

Oculus: The Eye into Chaos

Dr. S.R. Hudson*

Princeton Plasma Physics Laboratory, PO Box 451, Princeton NJ 08543, USA

Prof. Y. Suzuki†

National Institute for Natural Sciences, National Institute for Fusion Sciences, 322-6 Oroshi, Toki, 509-5292, Japan

(Dated: November 30, 2016)

The `Oculus` package is a continually-under-development suite of diagnostic subroutines for non-integrable, toroidal magnetic fields used in the numerical simulation of magnetic confinement of fusion-research plasmas. `Oculus` is freely distributed, with the expectation that users will promptly inform the developer(s) of any errors.

Suggestions and requests are welcome, indeed encouraged! Subroutines, expanded documentation etc. will be developed on demand.

documentation for oculus version 16

Contents

I. user supplied magnetic field	2
II. user supplied coil description	2
III. macro expansion and compilation	3
IV. error flag	3
V. Biot-Savart subroutines	4
A. bs00aa : compute the magnetic field produced by a filamentary current loop of arbitrary shape;	5
VI. “cylindrical” subroutines	7
A. ga00aa : find the magnetic axis;	8
B. ho00aa : find the homoclinic points (of the stable/unstable manifold);	12
C. ec00aa : find action extremizing curves using global integration;	15
D. tr00aa : measure rotational-transform;	18
E. pp00aa : fieldline tracing for Poincaré plot, calculate Lyapunov exponent;	20
F. gc00aa : follow guiding center;	22
G. rz00aa : construct cylindrical Fourier harmonics of flux surface using fieldline tracing;	23
H. ad00aa : anisotropic diffusion using locally-field-aligned coordinates;	24
I. bn00aa : compute $(\mathbf{B} \cdot \mathbf{n})_{m,n}$ on a given toroidal surface;	28
VII. toroidal-cylindrical coordinate-vector transformation	30
A. bc00aa : interpolation of toroidal surfaces; construction;	31
B. bc00ab : interpolation of toroidal surfaces; evaluation;	33
VIII. “toroidal” subroutines	34
A. aa00aa : construct vector potential in toroidal coordinates;	35
B. aa00ba : evaluate vector potential (in toroidal coordinates);	37
C. ir00aa : construct irrational flux-surface (incomplete);	38
D. qf00aa : construct quadratic-flux minimizing surface using pseudo fieldline following algorithm;	39
IX. miscellaneous/auxilliary subroutines	42

*Electronic address: shudson@pppl.gov

†Electronic address: suzuki.yasuhiro@lhd.nifs.ac.jp

I. USER SUPPLIED MAGNETIC FIELD

The user must provide a subroutine, `bfield(RpZ, itangent, BRpZ, ifail)`, which returns the magnetic field, \mathbf{B} , in cylindrical coordinates, (R, ϕ, Z) .

1. `RpZ(1:3)` is `real*8`; input;
 - i. contains the R , ϕ and Z coordinates at which the field, and possibly the derivatives, are required.
2. `itangent` is integer; input;
 - i. if `itangent=0` then only \mathbf{B} is required;
 - ii. if `itangent=1` then both \mathbf{B} and its derivatives are required.
3. `BRpZ(1:3,0:3)` is `real*8`; output;
 - i. The contravariant components of the magnetic field, namely $B^R \equiv \mathbf{B} \cdot \nabla R$, $B^\phi \equiv \mathbf{B} \cdot \nabla \phi$, and $B^Z \equiv \mathbf{B} \cdot \nabla Z$.
 - ii. The required format is

