resort adjustment is to set the boundary a small fraction of the poloidal flux inside the boundary flux provided by EFIT and continue incrementing this cutoff until a closed surface is successfully obtained. Occasionally this results in an excessive cutoff and some manual adjustment of the cut-off parameters is required. In addition, mapping failures can occur at internal flux surfaces as a result of a failure in the interpolation routines. These failures are usually detected and self corrected by the code and otherwise can be eliminated by minor changes to the default equilibrium mapping parameters. The most common example is in generating internal arclength grids with points unevenly spaced or spaced too close together. This is remedied by adjusting the parameters used to decide removal of the points in question. Also, interpolation of the q profile in the mesh packing routine can fail when q is rapidly changing. Generally, the code selfcorrects by resorting to linear interpolations and providing warnings in such cases.

The most common failure outside of the mapping is insufficient disk space to handle the large temporary files created during the inverse iterations for the eigenvalue. Typically, the consequence is machine dependent but under UNIX and LINUX the files are left empty and later read as zeros in the data leading to a floating-point overflow.

Several other failures can occur due to poor quality input. The most common of these is a failure in the vacuum calculation due to either an insufficiently smooth conducting wall or one that cuts inside the equilibrium plasma surface. In either case, unphysical negative eigenvalues appear in the nominally non-negative definite vacuum with zero inertia and hence become numerically large and negative. Occasionally the code fails to detect this condition before the eigenvalue solver is initiated.

Following are some descriptions of each of the four packages and some of the key Namelist input options corresponding to each package.

3. Mapping Code smap

3.1. GATO Equilibrium Quantities

Smap.f constructs the mapping from the input equilibrium to a flux mesh suitable for the stability calculation. The equilibrium quantities are defined on a (ψ,χ) mesh for the mesh cell centers as in ERATO. Up-down symmetry is not assumed unless set by *isym = 1* — the poloidal mesh extends over all 360 degrees. The zero poloidal angle is on the outboard midplane $(Z=Z_{axis})$ so that the first mesh point in the poloidal direction is the one just above the midplane. The last angle is the one just below the midplane. Figure 2 shows the overall mesh structure constructed in this manner.

The flux mesh is not equidistant. The values are defined in a variable *PSIVAL*(j) from the first cell formed from the magnetic axis and the next finite element node to the last value corresponding to the outermost cell bounded by the plasma surface. The *PSIVAL* values correspond to the midpoints of the cells. Note that ***smap.f*** stores *PSIVAL*(j) and the profiles and functions of *PSIVAL*(j) in reverse order from the plasma boundary, *PSILIM,* to the magnetic axis, *PSIMAX*. This is done since the mapping requires construction of the plasma boundary first. The other codes, however, store the flux quantities from *PSIMAX* to *PSILIM*. This is handled by the file interface between the codes; the arrays are written from ***smap.f*** to a file "egta" in the reverse order $(j = N_\psi, N_\psi - 1, ...2,1)$ from their storage order and read back in by swnw.f in forward order $(j = 1,2,...N_\psi - 1, N_\psi)$.

The poloidal angle has two choices — either equal arclength ("EQUALARC") or straight field line ("PEST") coordinate. The values are calculated on the mesh and given in the variables:

$$\begin{cases} CHIARCL & (j = 1, N_\psi, i = 1, N_\chi) \quad \text{EQUALARC} \\ CHIPEST & (j = 1, N_\psi, i = 1, N_\chi) \quad \text{PEST} \end{cases}$$

Both arrays are calculated and stored for each grid cell — one of them will be constant with respect to j. A restriction is that the "PEST" coordinate can only be used with a wall on the plasma boundary or a wall at infinity. There is also a third poloidal angle that is not used as a coordinate but can be used for final diagnostics and plotting. This is

defined by a Jacobian given by $J(\psi,\chi_H,\phi)=r^{nham1}B_p^{nham2}B^{nham3}$ where *nham1*, *nham2*, and *nham3* are set as Namelist parameters.

CHIHAML $(j=1,N_\psi,i=1,N_\chi)$ HAMILTONIAN

The (r,z) values on the mesh cell centers are given by the arrays $RCC(j=1,N_\psi,i=1,N_\chi)$ and $ZCC(j=1,N_\psi,i=1,N_\chi)$.

The other equilibrium quantities are given in arrays $FK(j=1,N_\psi,i=1,N_\chi)$ for $K=3,\ 4,\ 5,7,\ 8,\ 9,10,\ 11,\ 12,\ 13,\ 14,\ 20,22,23,\text{and }24$. These are listed below. Note that all units are MKS and no factors are put in implicitly. In these expressions,

$\left.\frac{\partial}{\partial\psi}\right|_\nu$ is the normal derivative: $\left.\frac{\partial}{\partial\psi}\right|_\nu=\left.\frac{\partial}{\partial\psi}\right|_\chi+\beta_\chi\left.\frac{\partial}{\partial\chi}\right|_\Psi$.

$$\partial_\psi(F)_\nu\equiv\frac{\nabla F\bullet\nabla\psi}{|\nabla\psi|^2}\equiv(\partial_\psi+\beta_\chi\partial_\chi)F\quad;\quad\beta_\chi\equiv\frac{\nabla\chi\bullet\nabla\psi}{|\nabla\psi|^2}\tag{2}$$

$\mu_0 rj_\phi=+(\mu_0 r^2p'(\psi)+ff'(\psi))=+\Delta^*\psi$ is the toroidal current density,
(3)

$$B_p=\frac{|\nabla\psi|}{r},\quad B_\phi=\frac{f(\psi)}{r},\text{ and}\tag{4}$$

$$J\equiv|\nabla\psi\times\nabla\chi\bullet\nabla\phi|^{-1}\text{ is the Jacobian.}\tag{5}$$

Note that the coordinate system is not orthogonal so $\beta_\chi\neq0$.

For the equal arclength poloidal angle $\chi_l\equiv l$:

$$J=L(\psi)\Big/2\pi B_p=\oint dl\Big/2\pi B_p\ ,\tag{6a}$$

$$\beta_\chi(\psi,l)=\frac{2\pi}{L(\psi)}\int^l\left[\frac{\mu_0 rj_\phi}{r^2B_p^2}-\frac{1}{B_p}\left(\frac{\partial B_p}{\partial\psi}\right)_\nu-\frac{1}{L}\frac{dL}{d\psi}\right]dl',\text{ and}\tag{6b}$$

$$\frac{1}{L}\frac{dL}{d\psi}=\frac{2\pi}{L(\psi)}\oint\left[\frac{\mu_0 rj_\phi}{r^2B_p^2}-\frac{1}{B_p}\left(\frac{\partial B_p}{\partial\psi}\right)_\nu\right]dl\tag{6c}$$

For the PEST straight field line poloidal angle χ_P ;

$$J = r^2 q \Big/ f = \frac{r^2}{2\pi} \oint \frac{dl}{r^2 B_p}, \tag{7a}$$

$$\beta_\chi(\psi, \chi_P) = \int^{\chi_P} \left[\frac{\mu_0 r j_\phi}{r^2 B_p^2} - \frac{1}{r^2 B_p^2} \left(\frac{\partial r^2 B_p^2}{\partial \psi} \right)_\nu - \left(\frac{1}{q} \frac{dq}{d\psi} - \frac{1}{f} \frac{df}{d\psi} \right) \right] d\chi'_P, \text{ and} \tag{7b}$$

$$\left[\frac{1}{q} \frac{dq}{d\psi} - \frac{1}{f} \frac{df}{d\psi} \right] = \frac{1}{2\pi} \oint \left[\frac{\mu_0 r j_\phi}{r^2 B_p^2} - \frac{1}{r^2 B_p^2} \left(\frac{\partial r^2 B_p^2}{\partial \psi} \right)_\nu \right] dl \tag{7c}$$

Then the FK are:

$$F3 = f(\psi)/r^2 \tag{8a}$$

$$F4 = -\mu_0 j_{tor}/(rB_p)^2 \tag{8b}$$

$$F5 = f(\psi) \, \partial \Big/ \partial\psi (J/r^2)_\chi \tag{8c}$$

$$\begin{aligned} F7 &= 2J \left[[(\mu_0 j_\phi)/(rB_p)]^2 + (\mu_0 j_\phi)/(r^2 B_p) \frac{\partial}{\partial \psi} \left(rB_p \right)_\nu - ((\mu_0 p'/r) \left(\frac{\partial r}{\partial \psi} \right)_\nu \right] \\ &= 2J \left[[(\mu_0 j_\phi)/(rB_p)]^2 + (\mu_0 j_\phi)/(rB_p) \frac{\partial}{\partial \psi} \left(B_p \right)_\nu + ((ff'/r^3) \left(\frac{\partial r}{\partial \psi} \right)_\nu \right] \end{aligned} \tag{8d}$$

$$F8 = 1/[J(rB_p)^2] \tag{8e}$$

$$F9 = JB_p^2 \tag{8f}$$

$$F10 = r^2/J \tag{8g}$$

$$F11 = \mu_0 \gamma p / J \tag{8h}$$

$$F12 = Jf/r^2 \tag{8i}$$

$$F13 = J \tag{8j}$$

$$F14 = \left(\frac{\partial J}{\partial \psi} \right)_\chi \tag{8k}$$

$$F20 = -\left(\frac{\partial \chi}{\partial \psi} \right)_\nu = -\frac{\nabla\psi \bullet \nabla\chi}{|\nabla\psi|^2} \tag{8l}$$

$$F22 = 2(r^2/f) \left(\frac{\partial \log(r)}{\partial \chi} \right)_\psi \tag{8m}$$

$$F23 = S = \left(\frac{B\times\nabla\psi}{|\nabla\psi|^2}\right)\bullet\left[\nabla\times\left(\frac{B\times\nabla\psi}{|\nabla\psi|^2}\right)\right]$$
$$= f'/r^2 - \left[\mu_0 r j_\phi \Big/ \left(r^2B_p^2\right) + \frac{\partial}{\partial\psi}\left(r^2B_p^2\right)_\nu \Big/ \left(r^2B_p^2\right)\right] \quad (8n)$$

$$F24 = \frac{\mu_0 j\bullet B}{|\nabla\psi|^2} - S \quad (8o)$$

In addition, several equilibrium flux surface averages are calculated and passed through for use in constructing the matrix and final diagnostics. These are:

$$S_0 = \oint\frac{dl}{B_p} \quad (9a)$$

$$S_1 = \oint\frac{dl}{r^2B_p} \quad (9b)$$

$$S_2 = \oint\left[\frac{\partial}{\partial\psi}\left(r^2B_p^2\right)_\nu/\left(r^4B_p^2\right)\right]\frac{dl}{B_p} \quad (9c)$$

$$S_3 = \oint\frac{dl}{(r^2B_p^2)B_p} \quad (9d)$$

$$S_4 = \oint\frac{dl}{(r^4B_p^2)B_p} \quad (9e)$$

$$S_q = \oint S\frac{dl}{B_p} \quad (9f)$$

$$S_T = \oint\left(\frac{\mu_0 j\bullet B}{|\nabla\psi|^2} - S\right)\frac{dl}{B_p} \quad (9g)$$

Two mapping options are provided. These are selected by the Namelist parameter *nmap*. For *nmap = 0*, the input equilibrium is assumed to consist of a direct equilibrium specified by a set of poloidal flux values $\psi(r,z)$ on a rectangular grid of $[(r_k,z_l),k=1,N_x,l=1,N_z]$ values with the flux functions $p(\psi), p'(\psi), f(\psi),$ and $ff'(\psi)$ specified on a uniform ψ grid from *PSIMAX* at the magnetic axis to *PSILIM* at the plasma surface. The format is that of the ASCII files from the EFIT code. For *nmap* $\neq$ *0,* the input equilibrium is assumed to consist of an inverse equilibrium specified by a set of $[(r(\psi,\chi),z(\psi,\chi)]$ defined on a mesh of $[(\psi_l,\chi_k), l=1,N_\psi,k=1,N_\chi]$ values. The flux functions $p(\psi), p'(\psi), f(\psi),$ and $ff'(\psi)$ in this case are specified on the ψ_l grid from *PSIMAX* to *PSILIM*. The default format is that for an equilibrium from the TOQ code

(*nmap = +1*). However, equilibria from the JSOLVER code can also be used (*nmap = -1*) in addition to TOQ equilibria in which all the *FK* functions are also provided on the mesh (*nmap = +2*). The mapping procedure is detailed in Appendix B.

3.2. GATO Input Equilibrium

Smap.f reads the equilibrium as either direct or inverse. In either case, the data is read from a file called "eqgta". For *nmap = 0* the file is assumed to be a direct equilibrium and for *nmap* $\neq 0$ it is assumed to be an inverse equilibrium of either the TOQ style (*nmap* > 0) or JSOLVER style (*nmap* < 0).

For direct equilibria a line containing a title, date, equilibrium type, and the dimensions is read first:

read(kueql,1000) (ETITL(it),it=1,NFT),DATE,IPESTG,NX, NZ

The title expects *NFT* = 5 words each of form character *8. The date is a single character *8. The dimensions N_r = *NX* and N_z =*NZ* are checked against the code dimensions N_{xx} and N_{zz} and against each other for a square input mesh; the dimensions must be equal to the code dimensions and $N_r = N_z$. The parameter *IPESTG* determines if the profiles for $p'(\psi)$ and $ff'(\psi)$ are to be derived from $p(\psi)$ and $f(\psi)$ (*IPESTG* = 1 or 2) or the input $p'(\psi)$ and $ff'(\psi)$ profiles (*IPESTG* > 2) and if the equilibrium is a fixed boundary with fake values in the vacuum (*IPESTG* = 4); in the latter case, linear interpolations on ψ are used if any of the collocation points lie in the vacuum. Normally this is not used and *IPESTG* is set to 3.

The equilibrium constants are then read in:

read(kueql,1010) XDIM,ZDIM,RCNT,REDGE

read(kueql,1010) XMA,ZMA,PSIMX,PSILIM,BTOR

read(kueql,1010) TOTCUR,PSIMX(1),PSIMX(2),XAX(1),XAX(2)

read(kueql,1010) ZAX(1),ZAX(2),PSISEP,XSEP,ZSEP

followed by the equilibrium profiles:

read(kueql,1010) (SF (kk), kk = 1,NPROFL)

read(kueql,1010) (SP (kk), kk = 1, NPROFL)

read(kueql,1010) (SFFP(kk), kk = 1, NPROFL)

read(kueql,1010) (SPP (kk), kk = 1, NPROFL)

and the poloidal flux:

read(kueql,1010) ((PSI(ii,jj), ii = 1,NX), jj = 1,NZ)

The quantities *XDIM* and *ZDIM* specify the r and z dimensions of the box over which the mesh is defined. *RCNT* is a nominal major radius and *REDGE* is the inner edge of the r mesh. Then the uniform rectangular mesh on which psi is defined runs from:

X(1) = REDGE to *X(NX) = REDGE + XDIM/(NX-1)* and

*Z(1) = -0.5*ZDIM/(NZ-1)* to *Z(NZ) = +0.5*ZDIM/(NZ-1)*.

XMA and *ZMA* are the magnetic axis, *PSIMX* and *PSILIM,* are the poloidal flux at the magnetic axis and plasma boundary, and *BTOR* and *TOTCUR* are the toroidal vacuum field (T) at *RCNT* and the total current (MA) respectively. The variables *PSIMX(1),PSIMX(2),XAX(1),XAX(2), ZAX(1),ZAX(2),PSISEP,XSEP* and *ZSEP* are all used to specify a Doublet equilibrium with two magnetic axes specified by (ψ, r, z) from (*PSIMX(1),XAX(1),ZAX(1)*) and (*PSIMX(2),XAX(2),ZAX(2)*), and an X-point with (*PSISEP,XSEP,ZSEP*). For a standard simply connected cross-section, *PSIMX(1) = PSIMX*, *XAX(1) = XMA,* and *ZAX(1) = ZMA*, the second axis is set to zero (*PSIMX(1) = 0.0*, *XAX(1) = 0.0,* and *ZAX(1) = 0.0*), and the X-point values are set at *PSISEP = PSILIM*, *XSEP = 0.0,* and *ZSEP = 0.0*. Setting *PSISEP = PSILIM* will automatically force the second axis and X-point to be ignored.

The profiles $f(\psi) = SF$, $p(\psi) = SP$, $ff'(\psi) = SFFP$, and $p'(\psi) = SPP$, are taken on a uniform *PSI* mesh between *PSIMX* and *PSILIM*, with $NPROFL = NX = N_r$.