$$\begin{aligned} \text{BRpZ}(1,0) &= B^R, & \text{BRpZ}(1,1) &= \partial_R B^R, & \text{BRpZ}(1,2) &= \partial_\phi B^R, & \text{BRpZ}(1,3) &= \partial_Z B^R, \\ \text{BRpZ}(2,0) &= B^\phi, & \text{BRpZ}(2,1) &= \partial_R B^\phi, & \text{BRpZ}(2,2) &= \partial_\phi B^\phi, & \text{BRpZ}(2,3) &= \partial_Z B^\phi, \\ \text{BRpZ}(3,0) &= B^Z, & \text{BRpZ}(3,1) &= \partial_R B^Z, & \text{BRpZ}(3,2) &= \partial_\phi B^Z, & \text{BRpZ}(3,3) &= \partial_Z B^Z. \end{aligned}$$
 !!! Note that $B^\phi = \mathbf{B} \cdot \hat{\phi}/R$, and $\partial_R B^\phi = (\partial_R \mathbf{B} \cdot \hat{\phi} - B^\phi)/R$!!!
4. `ifail` is integer; output;
 - i. returns an error flag;
 - ii. `ifail=0` indicates that the calculation of \mathbf{B} was successful.

For many of the following subroutines, the periodicity of the field will be exploited, by which it is meant that the magnetic field must satisfy

$$\mathbf{B}(R, \phi + \Delta\phi, Z) = \mathbf{B}(R, \phi, Z), \quad (1)$$

where $\Delta\phi \equiv 2\pi/\text{Nfp}$, and `Nfp` is an integer that must be provided as required.

II. USER SUPPLIED COIL DESCRIPTION

The user must provide a subroutine, `iccoil(t, x, y, z, ifail)`, which returns the geometry of a closed curve embedded in three-dimensional space, i.e. $\mathbf{x} = x(t)\mathbf{i} + y(t)\mathbf{j} + z(t)\mathbf{k}$; where t is input (real), and $x(0:1)$, $y(0:1)$ and $z(0:1)$ are output (real). This routine is only used by `bs00aa`, and if `bs00aa` is not called a dummy routine can be provided.

III. MACRO EXPANSION AND COMPILATION

1. The `Oculus` package is available at <http://w3.pppl.gov/~shudson/Oculus/oculus.XX.tar>, where 20XX indicates the year (version).
2. The `oculus.h` file is converted to `oculus.F90` via `m4 -P oculus.macros oculus.h > oculus.F90`.
3. **On compilation, it is required to convert single precision to double precision.**
4. Presently, the NAG library is required. (Replacement routines are presently being implemented.)
5. At some time in the future, the routines will be kept under version control (perhaps under `github`).
6. Please inform `shudson@pppl.gov` of any errors; and suggestions and requests are very welcome!

IV. ERROR FLAG

1. Each subroutine has an input integer `ifail`.
2. On input: `ifail` controls the degree of screen output;
 - for `ifail.ge.0`, operation is “quiet”;
 - for `ifail.eq.0`, screen output is “terse”;
 - for increasingly negative `ifail` the screen output is increasingly “noisy”, which may be useful for debugging.
 - for maximum screen output set `ifail=-9`.
3. On output, `ifail=0` for normal execution.

V. BIOT-SAVART SUBROUTINES

In this section are described subroutines for computing the magnetic field produced by a current distribution.

A. bs00aa : compute the magnetic field produced by a filamentary current loop of arbitrary shape;

1. Given a closed (i.e. periodic), one-dimensional loop embedded in three-dimensional space, with position described by $\bar{\mathbf{x}}(t) \equiv \bar{x}(t)\mathbf{i} + \bar{y}(t)\mathbf{j} + \bar{z}(t)\mathbf{k}$, with the arbitrary curve parameter $t \in [0, 2\pi]$, and $\bar{\mathbf{x}}(t + 2\pi) = \bar{\mathbf{x}}(t)$, assumed to carry unit current, i.e. $I = 1$, the magnetic field at $\mathbf{x} \equiv x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ is given by the Biot-Savart integral,

$$\mathbf{B} \equiv \int_C \frac{d\mathbf{l} \times \mathbf{r}}{r^3}, \quad (2)$$

where $\mathbf{r} \equiv \mathbf{x} - \bar{\mathbf{x}}$.