If *idnsty > 0* is set in the namelist input, additional data is read sequentially from the EQGTA file in order to obtain the mass density profile. This consists of:

the equilibrium q profile

read(kueql,1010) (EFITQ(kk), kk = 1, NPROFL)

and the boundary and limiter dimensions and data

read(kueql,1020) NEFTBD,NEFTLM

read(kueql,1010) (EFITBDY(ll),ll = 1,NEFBDY)

read(kueql,1010) (EFITLIM(ll),ll = 1,NEFLIM)

where *NEFBDY = 2* NEFTBD < NBD* and *NEFLIM = 2* NEFTLM < NBL* with *NBD* and *NBL* set in parameter statements. The inequalities are checked on input. The data *EFITBDY(ll)* and *EFITLIM(ll)*, consist of pairs of (r,z) points describing the location of the

plasma boundary and the vessel limiter respectively for *NEFTBD* and *NEFTLM* points respectively.

For *idnsty = +1,* the density profile is then read directly:

read(kueql,1010) (SDNS(kk), kk = 1, NPROFL)

For *idnsty > +1*, some additional rotation data *KVTOR,* and *RVTOR* and density profile fitting data *NMASS* are read first:

read(kueql,1030) KVTOR,RVTOR,NMASS

followed by the density profile if *NMASS > 1* (if *NMASS < 2* it is assumed no fitting was done and no profile is present).

read(kueql,1010) (SDNS(kk), kk = 1, NPROFL)

All but the density profile data is ignored. The mass density is assumed to be given in Kg/m^3. Both the safety factor and density profile are taken to be on the same uniform psi mesh as the other profiles.

The formats for each of the reads are:

1000 format(6a8,3i4) for the title and date

1010 format(5e16.9) for the equilibrium constants, profiles, and arrays

1020 format(2(1x,i4)) for the boundary and limiter dimensions

1030 format(i5,e16.9,i5) for the rotation and density fitting data.

For inverse equilibria from TOQ, two options dependent on the Namelist parameter *neqtyp* exist. For *neqtyp = 0*, a date and title are read first:

read (kueql,1000) DATE

read (kueql,1000) (ETITL(it),it=1,NFT)

followed by the dimensions and symmetry option:

read (kueql,1010) NPSI,NTHT,NEQSYM

Otherwise, if *neqtyp ≠ 0*, a default heading and date are prescribed and only the dimensions *NPSI* and *NTHT* are read:

read (kueql,1020) *NPSI,NTHT*

with *neqsym* set at *neqsym = 1* for up-down symmetry; *neqsym = 0* assumes both halves of the equilibrium are present in the file but this is not normally the case when the date and title are not present.

The code then sets the full dimension *NTH2* - for *neqsym = 0* then *NTH2 = NTHT* and for *neqsym = 0* then *NTH2 = 2*(NTHT-1) + 1* - and checks for incorrect dimensions such as *NPSI < 1*, *NPSI > npp*, *NTH2 < 1*, or *NTH2 > ntt*

The equilibrium scalar data is then read in as:

read (kueql,2000) RCNT,XMA,ZMA,BTOR

followed by the total current *TOTCUR*, axis elongation *AXDDXZ*, and mass density normalization *DNNORM*

read (kueql,2000) TOTCUR,AXDDXZ *(neqtyp = 0)*

read (kueql,2000) TOTCUR,AXDDXZ,DNNORM *(neqtyp ≠ 0)*

For *neqtyp = 0*, *DNNORM* is defaulted to 1.0. In the most recent versions, *AXDDXZ* is recalculated and the input ignored.

The profiles for are then read:

read (kueql,2000) (PSIMSH (jj), jj = 1,NPSI)

read (kueql,2000) (SF (jj), jj = 1,NPSI)

read (kueql,2000) (SFFP (jj), jj = 1,NPSI)

read (kueql,2000) (SP (jj), jj = 1,NPSI)

read (kueql,2000) (SPP (jj), jj = 1,NPSI)

read (kueql,2000) (SQVL (jj), jj = 1,NPSI)

read (kueql,2000) (SDNS (jj), jj = 1,NPSI)

Here, *PSIMSH* is the *PSI* mesh and can be nonuniformly spaced and *SQVL* is the safety factor profile on this mesh. This is splined to the stability mesh and is also recalculated by the mapping on the stability mesh and the two compared; however, the recalculated version is actually used.

The derivatives $\partial\psi/\partial r$ and $\partial\psi/\partial z$ of the poloidal flux around the boundary are then read:

read (kueql,2000) (SEQDPDR(ii), ii = 1,NTHT)

read (kueql,2000) (SEQDPDZ(ii), ii = 1,NTHT)

followed by the full arrays of the inverse equilibrium $r(\psi,\theta)$, and $z(\psi,\theta)$:

read (kueql,2000) ((SEQRPS(jj,ii), jj = 1,NPSI), ii = 1,NTHT)

read (kueql,2000) ((SEQZPS(jj,ii), jj = 1,NPSI), ii = 1,NTHT)

An option also exists with *nmap = 2* to read the mapping data from the equilibrium file and interpolate to the stability mesh. This data consists of the actual arclength array for the mesh, *SEQARC*, followed by the full *FK* arrays:

read (kueql,2000) ((SEQARC(jj,ii), jj = 1,NPSI), ii = 1,NTHT)

read (kueql,2000) ((F3 (jj,ii), jj = 1,NPSI), ii = 1,NTHT)

read (kueql,2000) ((F4 (jj,ii), jj = 1,NPSI), ii = 1,NTHT)

…..

read (kueql,2000) ((F22 (jj,ii), jj = 1,NPSI), ii = 1,NTHT)

However, the *FK* up to *F22* only are expected; *F23* and *F24* as well as the surface integrals *SK* are not read or calculated. These are used only for the numerical correction (see Section 4) so this option is not then available.

The formats for the TOQ inverse equilibrium data are:

1000 format(6a8) for the title and date

1010 format(3i5) for the dimensions (*neqtyp = 0)*

1020 format(2i5) for the dimensions (*neqtyp ≠ 0)*

2000 format(1p4e19.12) for the equilibrium constants, profiles, and real arrays

Inverse equilibria in the form of PPPL "U-files" from JSOLVER can be read by setting *nmap < 0*:

First, the inverse equilibrium dimension data is read in and checked for consistency:

read (kueql,1000) NTHD1,NPSD1,NTHT,NPSI,NEQSYM,DLR,DLT

These files contain array data up to the full dimensions *NPSD1* and *NTHD1* set in the equilibrium calculation, as well as ghost points in the theta direction to handle periodicity. The input array data is then adjusted to exclude data between *NPSI* and *NPSD1* and between *NTHT* and *NTHD1* and shifted two points poloidally. The input *NPSI* and *NTHT* are checked against the available storage *npp* and *ntt* as in the TOQ equilibrium case, though taking account of the shift in this case. The dimensions *NPSD1* and *NTHD1* are also checked against the dimensions *npp* and *ntt* and if too large, the excess values beyond *NPSI* and *NTHT* are read to a dummy variable and discarded; typically *NPSD1* and *NTHD1* are very large and do exceed these dimensions. As in the TOQ equilibrium case, the actual theta dimension *NTH2* is set according to up-down symmetry *NEQSYM*. *DLR* and *DLT* are ignored.

The equilibrium scalar data is read in as:

read (kueql,1100) RCNT,PRNORM,PSILIMP,PSIMINP

Here, *PSILIMP = PSILIM* and *PSIMINP = PSIMIN*. *PRNORM* is the pressure on axis. The profiles are then read up to the full input dimension written if the available dimension is sufficient:

read (kueql,2000) (SP (jj), jj = 1,NPSD1)

read (kueql,2000) (SPP (jj), jj = 1,NPSD1)

read (kueql,2000) (SQVL (jj), jj = 1,NPSD1)

read (kueql,2000) (SQLVP (jj), jj = 1,NPSD1)

read (kueql,2000) (SF (jj), jj = 1,NPSD1)

read (kueql,2000) (SFFP (jj), jj = 1,NPSD1)

read (kueql,2000) (SG1 (jj), jj = 1,NPSD1)

read (kueql,2000) (SG2 (jj), jj = 1,NPSD1)

read (kueql,2000) (SG3 (jj), jj = 1,NPSD1)

read (kueql,2000) (SG4 (jj), jj = 1,NPSD1)

read (kueql,2000) (PSIMSH (jj), jj = 1,NPSD1)

If the dimension is insufficient each profile input line is replaced by the construction:

read (kueql,2000) (SQUANT (jj), jj = 1,NPPD) ,(SDUMMY , kk = 1,NREMP)

where *NPPD = npp* and *NREMP = NPSD1 – npp*. The arrays *SG1*, *SG2 SG3*, and *SG4* and *SQLVP*, are ignored, except for *SG1*(1) which is used to define *SF*(1) on axis. As for the TOQ equilibria, the safety factor profile *SQVL* is interpolated on to the stability mesh and compared against the calculated values from the mapping.

The inverse equilibrium $r(\psi,\theta)$, and $z(\psi,\theta)$ are read as:

read (kueql,2000) ((SEQRPS(jj,ii), ii = 1, NTHD1), jj = 1, NPSD1)

read (kueql,2000) ((SEQZPS(jj,ii), ii = 1, NTHD1), jj = 1, NPSD1)

if the available dimensions are sufficient. Otherwise, the excess data when exceeds is read into a dummy variable as for the profile arrays. In this case, both the *PSI* and poloidal dimensions are accounted for. In addition, for nmap = -2, the full Jacobian arrays are read in the same manner, though are never used:

read (kueql,2000) ((SEQAJ3(jj,ii), ii = 1, NTHD1), jj = 1, NPSD1)

read (kueql,2000) ((SEQAJ0(jj,ii), ii = 1, NTHD1), jj = 1, NPSD1)

Finally adjustments are made to the units and profiles for this input option. The input density is set constant and a default title and date are set. A scale factor *BFIELD* is applied to all magnetic fiels and poloidal flux variables. This is set to the Namelist input *bfieldf* if *bfieldf* $\neq 0$, or defaulted to 1.0 if *bfieldf* = *0*. Also, the pressure is and its derivative are rescaled by the factor $\mu_0/BFIELD^2$. The pressure and toroidal field arrays *SP* and *SF* are shifted from the input defined half cells to the same mesh as the other flux arrays with *SP*(1) set to *PRNORM* and *SF*(1) = *SG1*(1)**SQVL*(1).

The formats for the JSOLVER inverse equilibrium data are:

1000 format(5(1x,i4),2(1x,e11.4)) for the dimensions

1100 format(4e16.8) for the equilibrium constants

2000 format(5e16.8) for the profiles, and real arrays

3.3. Fitting Options

Several Namelist input parameters control the way the mapping of flux surfaces is done. The most common failure in mapping surfaces is the tracing of the plasma boundary. The parameter *dpsisl* moves the plasma surface by the corresponding fraction on the initial attempt at mapping the boundary; *dpsisl* < *0.0* moves the boundary inside the plasma. When this fails, subsequent attempts decrement *dpsisl* using the variable *dpsisd* by an amount *dpsisd*(2**(k-1))* where *k* represents the number of subsequent attempts. The maximum number of plasma boundary mapping attempts is set by *mapmaxd*. The details of the surface mapping procedure are described in Appendix B.

The calculation of the safety factor on axis, q_0, is performed two different ways for both the direct and inverse equilibrium cases. These should not differ much but small changes in q_0 can cause sensitivity when q_0 is near a rational value and the change results in a small dip or bump in q over the first few points. The Namelist parameter *nqaxis* can be used to force use of axis fit from either method, or to use a weighted average. This is particularly useful when the equilibriahave convergence problems near the axis – often one or the other calculation is then clearly poor. For *nqaxis = 0*, the code uses a default choice based on various subjective criteria of which estimate is most consistent with the neighboring values. For *nqaxis* $\neq 0$, one or the other estimate is used depending on the sign and with weight depending on the magnitude of *nqaxis*. The details depend on the equilibrium type.

The maximum number of arclength points used to define the mapped flux surface is specified by the Namelist input variable *narcmx*. This must be set less than the parameter *nlx*. *narcmx* determines the fine surface mesh used for interpolation, differentiation, and integration along the surface. When the surface mapping fails to produce a set of (r,z) points with increasing arclength the mapping is reattempted decrementing the number of points by *2*ntdecr* per failure. This is continued until either successful, or the number of points is reduced below ntmmin, or the number of attempts reaches the maximum *ntrymx*.

Several tolerances in the fitting can be set. The most important for the general mapping procedure are *qptol*, which specifies the tolerance between interpolated input and calculated values of q in the case of inverse equilibria, and *tolspln*, specifying the allowed tolerance in spline calculations of surface *(r,z)* values. When set at their standard defaults, these will flag serious equilibrium problems since they check key equilibrium quantities calculated in quite different ways. In addition, the code checks the convergence of the input equilibrium with estimates of the quality of global averaged convergence, convergence averaged over approximate surfaces, and pointwise convergence. This is done separately for the plasma and vacuum regions in the case of direct equilibria. The tolerances for flagging excessive convergence errors are *delac* and *delav* for the global error criteria for the plasma and vacuum respectively, *delstsf* for the surface average error criterion in the plasma, and *delstlp* and *delstlv* for the pointwise error criteria for plasma and vacuum respectively. The code reports when the errors exceed these input tolerances. However, due to the large potential number of pointwise error flags, the maximum number of poor convergence error reports in plasma and vacuum are set by *maxerlp* and *maxerlv* individually.

3.4. GATO Mesh Packing Options

A great deal of flexibility is provided in GATO to control the packing of flux surfaces. The packing is adjusted by the Namelist parameters, *nmesh*, *cspak*, *pkfrac*, *qpfrac*, *nrat*, *npak*, *mpak*, *plpak*, *pspak*, *nedge*, *sedge0*, and *sedge1*. The initial ψ mesh is constructed as an equally spaced mesh in the variable:

$$s = \left(\frac{(\psi - \psi_0)}{(\psi_1 - \psi_0)} \right)^{\alpha}, \text{with } \psi \equiv PSIVAL, \psi_0 \equiv PSIMAX, \psi_l \equiv PSILIM, \text{and } \alpha \equiv cspak. \qquad (9)$$

cspak is defaulted at $\alpha = 0.5$, which makes *cspak* an approximate radial minor radius variable. For *nmesh = 0*, this is the mesh used in the stability calculation. For *nmesh < 0* the mesh for ψ is read in from an unformatted file "mgta" from a previous mapping run,

or from a formatted table "ngta". This option can be used to ensure that two stability calculations utilize exactly the same mesh. The unformatted file "mgta" can be the "egta" file produced by smap.f or the "tgta" file produced by swnw.f. Several options exist for the formatted file "ngta" that are invoked by taking different negative values for *nmesh*.

nmesh	**Effect**
-1	Read *PSIVAL* from the unformatted file "mgta", where *PSIVAL* was written from *1* to *jpsi1* for use in swnw.f by a previous mapping and was written to include *PSILIM* but not *PSIMAX*
-2	Read *PSIVAL* from a formatted file "ngta" from *PSIMAX* to *PSILIM*, as a string of formatted values
-3	Read *PSIVAL* from a formatted file "ngta" from *PSILIM* to *PSIMAX*, as a string of formatted values
-4	Read *PSIVAL* from a formatted file "ngta" from *PSIMAX* to *PSILIM*, as one value per line
-5	Read *PSIVAL* from a formatted file "ngta" from *PSILIM* to *PSIMAX*, as one value per line

The option *nmesh > 0* is the switch for invoking the grid repacking. This is done in the subroutine *MESHPAK*. This routine sets up a non-equidistant ψ mesh with points automatically concentrated around the rational surfaces and optionally at up to *npkmax* specified locations. The default fractions of mesh points reserved for distributing equally in q or ψ^{α} or at rational surfaces is then controlled by the Namelist parameters *pkfrac*, *qpfrac*, and *nrat*. The points are arranged such that a fraction *pkfrac* are placed around the existing rational and the prescribed surfaces, a fraction *qpfrac* are equally spaced in q, and the remainder are set equally spaced in s, with s stored in the variable *CS*. If *pkfrac* $<$ *0.0*, the default rational surfaces are ignored and a fraction abs(*pkfrac*) points are packed around only the prescribed surfaces. Also, if $|pkfrac| + |qpfrac| > 1$ then $qpfrac = 1 - |pkfrac|$ is set and no contribution from equal spacing in s is done. If the number of rational and prescribed packing surfaces exceeds *nratmx*, the outermost ones are deleted except for the edge if *nedge* $\neq 0$.

The surfaces are packed around each rational or prescribed location according to a Gaussian density distribution with default width proportional to the shear. For rational surfaces, prescribed q or nq surfaces, and for prescribed flux surfaces, ψ^{α} or s, given by *pspak*), the global shear is used so that packing can be forced in shearless regions. For

the edge, the shear used is set by the choice of *nedge* as will be described below. The default number of surfaces within each packing location is the same for each surface, and is determined by $\frac{pkfrac}{N_p}$, where N_p is the total number of packing locations. The plasma edge is included as a packing surface as if it were prescribed by *pspak*.

For rational surfaces and packing surfaces prescribed by *plpak* the inverse width is limited by the two parameters *swidmn* and *swidmx*. For surfaces prescribed by *pspak*, these limits are ignored so that the limits can always be overridden if needed by using *pspak* instead of *plpak*. For the edge, the inverse width is limited only by *swidmx*.
The distribution of points is calculated by defining a weight distribution, $W(s)$ over the s domain, denoted in the code as *WGHT*(*SWGT*), which is the fraction of surfaces *CS(JP)* with *CS(JP)* < *SWGT*. Then the fraction of the flux surfaces in the final packed mesh placed within the interval from $s-\delta$ to $s+\delta$ is $\delta\left(\frac{dW}{ds}\right)_{SWGT}$.
The weight function W = *WGHT* is calculated on a fine non-equidistant numerically constructed mesh 0.0 ≤ *SWGT* ≤ 1.0, with points strategically concentrated around the surfaces that are to be packed - i.e. where *WGHT* varies most rapidly. This mesh is constructed so that there are $KPST + C_0\left(\frac{jpsi}{NRATNL}\right)KPAK$ points around each packing surface and $KPST + (1-C_0)\,jpsi\left|\frac{q'(\psi_{\lim}-\psi_{\min})}{q_{edge}-q_{axis}}\right|KPAK$ points between each pair of packing surfaces. C_0 is set from *pkfrak* and *KPST* and *KPAK* are set from the input Namelist variables *minpak*, *maxpak*, and *incpak*. *KPST* ≠ *0* ensures there are always a minimum number of points both between and around each packing surface so that the function *WGHT(SWGT)* always has sufficient resolution.

The default packing set in the code is generally optimal but can be changed. Additional packing at specified q or ψ values can be enforced using *npak* and *mpak* and the associated arrays *plpak* and *pspak*. The parameters *npak* and *mpak* control the manner in which the optional packing is done:

npak	**Number packed**	**Packing Location**		***mpak***	**Number packed**	**Packing Location**

	points				**points**	
0	0	No packing		0	0	No packing
> 0	*npak*	Prescribed *q* values		> 0	*mpak*	Prescribed ψ values
< 0	$\lvert npak \rvert$	Prescribed *nq* values		< 0	$\lvert mpak \rvert$	Prescribed *s* values

The optional packing q (or nq) locations are prescribed by the array *plpak(k1,k2)* and the optional prescribed ψ (or s) locations by the array *pspak(k1,k2)*. For packing at prescribed packing q or nq locations, $plpak(1,k2)$ gives the location for packing at $k = 1,2,..|npak|$ points. Then the factor by which the default width of the packed distribution is modified is prescribed by $(1 + plpak(2,k2))$ and the factor by which the number of points placed is to be modified from the default is prescribed by $(1 + plpak(3,k2))$. Similarly, for packing at prescribed packing ψ or s locations; $pspak(1,k2)$ gives the location for packing at $k = 1,2,..|mpak|$ points. Then the factor by which the default width of the packed distribution is modified is prescribed by $(1 + pspak(2,k2))$ and the factor by which the number of points placed is to be modified from the default is prescribed by $(1 + pspak(3,k2))$.

Extra packing at the edge is also obtained by using $nedge \neq 2$ and setting *sedge0* and *sedge1* nonzero. The parameter *nedge* affects both the search procedure for rational surfaces near the edge over interval between *CS(jpsi)* and *CS(jpsi+1)* of the initial mesh, and the packing weight at the edge. If $nedge = 0$, no packing is done at the edge, and the search for rational surfaces is not done over the last interval. If $nedge > 0$, additional packing is done at the edge, and the search for rational surfaces is performed right to the edge. If $nedge < 0$, no packing is done at the edge, but the search for rational surfaces is still performed right to the edge over the last interval.

nedge	**Searching between *CS(jpsi)* and *CS(jpsi+1)***	**Additional Edge Packing**	**Edge Packing Width**
< 0	Done	None	None
0	Not done	None	None

1	Done	Edge included as a rational surface only if last rational q has JPLPAK < *jpsi+1*	Global shear
2	Done	Edge included as a rational surface even if already included since last rational q has JPLPAK = *jpsi+1*	Global Shear
3	Done	Edge included as a rational surface only if last rational q has JPLPAK < *jpsi+1*	Local shear
4	Done	Edge included as a rational surface even if already included since last rational q has JPLPAK = *jpsi+1*	Local shear

Other parameters *nrat* and *nrepeat* modify the packing further. If *nrat* is specified as nonzero, then *pkfrac* is multiplied by *nrat/jpsi* and *qpfrac* by a factor *(1 - nrat/jpsi)*. If *pkfrac = 0.0* and *nrat* is specified then *pkfrac* is taken to be just *nrat/jpsi*. The parameter *nrepeat* determines the packing when multiple rational surfaces of the same helicity are present because the shear is reversed in part of the cross section. If *nrepeat = 0* then all occurrences of a rational q surface specified in *plpak* are packed. If *nrepeat > 0* then only the *nrepeat*'th occurrence of the q surface is packed, whereas if *nrepeat < 0* the *nrepeat*'th occurrence of the q surface is skipped and all the others are packed. Two additional parameters, *nppack* and *nqpack*, can be used to modify the packing distribution for equilibria with low or negative shear. If *nppack* $\neq 0$ the weights of the default rational q packing points (those not set by *plpak* and not including the edge) are modified to eliminate packing for points in the negative shear region (*nppack > 0*) or those in the positive shear region (*nppack < 0*). If *nqpack* $\neq 0$ the weights of the distribution in q are modified to be evenly distributed in $s^{nqpack}q$.

3.5. GATO Mapping Plot Options

Several plots are produced from the mapping routine ***smap.f***. These show the various equilibrium profiles as well as diagnostics for the input equilibrium errors and the mesh packing. Of these, the most useful equilibrium plot is the q profile plot and to a lesser extent the line plots for ff', p', p, j versus r across the midplane and ff' and p', versus Ψ. These, however, are essentially read from the input. In terms of diagnostics for the mapping process itself, the error map and the packing distribution are generally most useful.

To control the plots use the namelist variable *iplotm*. This works by adding additional plots as *iplotm* is incremented:

iplotm	**Effect**
0	No plots
1	Title page and Namelist
2	Add grid plot
3	Add equilibrium values plot
4	Add equilibrium errors diagnostic plot
5	Add packing mesh diagnostic plot
6	Add q, pressure, and density profile plot
7	Add profile table
8	Add line plots for ff', p', p, j versus r across the midplane
9	Add line plots of ff', p' versus Ψ

Note, however, that the line plots for ff', p', p, j versus r across the midplane are not included for the case of inverse equilibria.

4. Vacuum and Matrix Construction Code swnw

Swnw.f constructs the wall position, computes the vacuum contribution, and constructs the potential and kinetic energy matrices.

4.1. Wall Options

This routine sets up two vectors *srw* and *szw*, defining the wall position points adjacent to the plasma boundary points (*sr*,*sz*). The points (*srw*,*szw*) are placed at the same angle θ as (*sr*,*sz*) where θ is defined from an origin located inside the plasma. Normally the origin is taken at a major radius value midway between the plasma radial extremes and a vertical axial value of the equilibrium input variable *ZLIM*. The origin can be moved if the parameterization of either the wall or the plasma boundary becomes non-unique.

The wall is defined by option *iwal*. For $iwal = 0$, the wall is taken from the plasma surface shape. This is either extended out by a constant normal distance given by

$(rext - 1)a_{\text{midplane}}$ where a_{midplane} is measured from the geometric center of the plasma boundary (conformal wall), extended by a constant factor (self similar wall), or constructed as an ellipse with the same elongation as the plasma (same shape wall). If $rext \le 1$, the wall is placed on the plasma surface. If $rext > rextmax$, a wall at infinity is used. Typically, $rextmax = 10^3$.

The details of the expansion are set by the parameters *irext* and *norign*. These also set options for defining the origin of the polar coordinate systems used to parameterize the wall points; the parameter *irext* sets the origin of the expansion of the wall and *norign* defines the origin of the coordinates used to interpolate the constructed wall points to the grid values that coincide with the plasma surface points. The actual wall expansion is set by *irext* as follows:

irext = *0*:	Conformal wall			
irext = *+1*:	Self similar wall	subtended from	(*rcnt*,	*0.0*)
irext = *+2*:	Self similar wall	subtended from	(*rcnt*,	*zlim*)
irext = *+3*:	Self similar wall	subtended from	(*xcentr*,	*ycentr*)
irext = *+4*:	Self similar wall	subtended from	(*rmagx*,	*zmagx*)
irext = *–1*:	Same shape wall	subtended from	(*rcnt*,	*0.0*)
irext = *–2*:	Same shape wall	subtended from	(*rcnt*,	*zlim*)
irext = *–3*:	Same shape wall	subtended from	(*xcentr*,	*ycentr*)

irext = *–4*: Same shape wall subtended from (*rmagx*, *zmagx*)

Figure 3 shows the construction of the conformal wall and Figure 4 shows the construction of the self similar wall. The “same-shape” wall is a simplified form of the self similar wall constructed from the elongation and triangularity of the plasma surface and a minor radius extended by *rext* from the plasma surface.

For $iwal = \pm 1$ the wall is read from the file "inwgta" in Namelist format as a series of points $[rwi(k),\ k = 1,\ nwall]$, $[zwi(k),\ k = 1,\ nwall]$. If:

iwal = *+1*: The points (*rwi*, *zwi)* are assumed to include complete circuit

iwal = *–1*: The points (*rwi*, *zwi*) are assumed to include only upper half plane.

The points can be read in any reasonable order (clockwise or counterclockwise, and starting points at outboard or inboard); GATO tests for the sense and starting point and adjusts the points accordingly. The final wall used, however, is ordered counterclockwise, starting from the outboard midplane. This wall can also be extended or reduced by the factor *rext*:

$rext = 1.0$ uses the wall as read in,

$rext < 1.0$ reduces the wall radius,

$rext > 1.0$ extends the wall radius.

The point from which the wall is subtended is controlled by the parameter *irext* as follows:

irext = *0*: Wall extension subtended from (*rcnt*, *0.0*)

irext = *±1*: Wall extension subtended from (*rcnt*, *0.0*)

irest = *±2*: Wall extension subtended from (*rcnt*, *zlim*)

irext = *±3*: Wall extension subtended from (*xcentr*, *ycentr*)

irext = *±4*: Wall extension subtended from (*rmagx*, *zmagx*)

Figure 4 also shows the wall expansion using the *irext* option.

For $iwal = 2$, the wall is constructed from a harmonic series:

$$r_w = R_0 + a\sum_k^N b_k \cos k\theta \ , \tag{10a}$$

$$Z_w = Z_0 + \kappa a\sum_k^N c_k \sin k\theta \ , \tag{10b}$$

With N, R_0, a, κ, b_k, c_k read in from the file "inwgta" in Namelist format. A default is given for the DIII–D wall as it existed prior to the addition of upper divertor hardware in 1997.

Note that: For *iwal = 0* the parameter *nwp* set in the code must be greater than or equal to *itht+2* (for *irext=0*) or *itht+1* (for $irext \neq 0$).
For $|iwal| = 1$ the parameter *nwp* must be greater than or equal to *nwall*.
For $|iwal| = 2$ the parameter *nwp* must be greater than or equal to *itht+1*. (If this is not true, the number of points set for the wall is limited to *nwp*.)

An important restriction is that one should always keep *rext* either of order unity ($rext \leq 5$) or set it greater than *rextmax*. Otherwise, if $10 \leq rext < rextmax$ the vacuum calculation has insufficient resolution and results in a large numerical vacuum matrix asymmetry and the vacuum matrix may not be positive definite. In that case, spurious very negative eigenvalues result. Generally the results with $rext \approx 5$ and $rext > rextmax$ are indistinguishable. In all cases, if the wall cuts the plasma surface, a warning should be given. However, occasionally this condition is not properly detected. In that case, the vacuum matrix is constructed with one or more large and negative eigenvalues, which is then flagged.

4.2. Construction of the Vacuum Contribution

The vacuum contribution to the potential energy matrix is constructed using the Greens function technique described in F. Troyon, *et al.*, Comput. Phys. Commun. **19**, 161 (1980).

4.3. Construction of the Potential and Kinetic Energy Matrices

The Finite Hybrid Element construction of the potential energy matrix is described in detail for the ERATO code in R. Gruber *et al.*, Comput. Phys. Commun. 21, 377 (1981). The construction in GATO follows essentially an identical method.

5. Eigenvalue Code seig

Seig.f solves the numerical eigenvalue problem $AX = \lambda BX$ using the method of an eigenvalue shift λ_0 and Cholesky Decomposition to determine an approximate eigenvalue λ_0 such that the shifted eigenvalue problem:

$$(A - \lambda_0 B)X = (\lambda - \lambda_0)BX \equiv \tilde{\lambda}BX \tag{11}$$

has an eigenvalue $\tilde{\lambda}$ near zero. Sylvesters Theorem provides the number of negative eigenvalues of the shifted matrix $(A - \lambda_0 B)$. The final solution for $\tilde{\lambda}$ and X is then obtained by inverse iteration. Section 5.1 describes the Cholesky Decomposition. Section 5.2 then describes the eigenvalue shift procedure to find the λ_0 closest to the desired eigenvalue.

5.1. General Solution Procedure by Cholesky Decomposition

The Cholesky decomposition decomposes a symmetric Hermitian indefinite matrix such as $(A - \lambda_0 B) \equiv \tilde{A}$ in the eigenvalue equation:

$$(A - \lambda_0 B)X = \tilde{A}X = (\lambda - \lambda_0)BX \equiv \tilde{\lambda}BX \tag{12}$$

by factoring $\tilde{A}$ into a product of $\tilde{A} = U^T DU$, where U is an upper triangular matrix with unit diagonal elements, U^T is its transpose, and D is a diagonal matrix. This factorization is unique for a symmetric Hermitian matrix $\tilde{A}$. The number of negative elements in the diagonal matrix D is the number of negative eigenvalues of the original matrix $\tilde{A}$ (Sylvesters Theorem). The system is then solved by setting $Y = UX$ and solving the pair of matrix equations:

$$\begin{aligned} U^T DY &= \tilde{\lambda}X \\ UX &= Y \end{aligned} \qquad \text{for} \qquad X, Y \text{ and } \tilde{\lambda}. \tag{13}$$

5.2. Eigenvalue search procedure

The search for the eigenvalue in ***seig.f*** is complicated to explain but simple enough in practice. Essentially, it tries to bracket the desired eigenvalue, then isolate it, then converge nearer to it, in separate loops. Finally it homes in on the eigenvalue using inverse iteration. For up-down asymmetric cases, the eigenvalues come in pairs. Then the input parameter nev determines the *nev*'th pair of eigenvalues — *i.e.,* for *nev = 1* the code will bracket and isolate eigenvalues 1 and 2 for up-down asymmetric cases and eigenvalue 1 for up-down symmetric cases. In more detail, the eigenvalue searching is performed as follows:

1. Start with initial guess *al0*.

2. Bracket search: bracket desired eigenvalue (*nev*) by multiplying or dividing (*al0 – al0bas*) by *dal0*. Up to *nbrmax* iterations can be done. If it reaches this limit, or if one of the search limits *al0min* or *al0max* is reached or exceeded, the code quits.

3. Isolation search: isolate desired eigenvalue by binary search so that a single eigenvalue is bracketed between λ^{n+1} and λ^{n}. Up to *nismax* iterations can be done. If the limit is reached a warning is given but the procedure continues to the next step.

4. Convergence search: isolate desired eigenvalue by binary search. Up to *ncymax* times. A warning message is given if $\left|\lambda^{n+1} - \lambda^{n}\right| > epschy$. A separate warning is also given if the eigenvalue still not isolated and this search is completed. The first warning can be safely ignored but not the warning that the eigenvalue is not isolated since the inverse iterations can then converge to the wrong eigenvalue.