2. In component form, Eq. (2) is

$$B^x \equiv \int_0^{2\pi} \frac{\dot{y}(z - \bar{z}) - \dot{z}(y - \bar{y})}{r^3} dt, \quad (3)$$

$$B^y \equiv \int_0^{2\pi} \frac{\dot{z}(x - \bar{x}) - \dot{x}(z - \bar{z})}{r^3} dt, \quad (4)$$

$$B^z \equiv \int_0^{2\pi} \frac{\dot{x}(y - \bar{y}) - \dot{y}(x - \bar{x})}{r^3} dt, \quad (5)$$

where $\dot{x} \equiv d\bar{x}/dt$, etc.

3. The magnetic vector potential is

$$\mathbf{A} \equiv \int_C \frac{d\mathbf{l}}{r}. \quad (6)$$

4. The total length of the curve is

$$L \equiv \int_C dl, \quad (7)$$

where $dl \equiv (\dot{x}^2 + \dot{y}^2 + \dot{z}^2)^{1/2}$.

5. The user must supply a subroutine, `iccoil`, that returns \bar{x} , \bar{y} & \bar{z} , and $d\bar{x}/dt$, $d\bar{y}/dt$ & $d\bar{z}/dt$, given t :
subroutine iccoil(t, x(0:1), y(0:1), z(0:1), ifail)
 where t , $x(0:1)$, $y(0:1)$ and $z(0:1)$ are real and `ifail` is an integer, $x(0) \equiv \bar{x}(t)$ and $x(1) \equiv \dot{x}(t)$, and similarly for y and z .
6. The integration is performed using [NAG:D01AJF](#). (This routine is based upon the QUADPACK routine QAGS, which is freely available.)

7. The user must include

use oculus, only : biotsavart, bs00aa

type(biotsavart) :: bsfield

in their source that calls `bs00aa`, where `biotsavart` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `bsfield`, is arbitrary.

8. Required inputs

bsfield%x : real ;

bsfield%y : real ;

bsfield%z : real ;

- i. position $\mathbf{x} \equiv x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ at which magnetic field is required;

bsfield%tol : real ;

bsfield%N : integer ;

- i. integration accuracy parameters provided to `D01AJF`; (`EPSABS=tol`, `EPSREL=zero` and `LW=4*N`);

bsfield%LB : logical ;

i. set LB = `.true.` to compute the magnetic field;

`bsfield%LA` : logical ;

i. set LA = `.true.` to compute the magnetic vector potential;

`bsfield%LL` : logical ;

i. set LL = `.true.` to compute the length of the curve;

9. Execution

`call bs00aa(bsfield, ifail)`

10. Outputs

`bsfield%Bx` : real ;

`bsfield%By` : real ;

`bsfield%Bz` : real ;

i. only if LB = `.true.`;

`bsfield%Ax` : real ;

`bsfield%Ay` : real ;

`bsfield%Az` : real ;

i. only if LA = `.true.`;

`bsfield%length` : real ;

i. only if LL = `.true.`;

`ifail` : integer ;

`ifail=0` : normal execution;

`ifail=1` : input error;

`ifail=2` : D01AJF encountered a divide-by-zero; this is only possible if $\exists t \in [0, 2\pi]$ such that $\mathbf{x} = \bar{\mathbf{x}}(t)$.

VI. “CYLINDRICAL” SUBROUTINES

In this section are described subroutines that do not depend explicitly on a pre-defined, ‘background’, toroidal coordinate framework.

A. ga00aa : find the magnetic axis;

1. Iterative fieldline tracing methods are used to find the magnetic axis, defined as the magnetic fieldline that closes on itself after a toroidal distance of $\Delta\phi = 2\pi/\text{Nfp}$, i.e. $\mathbf{x}(\Delta\phi) = \mathbf{x}(0)$, where Nfp is the field periodicity.
- * The fieldline mapping is defined by integrating along the magnetic field, and is constructed numerically in cylindrical coordinates by integrating the o.d.e.'s

$$\frac{dR(\phi)}{d\phi} = \frac{B^R(R, \phi, Z)}{B^\phi(R, \phi, Z)} \equiv \dot{R}(R, \phi, Z), \quad (8)$$

$$\frac{dZ(\phi)}{d\phi} = \frac{B^Z(R, \phi, Z)}{B^\phi(R, \phi, Z)} \equiv \dot{Z}(R, \phi, Z), \quad (9)$$

from an initial, user-supplied starting point, $(R_0, 0, Z_0)$. The toroidal angle, ϕ , is used as the integration parameter, and so B^ϕ cannot be zero. Upon request, this routine will be modified in order to follow field lines in regions where $B^\phi = 0$.