5. Inverse iterations: Perform one more Cholesky decomposition on $A - \lambda^{n} B$ and converge to λ by inverse iteration: up to *nitmax* iterations or until the difference in two iterations of λ is less than the tolerance *epscon*. A warning is given if the number of iterations exceeds *nitmax*. This warning should not be ignored if the final eigenmode is a physically unstable mode.

6. Final Cholesky decomposition on $A - \lambda_c B$, where λ_c is the converged eigenvalue from the inverse iterations, to get the eigenvector truly corresponding to λ_c. This is skipped if *ncyfin = 0* and done if *ncyfin = 1*.

7. For up-down asymmetric plasmas, the eigenvalues come in degenerate pairs. Then the Cholesky decomposition searches are actually performed for the 2**nev*'th eigenvalue and only isolate the eigenvalue down to the pair of degenerate eigenvalues. On final convergence to the degenerate pair, sometimes the inverse iterations pick out one of the pair and sometimes the other. *nreslv* = ±1 can be used to try to isolate them individually before the inverse iterations. However, this seldom works well since the eigenvalues tend to very close – near machine precision. *nreslv*

= 0 is the default, in which case the actual eigenvalue and eigenvector obtained is uncontrolled. The degenerate eigenvectors in the pair are essentially complex conjugates.

6. Plotting and Diagnostic Code splt

Splt.f performs the plotting for the eigenvector $\underline{\xi}$ and derived quantities such as δB and δA in a variety of representations:

Normal orthogonal representation: $\xi_\psi = \xi \cdot \nabla\psi \big/ |\nabla\psi|$ $\xi_{\rm pol} = \dfrac{\xi \cdot \nabla\psi \times \nabla\phi}{|\nabla\psi \times \nabla\phi|}$ $\xi_{tor} = \xi \cdot \nabla\phi \big/ |\nabla\phi|$

Native GATO representation: $X = \xi \cdot \nabla\psi,$ $U = \xi \cdot \nabla\chi_G$ $Y = \xi \cdot \nabla\phi$

Cylindrical representation: $\xi_r = \xi \cdot \nabla r$ $\xi_z = \xi \cdot \nabla z$ $\xi_\phi = \xi \cdot \nabla\phi$

Field line representation: $\xi_n = \xi_\Psi$ $\xi_\perp = \dfrac{\xi \cdot \nabla\psi \times B}{|\nabla\psi \times B|}$ $\xi_{\rm parallel} = \xi \cdot B \big/ |B|$

Similarly for the perturbed magnetic field components δB. Here *X, U,* and *Y* are the quantities GATO actually computes and the other representations are derived from them: for example, $\xi_\psi = X / |\nabla\Psi|$, *etc*. *X, U,* and *Y* are essentially the covariant components of the displacement in a special nonorthogonal coordinate system (ψ, χ_G, ϕ), where χ_G is closely related to the poloidal coordinate χ. The code performs similar diagnostics for the perturbed magnetic field components δB and the perturbed vector potential field components δA.

Splt.f provides displacement arrow plots in the poloidal plane of ξ, δB, and δA. For each of the components of ξ, δB, and δA in each of the representations, ***splt.f*** also performs Fourier decompositions, plots line plots along chosen poloidal and flux grid lines, and contour contour plots in the poloidal plane. Contour plots of the equilibrium quantities and of the grid values are also provided as an option. In addition, contour plots of the components of the integrand for δW and a contour plot of the perturbed poloidal flux surfaces are provided.

6.1. General Plotting Options

The overall plotting is controlled by the parameter *ioutp*. This sequentially adds further plots as its value is increased and sets the default. However, the basic collection of plots can be substantially modified from the default by a number of other options. These are described below. The parameter *ioutp* provides the following default collection of plots.

ioutp =

0	No plots
+ 1	Plot profiles and contour plots of grid quantities r, z, χ
+ 2	Plot also contour plots of equilibrium quantities
+ 3	Plot wall and rational q surface positions
+ 4	Plot also displacement vectors ξ
+ 5	Plot also perturbed field vectors δB
+ 6	Plot also perturbed vector potentiol vectors δA
+ 7	Plot also perturbed flux contours $\delta\psi$
+ 8	Plot also line plots for displacement ξ
+ 9	Plot also line plots for perturbed magnetic field δB
+ 10	Plot also line plots for perturbed vector potential δA
+11	Plot also Fourier plots for displacement ξ
+12	Plot also Fourier plots for perturbed magnetic field δB
+13	Plot also Fourier plots for perturbed vector potential δA
+14	Plot also contour plots of ξ components
+15	Plot also contour plots of perturbed magnetic field δB
+16	Plot also contour plots of perturbed vector potential δA
+17	Plot also contour plots of δW components

If *ioutp* $= -k$, the code plots the wall position and displacement vectors and only the kth plot: For example:

if *ioutp* = *–9*, the code plots the wall position, displacement factors, and the line plots of δB

if *ioutp* = *–11*, the code plots the wall position, displacement factors, and the Fourier analysis of ξ.

In addition, each of the defaults set by *ioutp* can be individually overridden using a set of 17 Namelist variables each named in the form *ioxxxp* with *xxx = wal, eig*, etc. as follows:

Use:

iowalp	Wall position
iomshp	Equilibrium quantity contour plots
ioeqlp	Mesh quantities
ioeigp	Displacement vectors ξ
iodbvp	Perturbed magnetic field vectors δB
iodbva	Perturbed vector potential vectors δA
iopsip	Perturbed ψ contours
iolinp	Displacement line plots $\xi(\psi, \theta = c)$ and $\xi(\psi = c, \theta)$
iolnbp	Perturbed magnetic field line plots $\delta B(\psi, \theta = c)$ and $\delta B(\psi = c, \theta)$
iolabp	Perturbed vector potential line plots $\delta A(\psi, \theta = c)$ and $\delta A(\psi = c, \theta)$
iofftp	Displacement Fourier plots $\xi_m(\psi)$
ioffbp	Perturbed magnetic field Fourier plots $\delta B_m(\psi)$
ioffap	Perturbed vector potential Fourier plots $\delta A_m(\psi)$
ioconp	Displacement contour plots $\xi(r, z)$
iodlbp	Perturbed magnetic field contour plots $\delta B(r, z)$
iodlap	Perturbed vector potential contour plots $\delta A(r, z)$
iodlwp	Perturbed potential energy contour plots $\delta W(r, z)$

To override *ioutp* using the individual *ioxxxp*, set:

ioxxxp = + *2*: to force plot
ioxxxp = + *1*: to perform calculations but don't plot
ioxxxp = *0*: to use default
ioxxxp = – *2*: not perform calculations and not plot

6.2. Options for Control of Vector Component Plotting

The parameters *nxiplt*, *nxuplt*, *nxrplt*, and *nxpplt* influence the displacement components that are shown in each of the plot types specified by *ioutp* and the *ioxxxp*. Generally, these work by adding an additional component for the respective representation as the parameter value is incremented when the parameter is postive and plotting just one component if the parameter value is negative as follows:

nxiplt =

0	None
+1	Plot ξ_ψ in line plots/Fourier plots/contour plots
+2	Plot ξ_ψ, ξ_x in line plots/Fourier plots/contour plots
+3	Plot $\xi_\psi, \xi_x, \xi_\phi$ in line plots/Fourier plots/contour plots
-1	Plot ξ_ψ in line plots/Fourier plots/contour plots
-2	Plot ξ_x in line plots/Fourier plots/contour plots
-3	Plot ξ_ϕ in line plots/Fourier plots/contour plots

nxuplt =

0	None
+1	Plot X in arrow plots/line plots/Fourier plots/contour plots
+2	Plot X and U in arrow plots line plots/Fourier plots/contour plots
+3	Plot X, U, and Y in arrow plots line plots/Fourier plots/contour plots
-1	Plot X in arrow plots line plots/Fourier plots/contour plots
-2	Plot U in arrow plots line plots/Fourier plots/contour plots
-3	Plot Y in arrow plots line plots/Fourier plots/contour plots

Similarly, the parameters *nxrplt*, *nxpplt*, control plotting of the three individual components of ξ in the cylindrical and field line representations. The specific components selected by *nxiplt*, *nxuplt*, *nxrplt*, and *nxpplt* are:

nxiplt Components of ξ in normal orthogonal representation:

$$\xi_\psi = \frac{\xi \cdot \nabla\psi}{|\nabla\psi|}, \quad \xi_{\text{poloidal}} = \frac{\xi \cdot (\nabla\psi \times \nabla\phi)}{|\nabla\psi \times \nabla\phi|}, \quad \xi_{\text{toroidal}} = \frac{\xi \cdot \nabla\phi}{|\nabla\phi|}$$

nxuplt Components of ξ in covariant GATO representation:

$$X = \xi \cdot \nabla\psi, \; U = \xi \cdot \nabla\chi, \; Y = \xi \cdot \nabla\phi$$

nxrplt Components of ξ in cylindrical representation:

$$\xi_r = \xi \cdot \nabla r, \; \xi_z = \xi \cdot \nabla z, \; \xi_\phi = \frac{\xi \cdot \nabla\phi}{|\nabla\phi|}$$

nxpplt Components of ξ in field line representation:

$$\xi_n = \xi_\psi = \frac{\xi \cdot \nabla\psi}{|\nabla\psi|}, \quad \xi_\perp = \frac{\xi \cdot (\nabla\psi \times B)}{|\nabla\psi \times B|}, \quad \xi_{\text{parallel}} = \frac{\xi \cdot B}{|B|}$$

The parameters *nbiplt*, *nbuplt*, *nbrplt*, and *nbpplt, naiplt*, *nauplt*, *narplt*, and *napplt* similarly control plotting of the three individual components of δB and δA in each representation:

nbiplt Components of δB in normal orthogonal representation:

$$\delta B_n = \delta B_\psi = \frac{\delta B \cdot \nabla\psi}{|\nabla\psi|}, \quad \delta B_{\text{poloidal}} = \frac{\delta B \cdot (\nabla\psi \times \nabla\phi)}{|\nabla\psi \times \nabla\phi|}, \quad \delta B_{\text{toroidal}} = \frac{\delta B \cdot \nabla\phi}{|\nabla\phi|}$$

nbuplt Components of δB in covariant representation:

$$\delta B_{\mathbf{X}} = \delta B \cdot \nabla\psi, \ \delta B_{\mathbf{U}} = \delta B \cdot \nabla\chi, \ \delta B_{\mathbf{Y}} = \delta B \cdot \nabla\phi$$

nbrplt Components of δB in cylindrical representation:

$$\delta B_r = \delta B \cdot \nabla r, \ \delta B_z = \delta B \cdot \nabla z, \ \delta B_{\text{toroidal}} = \frac{\delta B \cdot \nabla\phi}{|\nabla\phi|}$$

nbpplt Components of δB in field line representation:

$$\delta B_n = \delta B_\psi = \frac{\delta B \cdot \nabla\psi}{|\nabla\psi|}, \quad \delta B_\perp = \frac{\delta B \cdot (\nabla\psi \times B)}{|\nabla\psi \times B|}, \quad \delta B_{\text{parallel}} = \frac{\delta B \cdot B}{|B|}$$

naiplt Components of δA in normal orthogonal representation:

$$\delta A_n = \delta A_\psi = \frac{\delta A \cdot \nabla\psi}{|\nabla\psi|}, \quad \delta A_{\text{poloidal}} = \frac{\delta A \cdot (\nabla\psi \times \nabla\phi)}{|\nabla\psi \times \nabla\phi|}, \quad \delta A_{\text{toroidal}} = \frac{\delta A \cdot \nabla\phi}{|\nabla\phi|}$$

nauplt Components of δA in covariant representation:

$$\delta A_{\mathbf{X}} = \delta A \cdot \nabla\psi, \ \delta A_{\mathbf{U}} = \delta A \cdot \nabla\chi, \ \delta A_{\mathbf{Y}} = \delta A \cdot \nabla\phi$$

narplt Components of δA in cylindrical representation:

$$\delta A_r = \delta A \cdot \nabla r, \ \delta A_z = \delta A \cdot \nabla z, \ \delta A_{\text{toroidal}} = \frac{\delta A \cdot \nabla\phi}{|\nabla\phi|}$$

napplt Components of δA in field line representation:

$$\delta A_n = \delta A_\psi = \frac{\delta A \cdot \nabla\psi}{|\nabla\psi|}, \quad \delta A_\perp = \frac{\delta A \cdot (\nabla\psi \times B)}{|\nabla\psi \times B|}, \quad \delta A_{\text{parallel}} = \frac{\delta A \cdot B}{|B|}$$

Two other Namelist variables allow additional plots with the displacements but have no counterpart for the perturbed field or vector potential. The variable *nxdplt* adds plots of the derivatives of the displacement vector as described in the Table below. This operates in the same way as the parameters *nxiplt*, *nxuplt*, *nxrplt*, and *nxpplt*. Plots of the Fourier analysis of the electric potential Φ ($E = -\underline{V} \times \underline{B} = \Phi$) can also be obtained using *ncphip* if *ncphip = +1*, and the Hamiltonian poloidal angle coordinate from *mshchi = +4*, and (*nham1, nham2, nham3) = (0,0,2)* (see Section 6.3 below). For that choice, the Fourier coefficients of the electric potential are given fairly trivially by the Fourier components of the normal displacement. Otherwise, if *mshchi* $\neq$ *+4*, and (*nham1, nham2, nham3)* $\neq$ *(0,0,2)*, then *ncphip* is ignored.

$$nxdplt = \begin{cases} 1 & \text{Plot } \partial X/\partial\psi \\ 2 & \text{Plot } \partial X/\partial\theta \\ 3 & \text{Plot } \partial U/\partial\theta \\ 4 & \text{Plot } \partial Y/\partial\theta \end{cases}$$

6.3. Coordinate Mesh Options

The parameters *mshpsi* and *mshchi* control the radial and poloidal coordinates the actual plots are drawn in, and the poloidal Fourier decomposition is done in for the case of *mshchi*. *mshpsi* affects the line plots and Fourier plots versus 'radius'' and in the label for line plots versus poloidal angle. *mshchi* affects the line plots versus angle and the poloidal angle used in the Fourier analysis and the labels for the plotted angle in the line plots versus radius. For the radial flux surface coordinate, there are 12 separate options divided into four sets of basic radial variable, poloidal flux, radius across outboard midplane, toroidal flux, and volume contained within the given flux surface. For each of these, an option is provided to plot the unnormalized variable, normalized variable, or either the square or the square root of the basic variable as appropriate. For the poloidal angle, two correspond to geometrical angles subtended from different centers (magnetic axis or actual center of the surface). These are occasionally useful for comparing with analytic results. In addition the three basic poloidal angles discussed in the mapping of

EQALARC, PEST, or HAMILTONIAN are available. However, normally only the straight field line coordinate makes sense for the Fourier analysis.

mshpsi =

0	Use Ψ (unnormalized) for horizontal axis coordinate (base)
1	Use Ψ (unnormalized) for horizontal axis coordinate
2	Use ψ (normalized) for horizontal axis coordinate
3	Use $\sqrt{\psi}$ for horizontal axis coordinate
4	Use squared radius ρ^2 on outboard midplane
5	Use radius ρ on outboard midplane
6	Use normalized radius ρ on outboard midplane
7	Use (unnormalized) toroidal flux Γ
8	Use normalized toroidal flux Γ
9	Use square root of normalized toroidal flux $\sqrt{\Gamma}$
10	Use (unnormalized) volume within the surface V
11	Use normalized volume within the surface V
12	Use square root of normalized volume within the surface $\sqrt{V}$

mshchi =

0	Use poloidal geometrical angle from magnetic axis
1	Use poloidal geometrical angle from surface center
2	Use arclength as poloidal angle ($\times 2\pi$)
3	Use PEST angle as poloidal angle
4	Use Hamiltonian angle: $J = r^{\text{nham1}} B_{\text{p}}^{\text{nham2}} B^{\text{nham3}}$

The parameters *ntphase* and *torphase* fix the arbitrary toroidal phase of the eigenmode. *ntphase = 0* is the default phase and can result in an arbitrary phase dependent only on how the initial guess for the eigenvector is loaded. Other values have the effects shown in the following table. Normally, the two options to maximize either the real or imaginary component of the normal displacement *ntphase = ±4* is the most useful. The options *ntphase = ±1 or ±2* are occasionally useful for simply reversing the phases from an earlier run. Similarly, the options *ntphase = ±5* are occasionally useful for balancing the real and imaginary parts.

$$ntphase = \begin{cases} 0 & \text{None} \\ -1 & \text{Phase shift } -\pi/2 \text{ (switch Re } (\xi) \text{ and Im } (\xi)) \\ +1 & \text{Phase shift } +\pi/2 \text{ (switch Re } (\xi) \text{ and Im } (\xi)) \\ -2 & \text{Phase shift } -\pi/4 \text{ (add equal mixture Re } (\xi) \text{ and Im } (\xi)) \\ +2 & \text{Phase shift } +\pi/4 \text{ (add equal mixture Re } (\xi) \text{ and Im } (\xi)) \\ -3 & \text{Phase shift } ntor \times torphase \times \pi \\ +3 & \text{Phase shift } torphase \times \pi \\ -4 & \text{Maximize Im } (\xi) \\ +4 & \text{Maximize Re } (\xi) \\ -5 & \text{Set Re } (\xi)^2 = \text{Im } (\xi)^2 \text{ with opposite sign} \\ +5 & \text{Set Re } (\xi)^2 = \text{Im } (\xi)^2 \text{ with same sign} \\ -6 & \text{Set Re } (\xi) \text{ zero where Im } (\xi) \text{ is maximum} \\ +6 & \text{Set Im } (\xi) \text{ zero where Re } (\xi) \text{ is maximum} \\ -7 & \text{Set Re } (\xi) \text{ zero where Re } (\xi) \text{ is maximum} \\ +7 & \text{Set Im } (\xi) \text{ zero where Im } (\xi) \text{ is maximum} \\ -8 & \text{Maximize } \Sigma(\text{Im } (\xi)^2) / \Sigma(\text{Re } (\xi)^2) \\ +8 & \text{Maximize } \Sigma(\text{Re } (\xi)^2) / \Sigma(\text{Im } (\xi)^2) \end{cases}$$

6.4. Other Plotting Options

Several plots show the data as selected phase values – i.e. toroidal angles in successive planes. This includes the arrow displacement and field vector plots, the perturbed flux surface plots, and the contour plots of the perturbed quantities (when $ncont \leq 0$ – see below). The phase values are selected by taking *NTANGL* toroidal planes with *NTANGL = min((ntor + 1), nph)* where *nph* is set in a parameter statement as *nph=5*.

The parameters *nskpj* and *niskpi* control the number and spacing of the arrows in the displacement and perturbed field and vector potential phase plots. These determine the number of mesh points skipped in plotting the arrows in the flux (*nskpj – 1*) and poloidal (*nskpi – 1*) directions. The parameter *ncont* controls the number of contours selected in contour plots, which is taken as $|ncont|$. This is for contour plots of both the equilibrium and the perturbed quantities. For the perturbed quantities, the values are complex and the sign of *ncont* is used to select whether the contour plots of are taken as the real and imaginary parts ($ncont > 0$) or as selected toroidal angles ($ncont \leq 0$). If $ncont = 0$ a warning is given but the plots are still done with no actual contours. The parameter *ncplot* sets the number of flux surfaces shown in the perturbed flux surface plot.

Two parameters, *lineplt* and *lampplt* control the type of line plots produced when the global line plot option is selected. Setting *lineplt > 0* selects only the line plots versus flux for selected poloidal angles, whereas setting *lineplt < 0* selects only the line plots versus poloidal angle for selected flux surfaces. Setting *lineplt = 0* selects both. The line plots in each case consist of either plots of the magnitude versus flux or angle for key selected phase values (*lampplt > 0*) – i.e. toroidal angles – or the phase and amplitude separately (*lampplt < 0*), or both in separate plots (*lampplt = 0*). For *lampplt = 0* or *lampplt > 0* the phase values are selected by taking *NTANGL* toroidal planes. The parameters *njplot* and *niplot* control the actual surface and angle selected for the line plots. For *njplot > 0* the line plot shows the selected components of ξ, δB, and δA for the *njplot 'th* flux surface as a function of the poloidal angle selected by *mshchi*. For *njplot = 0* the last flux surface is selected. When *njplot < 0* line plots are done for each of the *(1 + k ×(JPSMAX/njplot))'th* flux surfaces. Here, *JPSMAX* is the last surface selected according to *njedge*. Similarly, *niplot* selects line plots for the *niplot'th* poloidal angle if *niplot > 0* and line plots are done for each of the *(1 + k ×(itht/niplot))'th* poloidal angles if *niplot < 0.* When *niplot = 0* the outboard midplane is selected as an average of the first and *itht'th* poloidal rays since the perturbed quantities are defined at the centers of the mesh cells so the first and last poloidally are half a grid cell above and below the midplane respectively.

For the displacement as a whole and the corresponding δB and δA quantites, the sign can be reversed using *nxisgn*; *nxisgn = +1* uses the default sign selected somewhat arbitrarily from the initial loading of the eigenvector solution. *nxisgn = -1* reverses this sign. This feature is useful for obtaining 'nice' plots with predominantly positive sign mode structures.

The parameter *njedge* controls whether the plasma edge displacement is included and plotted in the plot normalizations. Sometimes this displacement can be very large and swamp the remainder of the plot. To suppress the edge in the displacement vector plots and line and Fourier plots use *njedge = +1*. To suppress the edge in the displacement vector plots only use *njedge = +2*. To suppress more edge surfaces set *njedge < 0* to a value set at the number of surfaces to be cut in the plots.

6.5. Output Files

Output files from ***splt.f*** contain the data in severable formats in addition to the standard output file "o4gta" and the standard graphics file "gato4.cgm". The file "vacuum.dat" is an interface for vacuum codes. It contains the plasma boundary, wall, displacement components and δB on the plasma boundary, the Jacobian on the boundary, the symmetrized vacuum matrix and the local vacuum matrix asymmetry. The latter data consists of an array of $N_\theta \times N_\theta$ matrix elements and can be read and plotted by other graphics codes for diagnostic purposes. The file "diagnostic.dat" has a subset of the data in ASCII. The file "nimrod.dat" is a binary file containing data suitable for comparison with linear runs from the NIMROD code. Both these are considered defunct. Finally, the file "o4dump" contains all the input equilibrium and output perturbation data calculated for the run in ASCII format for writing to MDSplus or interfacing with other graphics codes. Each of the output files is renamed by the standard scripts as follows:

"o4gta"	→	"outpt.out4"
"gato4.cgm"	→	"outpt.cgm4"
"vacuum.dat"	→	"outpt.vac"
"diagnostic.dat"	→	"outpt.diag"
"nimrod.dat"	→	"outpt.nim"
"o4dump"	→	"outpt.dmp"

7. Interpretation of Code Results

The interpretation of the results from any stability code is always complicated. For GATO there is a particular feature that can simplify some interpretation but causes difficulties in other cases. The Finite Hybrid Elements in GATO can numerically destabilize some modes. Generally, modes that are highly localized on one or a few flux surfaces are numerically destabilized. This, in particular, includes the stable Alfven and acoustic continua – the marginal point for these modes is numerically shifted into the unstable region so that without any correction, GATO almost always finds an eigenmode with negative λ with a finite radial mesh. These modes generally converge quadratically (i.e. with $(N_\psi \times N_\chi)^{-1}$ for N_ψ / N_χheld fixed) to vanishing eigenvalue. However, there are also some physical modes, which either converge more slowly with the mesh (smaller factor with $(N_\psi \times N_\chi)^{-1}$) or in the other direction – i.e. to more unstable. These modes can be buried in the numerically destabilized continuum at finite mesh and not show up as the most unstable mode until a sufficiently fine mesh is used.

However, the most recent version now includes a numerical correction in the FHE matrix construction that restabilizes the locally singular continuum modes for any finite mesh. The correction can be invoked by setting the Namelist parameter *ncorr > 1* with a numerical factor *corrfac* also applied and defaulted to *corrfac = 1.0*. However, *corrfac* can be set otherwise if desired and setting *corrfac = 0.0* has the same effect as setting *ncorr = 0*. Essentially, this correction is the numerical δW for a localized mode evaluated according to the FHE approximation. This has the form:

$$\delta W_c = -\frac{1}{2}\int\left[\frac{\frac{1}{4}h_\psi^2}{\left\langle |B|^2/|\nabla\psi|^2\right\rangle}\langle S\rangle\langle T+\tfrac{1}{2}S\rangle\left|\frac{\partial\xi_\psi}{\partial\psi}\right|^2 d\psi\right]$$

Here, $S = \left(\frac{B\times\nabla\psi}{|\nabla\psi|^2}\right)\bullet\left[\nabla\times\left(\frac{B\times\nabla\psi}{|\nabla\psi|^2}\right)\right]$ is the local shear, which can also be written as:

$S = \frac{f'}{r^2} - \frac{f}{r^2|\nabla\psi|^2}\left(\frac{\partial}{\partial\psi}(|\nabla\psi|^2)_\nu - \Delta^*\psi\right)$, and

$T = \left(\frac{j\cdot B}{|\nabla\psi|^2} - S\right) = \frac{f}{r^2|\nabla\psi|^2}\left(\frac{\partial}{\partial\psi}(|\nabla\psi|^2)_\nu - 2\Delta^*\psi\right)$, and the $\langle\ \rangle$ are flux surface averages:

$\langle Q \rangle = \frac{1}{4\pi^2} \oiint Q J \, d\chi \, d\phi$, with $J \equiv \sqrt{g} \equiv |\nabla\psi \cdot \nabla\chi \times \nabla\phi|^{-1}$ the Jacobian.

For highly singular modes, the factor $|\partial\xi_\psi / \partial\psi|^2$ in the integrand for δW_c is large, making a large local contribution but this factor is multiplied by the local numerical mesh size h_ψ^2. In the limit where $h_\psi^2 \to 0$, the correction term vanishes; hence the results extrapolated to infinite resolution are unchanged. This is true whether the mode in question is highly localized or not. Several options exist for numerically implementing this correction. The default is *ncorr = +1* but the others appear to result in much the same effect in all cases studied so far. The options are:

ncorr = +3 Use standard surface averaged correction except for first cell
Use absolute value of standard surface averaged correction for first cell to force stabilization

ncorr = +2 Use standard surface averaged correction only when stabilizing
Use no correction when standard surface averaged correction is destabilizing

ncorr = +1 Use standard surface averaged correction
$\left\langle |B|^2 / |\nabla\psi|^2 \right\rangle^{-1} \langle S \rangle \langle T + \frac{1}{2} S \rangle$ and surface averaged Jacobian

ncorr = 0 No correction

ncorr = -1 Use local Jacobian *J* with surface averaged
$\left\langle |B|^2 / |\nabla\psi|^2 \right\rangle^{-1} \langle S \rangle \langle T + \frac{1}{2} S \rangle$

ncorr = -2 Use local Jacobian *J* and local $T + \frac{1}{2}S$ with surface averaged
$\left\langle |B|^2 / |\nabla\psi|^2 \right\rangle^{-1} \langle S \rangle$

ncorr = -3 Use local Jacobian *J*, local $S(T + \frac{1}{2}S)$, with surface averaged
$\left\langle |B|^2 / |\nabla\psi|^2 \right\rangle^{-1}$

ncorr = -4 Use all local terms $\left(|B|^2 / |\nabla\psi|^2 \right)^{-1} S\left(T + \frac{1}{2} S\right)$ in place of surface averaged standard correction

Note that *corrfac = 1.0* is analytically the optimum. The net result of applying the correction is that the singular continuum modes converge to marginal stability from the stable side instead of from the unstable side. Then, any unstable modes found with a finite mesh are truly unstable, making the determination of stability much simpler.

In principle, one or more of the following techniques should be used to avoid misinterpretation of the results. This is especially critical when the numerical correction is not applied but even with the numerical correction, these procedures will greatly increase the dependability of the result. While none is a guarantee of a correct interpretation, they are listed in order of increasing reliability:

(i) For a given finite mesh, choose a small but negative eigenvalue as a marginal stability criterion. For a mesh of $N_\psi \times N_\chi = 100 \times 200$ this criterion should be about $\lambda_c \approx -10^{-4}$. Eigenvalues below this (more negative) are then considered unstable and eigenvalues above this (more positive) are then considered stable. The cutoff can be chosen by observing the structure of the most unstable mode. Eigenvectors that are strongly localized across a couple of grid points with sharp local gradients are usually numerically destabilized continuum modes. Physically unstable modes are generally global and extend across at least several grid points. However, a physical mode that is near marginal stability can numerically pick up some mixture of the nearby continuum modes whether it is physically stable or unstable.

(ii) If a physical parameter is varied continuously the eigenvalue generally approaches zero fairly rapidly. Even if a given value of the parameter might not be clearly physically unstable, points on either side are clear and a reasonable marginal point can be identified. The mode structure also changes quickly from a global mode to a strongly localized continuum-like mode.

(iii) The most reliable method is to perform a partial or full convergence study. This is done by varying the mesh keeping N_ψ / N_χ fixed and plotting λ against $(N_\psi \times N_\chi)^{-1}$. Then λ can be extrapolated to $(N_\psi \times N_\chi)^{-1} = 0$. If the result is negative the mode is physically unstable. Otherwise, λ should extrapolate to zero. In the latter case, one should see the eigenfunction become increasingly singular.

Appendix A: Namelist Input

Line 1 of the Namelist input is reserved for a title
Line 2 of the Namelist input is reserved for a title
Line 3 of the Namelist input is reserved for a title

Type	Variable	Default	Definition (* *Option is not yet implemented)*
Physical Case	*ntor*	1	Toroidal mode number
	ncase	0	Set = 0 for full compressible; = 1 for incompressible
	norm	0	Set = 0 for full KE norm; > 0 for normal displacement only
	nlt	0	Set = 0 for ideal wall boundary conditions; =1 for line tying
	nmod	0	Set = 1 to force $\xi(q<1)=0$; Set = 2 for floating boundary condition
	nlim	0	Set limiter boundary condition at *nlim'th* poloidal angle
	idnsty	0	Read density profile if = 2 or set density profile if ≤ 0
	ndnxp0	0	Set density profile for *idnsty* < *0* as $\psi^{ndnxp0}(1-\psi^{ndnxp1})^{ndnxp2}$
	ndnxp1	2	Set density profile for *idnsty* < *0* as $\psi^{ndnxp0}(1-\psi^{ndnxp1})^{ndnxp2}$
	ndnxp2	2	Set density profile for *idnsty* < *0* as $\psi^{ndnxp0}(1-\psi^{ndnxp1})^{ndnxp2}$
	bfieldf	1.0	Set overall normalization for magnetic field
	qxin	0.0	Rescale q_0 to *qxin* and B_ϕ if non zero. Otherwise use input q_0
	btdes	0.0	Rescale B_ϕ to *btdes* and q_0 if non zero. Otherwise use input q_0
	qsurf	2.0	Calculate q_0 value required to obtain $q_{\lim} = qsurf$
	gamma	5/3	Adiabatic factor C_p/C_v
	rmantl		Fraction of edge plasma treated as cold mantle in calculation of beta
Equilibrium Type	*nmap*	0	Set = 0 for direct equilibrium; set > 0 for TOQ and < 0 for JSOLVER
	neqtyp	1	Set = 0 for old style TOQ input; set = 1 for TOQ input with heading
	ndoublt	0	Set = 0 for Dee coding options; set = 1 if doublet equilibrium
	ndivert	1	Set = 0 for limiter edge; set = 1 for diverted edge
Numerical correction	*ncorr*	0	Set = 0 for standard FHE; set ≠ 0 for numerical restabilization
	corrfac	1.0	Factor for numerical restabilization corrections
Grid	*jpsi*	*npx*	Number of flux surfaces
	itht	*ncx*	Number of poloidal angles
	isym	0	Set = 0 for up-down asymmetry; set = 1 for symmetric
	igrid	0	Set = 0 for equalarc poloidal angle; = 1 for PEST angle
	nham1	0	Alternative output angle χ_H; $J(\psi,\chi_H,\phi)=r^{nham1}B_p^{nham2}B^{nham3}$
	nham2	0	Alternative output angle χ_H; $J(\psi,\chi_H,\phi)=r^{nham1}B_p^{nham2}B^{nham3}$
	nham3	+2	Alternative output angle χ_H; $J(\psi,\chi_H,\phi)=r^{nham1}B_p^{nham2}B^{nham3}$
	nmesh	1	Set = 1 for repacking mesh; set = 0 for no packing; set < 0 to read mesh
	npak	0	Number of q values for additional packing; set < 0 to pack in *nq*
	mpak	0	Number of ψ values for additional packing; set ≠ 0 to pack in ψ^{cspak}

Mesh Packing	*nedge*	+4	Set > 0 to force edge packing; = 0 for no edge packing; < 0 to include search for rational surfaces near edge
	npkmax	*npx*	Maximum number of rational surfaces packed
	nrat	*1 +npx/3*	Number of flux surfaces reserved for packing
	nrepeat	0	Set > 0 (< 0) to pack only (skip) the *nrepeat*'th occurrence of q in *plpak*
	nppack	1	Set > 0 (< 0) to eliminate packing in negative (positive) shear
	nqpack	0	Set $\neq 0$ to set weights evenly distributed in $s^{nqpack}q$
	nsrcheg	+1	Set > 0 to include search for rational surfaces in last cell near edge
	ncutedg	0	Use linear interpolation in packing within *ncutedg* of *jpsi*
	minpak	1	Minimum number of points between packing points in packing mesh
	maxpak	4	Initial number of points between packing points in packing mesh
	incpak	4	Initial number of unweighted points in packing distribution mesh
	psipak	2.0	Set initial ψ distribution evenly spaced in $\tilde{\psi}^{psipak}$
	*chiwth**	0.0	Set weighting for poloidal coordinate distribution
	cspak	0.5	Set distribution for underlying ψ mesh distribution to $\tilde{\psi}^{cspak}$
	psincr	0.995	Increment for constructing initial ψ mesh near Doublet separatrix
	pkfrac	2/3	Fraction of flux surfaces reserved for packing ($\times pkfrac$)
	qpfrac	1/3	Fraction of flux surfaces reserved for distributing in q
	epsrat	$+10^{-10}$	Cutoff value to ignore packing fraction
	sedg0	0.0	Inverse width of packing weight at edge
	sedg1	0.0	Amplitude of packing weight at edge
	plpak(k,l)	0.0	Additional q packing: k=(q value, width, weight); (l=1,*npak*)
	pspak(k,l)	0.0	Additional ψ packing: k=(ψ value, width, weight); (l=1,*mpak*)
	epspak	$+10^{-2}$	Packing point tolerance for ignoring gap in constructing weight mesh
	spakmn	0.10	Minimum packing weight for any packed point
	swidmn	1.0	Minimum (inverse) width in special surface packing
	swidmx	100.0	Maximum (inverse) width in special surface packing
Boundary Mapping	*mapmaxd*	10	Maximum number of plasma boundary mapping attempts
	dpsisl	0.0	Cut plasma surface fraction *dpsisl* on initial boundary map attempt
	dpsisd	0.5×10^{-4}	Decrement *dpsisl* by $2^k \times$ *dpsisd* on each (k'th) mapping failure
Magnetic Axis Fitting	*nqaxis*	0	Force use of axis fit from one or other alternative or weighted mean
	nwtmag	25	Maximum number of Newton iterations for magnetic axis
	nfitmax	10	Maximum number of searches to find points to fit at magnetic axis
	nfitpts	14	Sets minimum number of points in fit to axis
	ifitrng	2	Range of r grid points in search for direct equilibrium axis fitting
	jfitrng	10	Range of z grid points in search for direct equilibrium axis fitting
	jfitchk	5	Range of ψ grid points in search for direct equilibrium axis fitting
	fitchek	10.0	Sets range of ρ grid points in search for direct equilibrium axis fit
	cnvmag	$+10^{-10}$	Convergence criterion for magnetic axis iterations
	epsaxs	$+10^{-7}$	Tolerance for axis position compared to input

Input Equilibrium Error Tolerances	*maxerlp*	200	Maximum number of poor convergence error reports in plasma
	maxerlv	100	Maximum number of poor convergence error reports in vacuum
	delbox	$+10^{-2}$	Radial increment for extending plasma error checking
	delboz	$+10^{-2}$	Radial increment for extending plasma error checking
	delac	$+10^{-3}$	Global error criterion for input plasma equilibrium accuracy
	delav	$+10^{-3}$	Global error criterion for input vacuum equilibrium accuracy
	delstsf	$+10^{-3}$	Surface average error criterion for input equilibrium accuracy
	delstlp	$+10^{-2}$	Pointwise error criterion for plasma input equilibrium accuracy
	delstlv	$+10^{-2}$	Pointwise error criterion for vacuum input equilibrium accuracy
Mapping and Vacuum Tolerances	*nerstop*	10	Maximum number of error messages of each type printed
	qptol	$+10^{-2}$	Tolerance for q value integration calculated from different schemes
	tolspln	$+10^{-4}$	Allowed tolerance in spline calculations of r,z of surfaces
	tolbtor	$+10^{-7}$	Tolerance in input *btdes* and equilibrium *BTOR* before resetting
	tolsymm	$+10^{-7}$	Equilibrium up-down symmetry tolerance (*isym=1*)
	errsep	$+10^{-5}$	Tolerance for equality of initial and final contour point
	toldrdz	$+10^{-6}$	General tolerance criterion for defining equality of two wall positions
	pvansh	$+10^{-8}$	Criterion for negligible pressure taken as plasma boundary with mantle
	precisn	$+10^{-13}$	Input precision required for iterative elliptic integral
	plossmx	0.25	Maximum allowed precision loss for iterative elliptic integral
	roundff	$+10^{-11}$	General roundoff parameter for testing equality
	bigno	$+10^{+30}$	General largest number possible
General Mapping	*narcmx*	*nlx*	Maximum number of arclength points in flux surface mapping
	ntrymx	10	Maximum attempts to eliminate mapping failure by deleting points
	ntdecr	*nlx/100*	Decrement number of contour points by 2**ntdecr* per failure
	ntmmin	*2 npx*	Minimum number of points taken as sufficient in contour definition
Inverse Equilibrium Mapping Parameters	*nccellr*	+1	Use r,z from cell values (<0) 2D spline ($=0$) or 1D spline (>0) for *Fk*
	peqpk0	0.50	r,z for inverse equilibria interpolated as $r(\psi^{peqpk0}), z(\psi^{peqpk0})$
	peqpk1	0.50	Fk $(1 \le k \le 10)$ for inverse equilibria interpolated as $Fk(\psi^{peqpk1})$
	peqpk2	0.50	Fk $(11 \le k \le 22)$ for inverse equilibria interpolated as $Fk(\psi^{peqpk2})$
Direct Equilibrium Mapping Parameters	*npfit*	120	Minimum points on direct rectangular map: Otherwise force polar map
	npcmin	5	Absolute minimum number of points from mapping surface
	numstp	1	Number of tries allowed to increment *stepfac* in Runge Kutta
	stepfac	0.020	Initial step factor for Runge Kutta integration of outboard ray
	flxstp	0.15	Vertical flux step in Runge Kutta integration of outboard ray in Doublet
	psispl	0.10	Force polar mapping of flux surface for $\psi <$ *psispl*
	delpakf	0.0	Minimum spacing ratio between adjacent points from rectangular map
	delpakc	0.0	Minimum spacing ratio between adjacent points from polar map
	delpkf	0.10	Minimum spacing between points on contour from rectangular map
	delpkc	0.01	Minimum spacing between points on contour from polar map

	psichek	$+10^{-8}$	Tolerance between input ψ and interpolation to starting point for surface
	boxtnd	0.1	Extension of flux surfaces to find approximate flux surface elongation
Rectangular Mesh Contour Mapping Parameters	*maptrace*	0	Mapping trace for surface *maptrace* > *0* or *maptrace* < *0* for boundary
	norient	0	Set orientation priority of search for multiple exits of grid cell
	maxcutc	5	Maximum number of excised points allowed in full contour
	dresolv	$+10^{-8}$	Tolerance for whether the contour passes through grid points
	dlclose	3.0×10^{-4}	Tolerance for deciding if surface contour is closed
	pntshft	$+10^{-8}$	Allowed shift in mapped point to move point back into grid cell
	endtol	0.10	Tolerance of interpolated total arclength vs interpolation input
Polar Mesh Contour Mapping Parameters	*narcln*	120	Sets initial arclength step relative to previous surface by factor *1/narcln*
	nangax	144	Sets maximum angular increment for contours near axis *1/nangax*
	nanglm	360	Sets maximum angular increment for contours near boundary *1/nanglm*
	nbpmax	5	Maximum number of contour restarts to obtain number points < *narcmx*
	nwtmax	20	Maximum Newtons iterations for intersections of ray with contour
	nslmax	10	Maximum sliding interval searches for contour
	nhfmax	4	Maximum number of angular increment divisions for contour points
	bperor	0.01	Allowed point to point change in B_{pol} in flux contour mapping
	sersnm	$+10^{-9}$	Newtons Method convergence criterion: Warning if exceeded
	sertnm	$+10^{-8}$	Newtons Method convergence criterion: Termination if exceeded
	arcmin	0.002	Minimum point to point arclength in flux contour mapping
	delgap	0.1	Gap permitted in closing contour
	stepcut	0.5	Fractional reduction in step size imposed when test for *bperor* fails
Wall and Vacuum Parameters	*iwal*	0	Set = 0 to construct wall; = 1 to read (r_{wall}, z_{wall}); = 2 to read coefficients
	iwalsym	0	Option to read in symmetric (= 0) or asymmetric (≠ 0) wall
	irext	0	Wall option to set center of wall; set = 0 for conformal wall
	norign	0	Define origin of coordinates used to interpolate final wall points
	nwall	60	Number of points to construct (*iwal=0*) or read (*iwal=1,2*) wall
	nekdefn	0	Use elliptic integral expansion (< 0), iterative scheme (> 0); default (= 0)
	maxitek	10	Maximum number of iterations for iterative elliptic integral method
	rext	1.0	Expand default wall by factor *rext*; set = 1.0 for input wall from *iwal*
	rexmax	$+10^{+3}$	Maximum wall expansion for equivalent infinite vacuum
	rcutoff	$+10^{-3}$	Minimum major radius for inboard toroidal wall
Eigenvector File Handling	*nrestrt*	0	Restart eigenvalue iterations from "rgta" file (= 1); set = 0 for no restart
	ndskopt	0	Storage of decomposed matrix in multiple (= 0) or single (set = 1) files
	ndsktim	0	Save and print eigenvalue solver timing with increased detail (> 0)
	ndsksz	0	Set maximum size for single disk file if set ≠ 0; otherwise no size limit
	buffact	0.3	Scale factor to scale buffer size in word addressable I/O
	nev	1	Compute *nev*'th eigenvalue
	neigmax	100	Maximum number of eigenvalues for any guess. Stop if exceeded.

Eigenvalue Solver	*nforce*	0	Set = ±1 to force convergence to one degenerate pair eigenvalue
	nreslv	0	Set = 0 ignore degenerate eigenvalues; = ±1 to resolve
	nbrmax	10	Maximum number of bracket iterations for eigenvalue search
	nismax	10	Maximum number of isolation iterations for eigenvalue search
	ncymax	2	Maximum number of Cholesky iterations for eigenvalue search
	nitmax	20	Maximum number of inverse iterations for eigenvalue search
	ncyfin	1	Set = 1 for Cholesky decomposition with final eigenvalue
	mxdcomp	20	Maximum total Cholesky decompositions; stop if exceeded.
	al0	-10^{-4}	Initial eigenvalue guess
	dal0	10.0	Scale factor for incrementing *al0* in bracket search
	al0bas	+0.0	Offset for scaling bracket search
	al0min	–1.0	Minimum allowed eigenvalue. Stop if exceeded.
	al0max	-10^{-9}	Maximum allowed eigenvalue. Stop if exceeded.
	epschy	$+10^{-5}$	Convergence criterion for Cholesky iterations
	epscon	$+10^{-5}$	Convergence criterion for inverse iterations
Plot Parameters	*lineplt*	0	Plot line plots versus flux surface (> 0), poloidal ray (< 0), or both (= 0)
	lampplt	0	Plot line plots of quantity (> 0), amplitude and phase (< 0), or both (= 0)
	njplot	0	Plot line plot of ξ versus θ of *njplot*'th surface or boundary (= 0)
	niplot	0	Plot line plot of ξ versus ψ of *niplot*'th poloidal angle or midplane (= 0)
	nskpi	+1	Skip every *nskpi*'th angle in displacement vector plot
	nskpj	+1	Skip every *nskpj*'th surface in displacement vector plot
	njedge	+1	Include or exclude plasma edge in plot normalizations
	ntphase	-4	Set toroidal phase option
	npowr	-2	Set transformation option for plot of logarithmically divergent quantities
	ncont	10	Number of contours in contour plots
	ncplot	10	Number of contours in perturbed flux surfaces
	mshpsi	12	Specify radial coordinate in Fourier and line plots
	mshchi	3	Specify poloidal angle in Fourier analysis and line plots
	nxisgn	+1	Reset sign of eigenvector (= ±1)
	nxiplt	+1	Set = +1 plot $\underline{\xi}\cdot\nabla\Psi/\lvert\nabla\Psi\rvert$; +2 add $\underline{\xi}\cdot\nabla\chi/\lvert\nabla\chi\rvert$; +3 add $\underline{\xi}\cdot\nabla\phi/\lvert\nabla\phi\rvert$
	nxuplt	+1	Set +1 to plot $\underline{X=\xi}\cdot\nabla\psi$; +2 to add U; +3 to add $\underline{Y=\xi}\cdot\nabla\phi$
	nxrplt	0	Set = 1,2,3 to plot radial, axial, toroidal ξ components
	nxpplt	0	Set = 1,2,3 to plot normal, perpendicular, and parallel ξ components
	nxdplt	0	Set = +1 plot $\partial X/\partial\psi$; +2 add $\partial X/\partial\theta$; +3 add $\partial U/\partial\theta$; +4 add $\partial Y/\partial\theta$
	ncphip	0	Set = +1 to plot perturbed electric potential contours
	nbiplt	0	Set +1 plot $\underline{\delta B}\cdot\nabla\psi/\lvert\nabla\psi\rvert$; +2 add $\underline{\delta B}\cdot\nabla\chi/\lvert\nabla\chi\rvert$; +3 add $\underline{\delta B}\cdot\nabla\phi/\lvert\nabla\phi\rvert$
	nbuplt	0	Set +1 to plot $\underline{\delta B}\cdot\nabla\psi$; +2 to add $\underline{\delta B}\cdot\nabla\chi$; +3 to add $\underline{\delta B}\cdot\nabla\phi$
	nbrplt	0	Set = 1,2,3 to plot radial, axial, toroidal δB components
	nbpplt	0	Set = 1,2,3 to plot normal, perpendicular, and parallel δB components
	naiplt	0	Set +1 plot $\underline{\delta A}\cdot\nabla\psi/\lvert\nabla\psi\rvert$; +2 add $\underline{\delta A}\cdot\nabla\chi/\lvert\nabla\chi\rvert$; +3 add $\underline{\delta A}\cdot\nabla\phi/\lvert\nabla\phi\rvert$

	nauplt	0	Set +1 to plot $\underline{\delta A}\cdot\nabla\psi$; +2 to add $\underline{\delta A}\cdot\nabla\chi$; +3 to add $\underline{\delta A}\cdot\nabla\phi$
	narplt	0	Set = 1,2,3 to plot radial, axial, toroidal δA components
	napplt	0	Set = 1,2,3 to plot normal, perpendicular, and parallel δA components
	nvfft	0	Specify number of Fourier harmonics as 2^{nvfft}; set = 0 for maximum
	torphase	0.0	Add *torphase* to default toroidal phase of computed eigenvector
Plot Scale Factors	*dpltfac*	0.10	Overall scale factor for real space eigenvector plots
	dsplnrm	0.10	Scale factor for displacement vector arrows
	dspldbv	5.0	Scale factor for δB vector relative to displacement
	dspldav	0.5	Scale factor for δA vector relative to displacement
	psiscal	0.4	Scale factor for perturbed flux surface displacement
	plsuprs	$+10^{-2}$	Suppression criterion for amplitudes in plot of fourier components
Plot Control	*iomshp*	0	Force on (+2) or off (–2) or leave (0) mesh plots
	ioeqlp	0	Force on (+2) or off (–2) or leave (0) equilibrium plots
	iowalp	0	Force on (+2) or off (-2) or leave (0) wall and plasma surface plots
	ioeigp	0	Force on (+2) or off (-2) or leave (0) ξ vector plots
	iodbvp	0	Force on (+2) or off (-2) or leave (0) δB vector plots
	iodavp	0	Force on (+2) or off (-2) or leave (0) δA vector plots
	iopsip	0	Force on (+2) or off (-2) or leave (0) perturbed surface plots
	iolinp	0	Force on (+2) or off (-2) or leave (0) ξ line plots
	iolnbp	0	Force on (+2) or off (-2) or leave (0) δB line plots
	iolnap	0	Force on (+2) or off (-2) or leave (0) δA line plots
	iofftp	0	Force on (+2) or off (–2) or leave (0) ξ fourier analysis plots
	ioffbp	0	Force on (+2) or off (–2) or leave (0) δB fourier analysis plots
	ioffap	0	Force on (+2) or off (–2) or leave (0) δA fourier analysis plots
	ioconp	0	Force on (+2) or off (-2) or leave (0) ξ contour plots
	iodlbp	0	Force on (+2) or off (-2) or leave (0) δB contour plots
	iodlap	0	Force on (+2) or off (-2) or leave (0) δA contour plots
	iodlbw	0	Force on (+2) or off (-2) or leave (0) δW contour plots
Diagnostic Output Control	*iplotm*	9	Specify default number of plots from ***smap.f***; set = 9 for all plots
	ioutm	0	Specify debug output from ***smap.f***; set = 0 for minimum output
	ioutw	0	Specify debug output from ***swnw.f***; set = 0 for minimum output
	iouta	0	Specify matrix pattern output from ***swnw.f***; set = 0 for minimum output
	ioute	0	Specify debug output from **seig.f**; set = 0 for minimum output
	ioutp	17	Specify output from ***splt.f***; set = 17 for all plots

Appendix B: Equilibrium Mapping Procedure

B1 Overall Procedure:

The overall structure of the mapping code ***smap.f*** can be summarized as follows:

1. Construct the ψ mesh in the variable *CS*, equally spaced in $s = \left(\frac{(\psi - \psi_0)}{(\psi_1 - \psi_0)} \right)^{\alpha}$ with $\psi \equiv PSIVAL$, $\psi_0 \equiv PSIMAX$, $\psi_l \equiv PSILIM$, and $\alpha \equiv cspak$, and interpolate the input profiles $p(\psi)$, $p'(\psi)$, $f(\psi)$, and $ff'(\psi)$ on to this mesh.

2. Compute the last closed flux surface using the sequence of calls:

$$\left(SURFACE \rightarrow SURFMAP \rightarrow \begin{cases} EQDMAP \rightarrow EQDCELL \\ \\ TOQMAP \rightarrow TOQCELL \end{cases} \right).$$

3. Initialize flux surface mapping:

   For Direct Equilibria (*nmap=0*): Compute the orthogonal ray from the axis to the outboard surface to get starting points for calculation of flux contours (*ORTGRAY* and *RUNKUT*)

   For Inverse Equilibria (*nmap≠0*): Compute the flux meshes (*SETPSIM*)

4. Compute flux meshes for $j = 1, jpsi$:

   (i) Compute flux surface grid:

   For Direct Equilibria (*nmap = 0*): (*EQDMAP)*

   For Inverse Equilibria (*nmap ≠ 0*): (*TOQMAP*)

   (ii) Compute flux surface grid and poloidal angles (*MAPARC*)

5. Recompute mesh with packing (*MESHPAK*).

6. Recompute flux surface mapping:

   For Direct Equilibria (*nmap = 0*): Compute the orthogonal ray from the axis to the outboard surface to get starting points for calculation of flux contours (*ORTGRAY* and *RUNKUT*)

   For Inverse Equilibria (*nmap ≠ 0*): Compute the flux meshes (*SETPSIM*)

7. Recompute flux meshes for $j = 1$, *jpsi:*

   (i) Compute flux surface grid:

   For Direct Equilibria (*nmap=0*): (*EQDMAP)*

   For Inverse Equilibria (*nmap≠0*): (*TOQMAP*)

   (ii) Compute Equilibrium quantities $Fk(j,i)$ on flux surface grid:

   For Direct Equilibria (*nmap=0*): (*EQDCELL)*

   For Inverse Equilibria (*nmap≠0*): (*TOQCELL*)

8. Output final $\psi(j), j = 1, N_\psi, \chi(i), i = 1, N_\chi$ mesh*: (WRITMAP)*

B2 Mapping of Direct Equilibria:

(i) For direct equilibria (*nmap = 0)*, *EQDMAP* performs the mapping for each flux surface *PSIVAL(j)*:

(ii) Compute start positions $(r1, z1)$ for the respective surface from *ORTGRAY* and *RUNKUT.*

(iii) Use *FURPLM* to get r,z coordinate intersections or *CNTOUR* to get r,z values of the flux surface on a fine mesh.

(iv) Use *SORTER* and *REMOVE* to sort and remove contour points too close together or too disparate.

(v) Find the arclength values on this fine (r,z) mesh.

(vi) Renormalize z so that the surface is approximately circular.

(vii) Construct an equal spaced arclength mesh on the new renormalized surface.

(viii) Expand the deformed surface back to the real mesh. Points on this will then be ''optimally'' spaced near the tips of elongated surface tips.

(ix) Compute q by integration using this optimal (r,z) mesh: compare Simpsons, trapezoidal, cubic spline and print error if differences > *qptol.*

(x) Compute χ mesh

a. If equal arc, find PEST χ by integration with optimal mesh
b. If PEST, find arclength χ

(xi) Optimal (r,z) mesh is passed to *EQDCELL* for use in computing the equilibrium quantities $Fk(j,i)$ on the mesh.

In this mapping procedure, the two routines *FURPLM* and *CNTOUR* are used to map the individual surfaces on to a fine grid. These two routines have their relative advantages and disadvantages. Most notably, *FURPLM* works very well away from the axis where there are sufficiently many grid intersections, and can handle strong shaping, including indentation and a divertor where $B_{pol} = 0$. In contrast, *CNTOUR* is highly accurate near the magnetic axis but is limited to simple contours. Figure 5 shows the two different procedures.

FURPLM works as follows:

Given an x-y grid with a variable p defined on the grid *FURPLM* finds *NP* points *(XP(k),YP(k))* describing the contour by finding the grid line intersections from cell to cell until the curve is completed. As the curve representing the surface enters a cell, the routine keeps track of where it entered and searches for all possible exits through one of the three remaining sides or through a grid point. The routine also keeps track of which direction the curve is moving. When multiple exits are detected (for example near a divertor) the code attempts to choose the exit leading to a closed contour. When it fails to do so, it begins the search again with various defoult choices switched.

On finding the entry and exit for a cell, an interpolation is used to determine the (r,z) value of the actual exit point, which is then stored. The final result is a set of (r,z) values from the intersection of the surface with the grid lines.

For any surface, a diagnostic trace can be set for by specifying *maptrace > 0* for surface *maptrace*. If *maptrace < 0*, the trace is done for the plasma boundary contour.

CNTOUR works as follows:

The contour is determined by defining a series of rays intersecting with the contour. For a given ray, the intersection is first bounded by a coarse sliding interval search along the ray to find two points on the ray bounding the intersection. A sliding interval search, rather than a binary search, is used for the coarse grid so that non-monotonic ψ can be handled. Newtons Method is then used to converge to the intersection point and the coordinates are stored as *(XP(k),YP(k))*.

The ray angular spacing is controlled dynamically and so the total number of contour points *(XP(k),YP(k))* is determined by the routine. Normally, the poloidal angle increment is defined as the minimum of *ARCL/RAD* and $2\pi\ DANG(RAD0/RAD)$, where *DANG* is an

angle fraction of 2π, *ARCL* is an arclength spacing between successive points, and *RAD* and *RAD0* are the computed radii at the respective and first (i.e. $\theta = 0$) contour points respectively. After each point on the contour is found, the relative change in B_{pol} from the previous point is checked to ensure that it is less than *bperor*. If the change in poloidal field exceeds *bperor*, the angular increment is successively reduced, up to *nhfmx* times, and the point is recalculated. If the number of contour points exceeds the maximum allowed dimension *ntmax*, the mapping for this contour is restarted with *bperor* increased and appropriate warnings are printed. The mapping can restart up to *nbpmax* tries, after which, the routine aborts.

The interpolation and extrapolation routines are inaccurate if some of points found by *FURPLM* or *CNTOUR* are too close together. The routine *SORTER* sorts the points into the correct order and the routine *REMOVE* chooses a subset of those points which are well spaced and redefines the arrays *(XP(k),YP(k))* to apply only to these points. The point deleted is chosen to make the spacing the most even, except at the endpoints where the endpoint is always retained and the neighbouring point is removed. The parameters *delpakf, delpakc, delpkf*, and *delpkc* control how close the spacing needs to be before points are deleted.

B3 Mapping of Inverse Equilibria:

For inverse equilibria (*nmap = 1)*, *TOQMAP* does most of the real mapping but this is much simpler than for *nmap = 0*. Essentially, $r(\psi,\theta)$ and $z(\psi,\theta)$ are interpolated using bicubic splines to obtain $r(\psi,\theta)$ and $z(\psi,\theta)$ on the desired GATO mesh. The remainder of the mapping is then done as in EQDMAP:

(i) Compute the flux meshes (SETPSIM)

(ii) Compute interpolation coefficients for the input equilibrium $r(\psi,\theta)$ and $z(\psi,\theta)$

(iii) Find the arclength values of this fine (r,z) mesh

(iv) Renormalize z so that the surface is approximately circular

(v) Construct an equal spaced arclength mesh on the new renormalized surface.

(vi) Expand the deformed surface back to the real mesh. Points on this will then be ''optimally'' spaced near the tips of elongated surface tips.

(vii) Compute q by integration using this optimal (r,z) mesh: compare Simpsons, trapezoidal, cubic spline and print error if differences > *qptol*.

(viii) Compute χ mesh (as in *EQDMAP*)

(ix) If equal arc, find PEST χ by integration with optimal mesh

(x) If PEST, find arclength χ

(xi) Optimal (r,z) mesh is passed to *TOQCELL* (as in *EQDMAP*) for use in computing the equilibrium quantities $Fk(j,i)$ on the mesh.

B4 Construction of the Poloidal Angle Grid:

MAPARC works as follows: Input and output quantities are:

Input quantities	**Definition**	**Output quantities**	**Definition**
PSIV	Psi value of the flux contour.	*QPVAL*	Safety factor value on this flux surface
JVAL	Index for the flux surface: *JVAL = 0* corresponds to the plasma surface)	*SV0, SV1, SV2, SV3,* and *SV4*	Surface integrals on this flux surface
FVAL	Toroidal Field function value for the flux surface	*TP(k)*	Arclength around the contour
(XP(k),YP(k))	Grid points defining the contour	*ST1(k)*	PEST angle around the contour
NPC	Number of grid points defining the contour	*ST2(k)*	Hamiltonian-like angle coordinate around the contour

The mapping from the *(r,z)* grid to the flux grid is done for each flux contour *PSIV* as follows:

1. Compute diagonal increments between the *(XP(k),YP(k))* from each *k* to *k+1* around the flux contour and store the incremented distances in *TP(k)*.

   Compute the diagonal increments on a transformed approximately circular surface. The transformation from the actual surface is a change in metric to:

   $dl = \sqrt{dr^2 + g_\kappa^2 dz^2}$, where g_κ^2 is the inverse elongation of the flux surface.

The incremented distances for the transformed surface are stored in *TPP(k)*. Figure 5 shows how the renormalization is done for elongated surface.

2. Compute the arclengths between successive points *(XP(k),YP(k))* for both the real and transformed surfaces by integrating *TP* and *TPP*. The arclengths are stored in *ARC(k)* and *ARCC(k)* for $k = 1,npc$ respectively.

3. Define a grid of arclength values around the surface:

   The odd points are computed from equal arclength around the transformed surface, stored in *TPP*, and mapped on to the real surface in *TP*. The total number of points, *NTMSH* is forced to be odd to ensure the start and end points are included – *NTMSH* is set equal to:

   either $NTMESH = \begin{cases} ntmax \text{ or} \\ ntmax - 1 \end{cases}$ to ensure maximum accuracy.

   The even points are then set halfway between the odd points on the real contour to ensure equal spacing for three point integrations. These are then mapped to the transformed surface and stored in *TPP*. The points should then be well distributed for future interpolations.

4. Define the integrands *ST1* and *ST2* for the later calculation of the PEST and Hamiltonian-like poloidal coordinates on this grid.

5. Compute the cubic spline coefficients for the integrands for the partial integrations that specify the PEST and Hamiltonian-like coordinates and integrate around the real flux surface to compute the normalizations corresponding to *QPVAL* and *FQVAL*.

   Reset the interpolation coefficients for *XP(ARC)* and *ZP(ARC)*.

6. Compute the PEST and Hamiltonian-like coordinates on the arclength mesh *TP* by partial integration of *ST1* and *ST2* around the real contour and store them back in *ST1* and *ST2*.

Appendix C: Potential and Kinetic Energy Matrix Construction

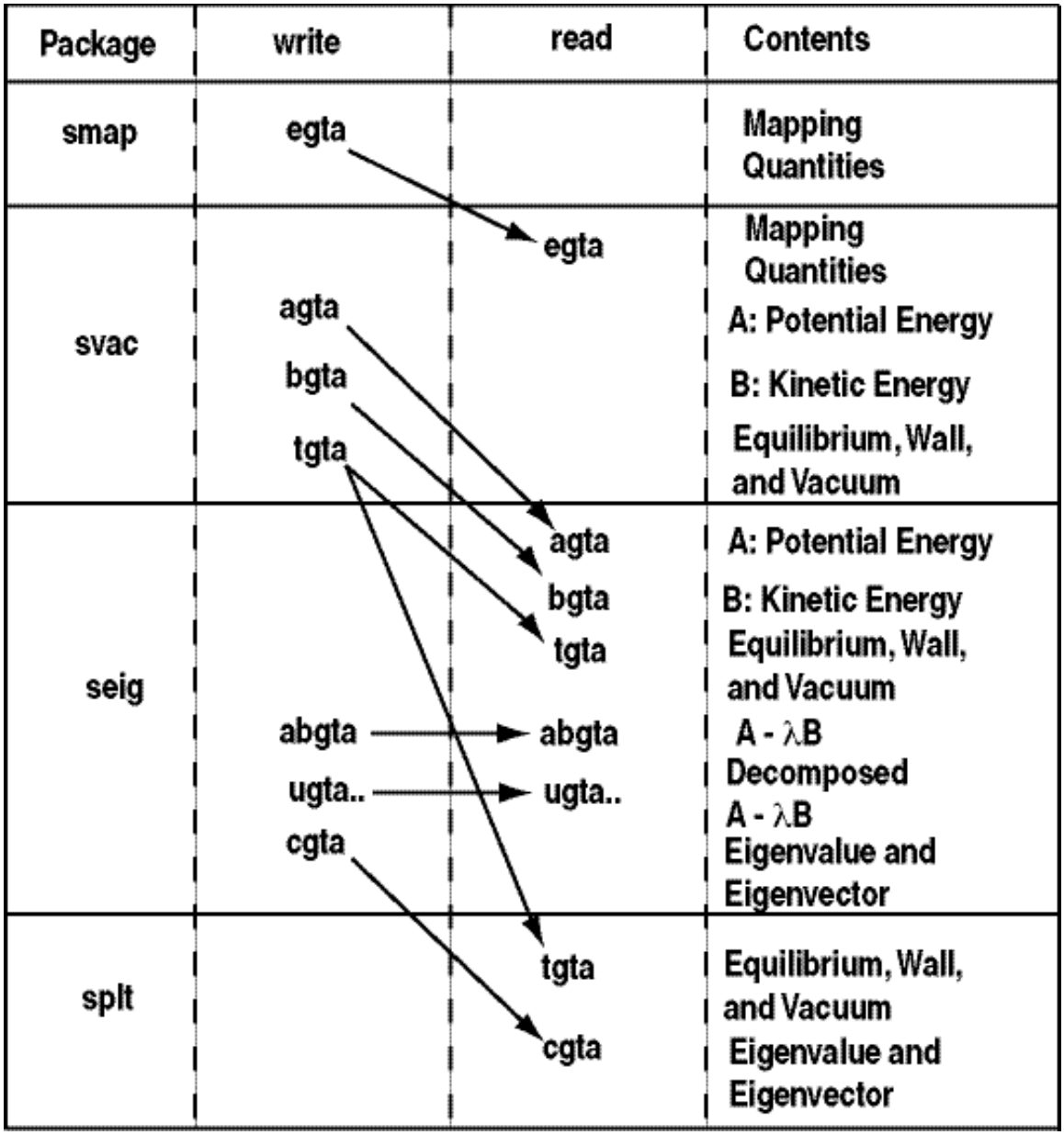
The matrices A and B representing the potential and kinetic energy are both sparse block diagonal. The storage order for the Finite Hybrid Elements is as follows:

$$
\begin{array}{l}
\{[X_0^r,X_0^i]\,,[(U_{1,l}^r,U_{1,l}^i),l=1,N_\chi)\,,(Y_{1,l}^r,Y_{1,l}^i),l=1,N_\chi),(X_{1,l}^r,X_{1,l}^i),l=1,N_\chi)]\,, \\
\qquad\qquad [(U_{2,l}^r,U_{2,l}^i),l=1,N_\chi)\,,(Y_{2,l}^r,Y_{2,l}^i),l=1,N_\chi),(X_{2,l}^r,X_{2,l}^i),l=1,N_\chi)]\,, \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\qquad\qquad [(U_{k,l}^r,U_{k,l}^i),l=1,N_\chi)\,,(Y_{k,l}^r,Y_{k,l}^i),l=1,N_\chi),(X_{k,l}^r,X_{k,l}^i),l=1,N_\chi)]\,, \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\qquad\qquad [(U_{N_\psi,l}^r,U_{N_\psi,l}^i),l=1,N_\chi)\,,(Y_{N_\psi,l}^r,Y_{N_\psi,l}^i),l=1,N_\chi),(X_{N_\psi,l}^r,X_{N_\psi,l}^i),l=1,N_\chi)]\}
\end{array}
$$

Here, $[X_0^r,X_0^i]$ represents the real and imaginary part of X at the magnetic axis. The coefficients $U_{k,l}^r,U_{k,l}^i$ and $Y_{k,l}^r,Y_{k,l}^i$ represents the real and imaginary part of U and Y at the half nodes $\psi_{k-1/2}$, χ_l, and the $X_{k,l}^r,X_{k,l}^i$ represent the real and imaginary part of X at the full nodes ψ_k , χ_l.

The data is efficiently stored in a single one dimensional array that accounts for zeros in the matrix pattern through a small set of integer arrays used to describe the block patterns, starting column in the full matrix and starting address in the stored one dimensional array, and relative storage addresses for the start of each column and the positions and numbers of zeros in each column. The connections between Finite Hybrid Elements yield a total of six separate block types for Figure 8 internal separatrix topologies, or three in the case of a simply nested Dee cross section. This yields a total of 63 different column patterns of zero and non-zero matrix elements, of which 25 can describe the column patterns in a simply nested Dee cross section. The data for the individual column patterns is stored by column pattern and block pattern with additional integer arrays used to map individual blocks and columns to their respective pattern types. This yields a large saving in the required storage since it effectively reduces the block dimension (ostensibly N_ψ) to just six and the column dimension (ostensibly $16N_\chi$

or $12N_\chi$ if block overlapping is accounted for). The rank of the full matrix is $144\,N_\psi^2\,N_\chi^2$ but the total length of the packed matrix is of the order of $198\,N_\psi\,N_\chi$ at the expense of a half dozen integer arrays of order at most $(l_p, 6N_\chi)$, where $l_p = 63$ is the number of column patterns.


Package
write
read
Contents
smap
egta
Mapping Quantities
svac
egta
Mapping Quantities
agta
A: Potential Energy
bgta
B: Kinetic Energy
tgta
Equilibrium, Wall, and Vacuum
seig
agta
A: Potential Energy
bgta
B: Kinetic Energy
tgta
Equilibrium, Wall, and Vacuum
abgta
abgta
A - λB Decomposed
ugta..
ugta..
A - λB Eigenvalue and Eigenvector
cgta
splt
tgta
Equilibrium, Wall, and Vacuum
cgta
Eigenvalue and Eigenvector

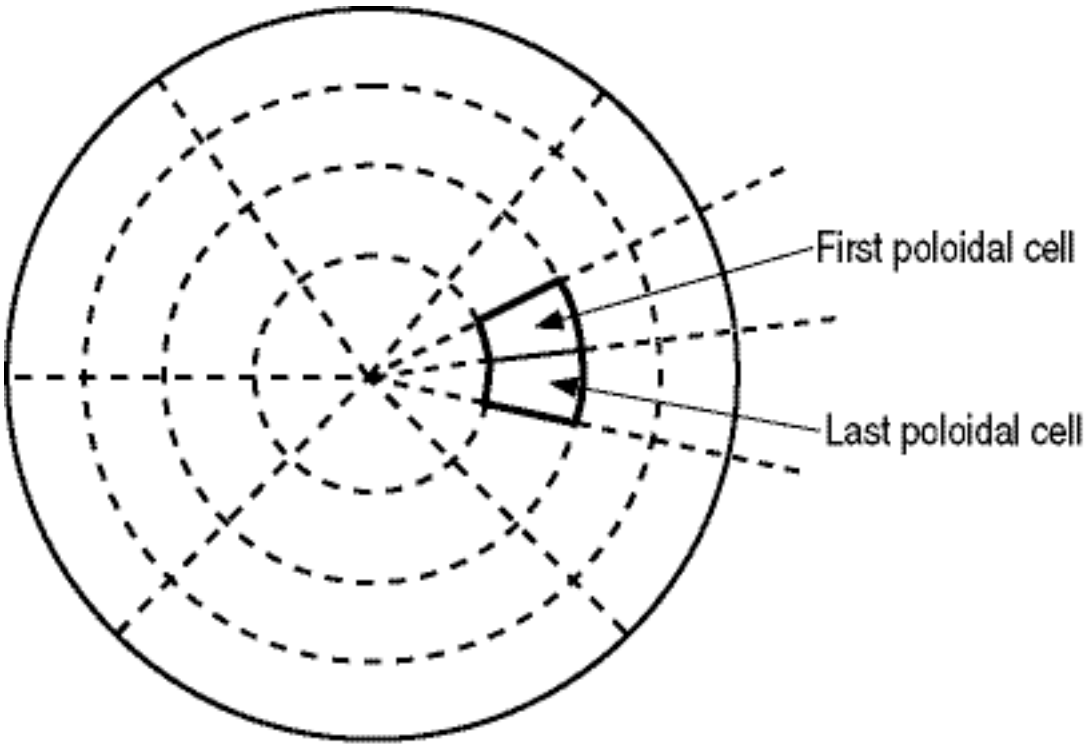


Figure 2

032-97 fig.4/rs

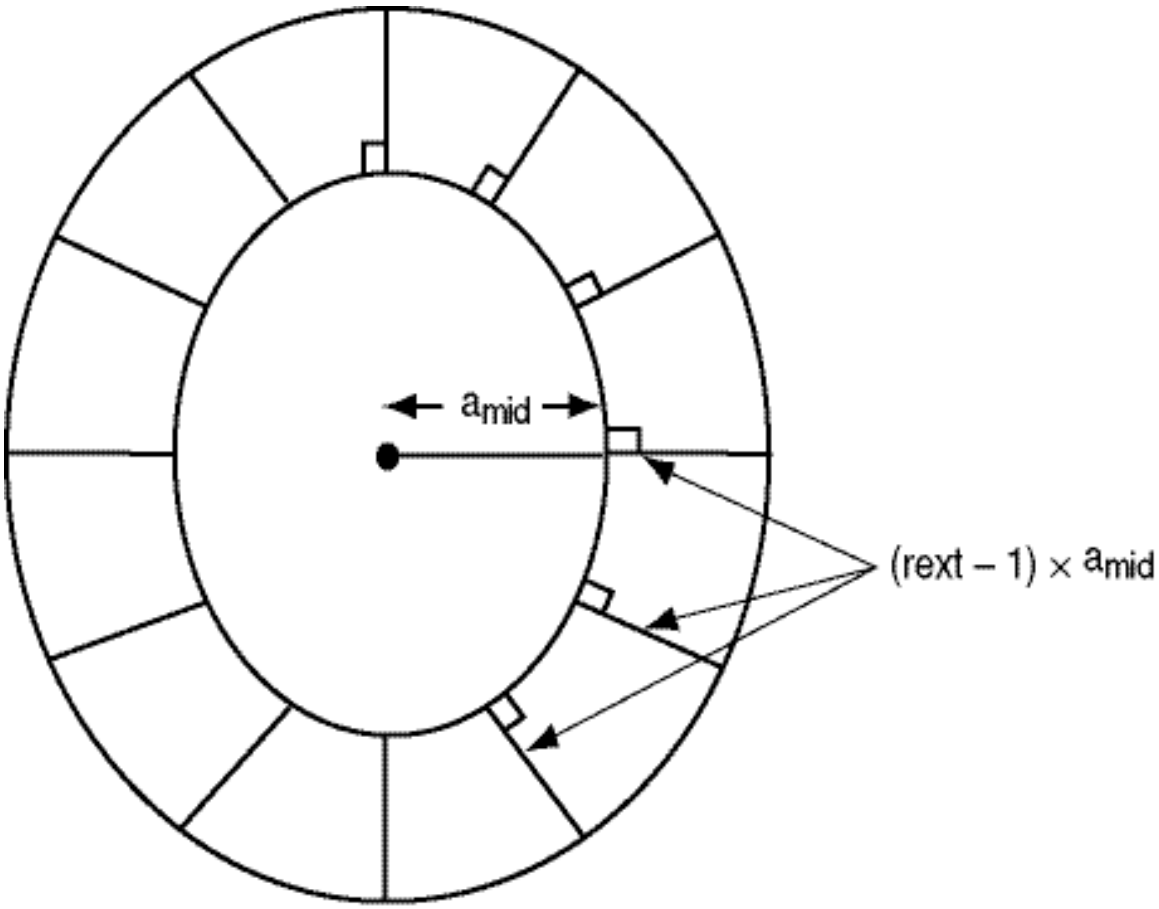


Figure 3 032-97 fig.2/rs

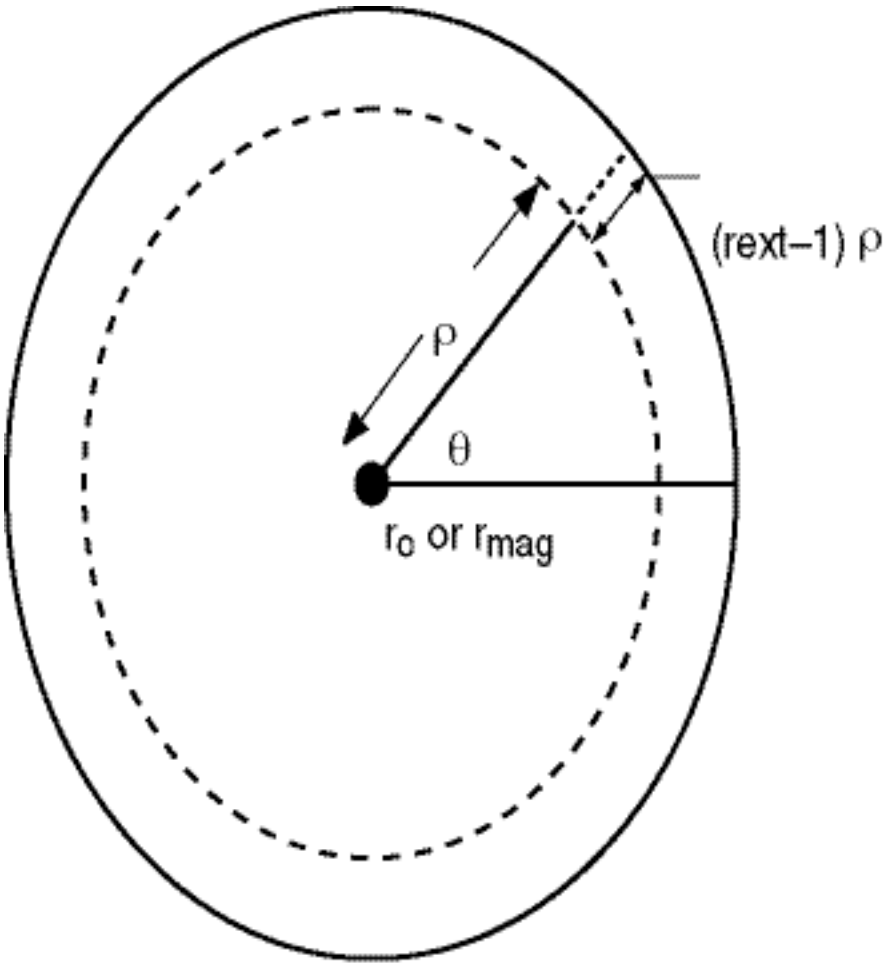
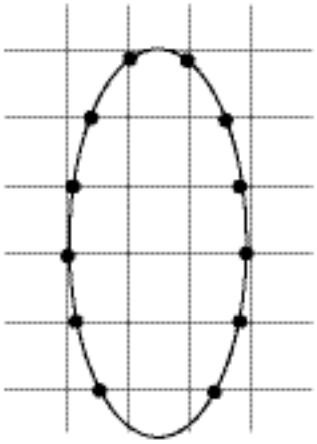


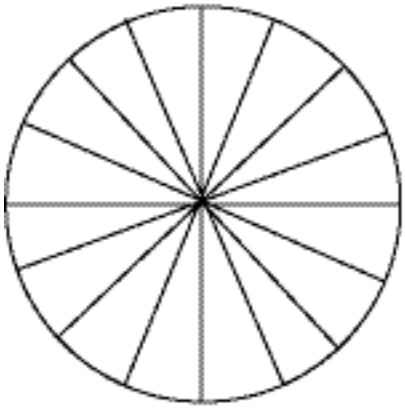
Figure 4 032-97 fig.3/rs

FURPLM



-Search for intersection of contour with grid lines as contour exits and enters each grid cell
-X-point treated when curve has 3 exits in one cell

CNTOUR



– Sliding search along ray
– Newtons iteration to converge
– Control of angular increment to ensure
$\Delta B_{pol} < B_{perror}$

Figure 5 032-97 fig.5/rs

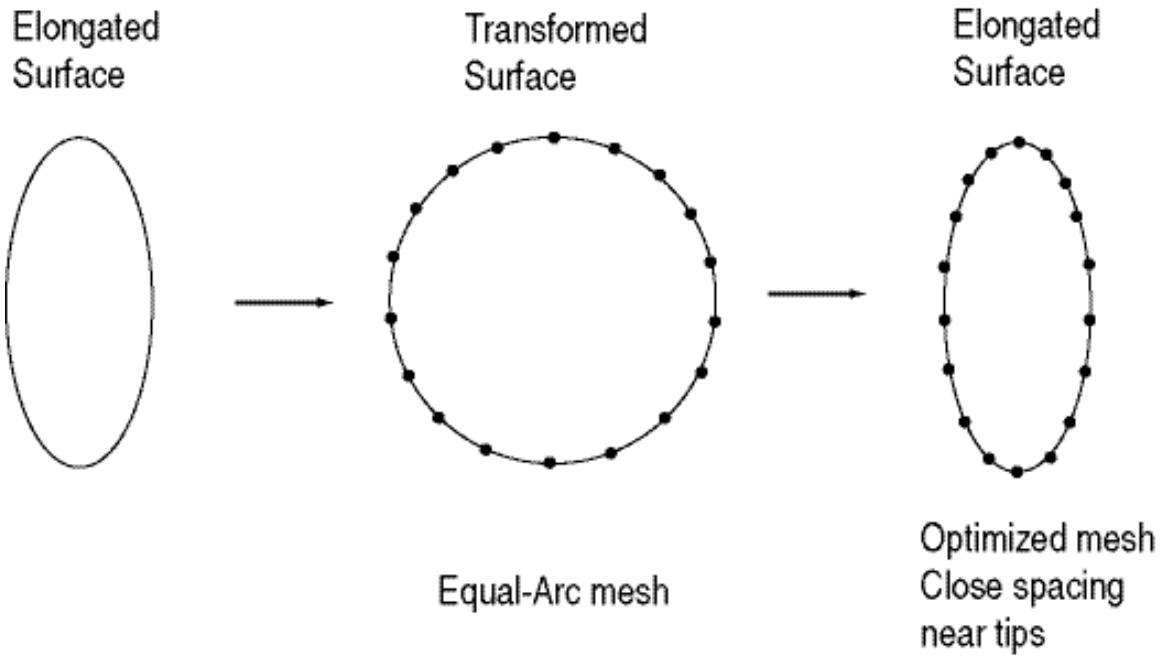


Figure 6

032-97 fig.6/rs