- * A Newton-iterative method is used to find the zero of

$$\mathbf{f} \begin{pmatrix} R_0 \\ Z_0 \end{pmatrix} \equiv \begin{pmatrix} R_1 - R_0 \\ Z_1 - Z_0 \end{pmatrix} \quad (10)$$

where $R_1 \equiv R(\Delta\phi)$ and $Z_1 \equiv Z(\Delta\phi)$.

- * Given an initial guess, $\mathbf{x} \equiv (R_0, Z_0)^T$, a better guess for the location of the axis, $(R_0, Z_0)^T + (\delta R, \delta Z)^T$, is given by the linear approximation

$$\mathbf{f} \begin{pmatrix} R_0 + \delta R_0 \\ Z_0 + \delta Z_0 \end{pmatrix} = \mathbf{f} \begin{pmatrix} R_0 \\ Z_0 \end{pmatrix} + \underbrace{\begin{pmatrix} \partial_{R_0} R_1 - 1 & , & \partial_{Z_0} R_1 \\ \partial_{R_0} Z_1 & , & \partial_{Z_0} Z_1 - 1 \end{pmatrix}}_{\nabla \mathbf{f}} \cdot \begin{pmatrix} \delta R_0 \\ \delta Z_0 \end{pmatrix} + \mathcal{O}(\delta^2) = 0, \quad (11)$$

and the correction is given by $\delta \mathbf{x} = -(\nabla \mathbf{f})^{-1} \cdot \mathbf{f}(\mathbf{x})$.

- * The derivatives, $\partial_{R_0} R_1$, $\partial_{Z_0} R_1$, etc. are determined by fieldline integration,

$$\frac{d}{d\phi} \begin{pmatrix} \partial_{R_0} R(\phi), & \partial_{Z_0} R(\phi) \\ \partial_{R_0} Z(\phi), & \partial_{Z_0} Z(\phi) \end{pmatrix} = \begin{pmatrix} \partial_R \dot{R}, & \partial_Z \dot{R} \\ \partial_R \dot{Z}, & \partial_Z \dot{Z} \end{pmatrix} \cdot \begin{pmatrix} \partial_{R_0} R(\phi), & \partial_{Z_0} R(\phi) \\ \partial_{R_0} Z(\phi), & \partial_{Z_0} Z(\phi) \end{pmatrix}, \quad (12)$$

from an initial starting point being the identity matrix,

$$\begin{pmatrix} \partial_{R_0} R(0), & \partial_{Z_0} R(0) \\ \partial_{R_0} Z(0), & \partial_{Z_0} Z(0) \end{pmatrix} = \begin{pmatrix} 1, & 0 \\ 0, & 1 \end{pmatrix}. \quad (13)$$

- * The iterative search is enabled by [NAG:C05PBF](#).
- * If `ifail=-4`, then instead the axis search is provided by [NAG:C05NBF](#), which does not require derivatives. Note that for this option, the tangent map, the transform on axis, and the residue will not be calculated.
- * The above definition of the magnetic axis does not have a unique solution: an $\iota = 1/1$ fieldline also satisfies this definition, as does the $\iota = 2/1, 3/1$, etc., as also does the ‘‘X’’ point at the separatrix. Furthermore, during a sawteeth cycle, the $\iota = 1/1$ fieldline and the original magnetic axis swap places. If there is a continuous family of ‘‘magnetic axes’’, e.g. there exists an intact $q = 1$ surface, then $\nabla \mathbf{f}$ will not be invertible (unless singular value decomposition methods are used). Thus, this routine should be used with care. Which closed field line that `ga00aa` locates is determined by the initial guess provided.

- * The returned information includes:

- i. the Fourier representation of $R(\phi)$ and $Z(\phi)$;
- ii. the tangent-mapping near the axis, which allows the rotational-transform on axis to be determined;
- iii. Greene’s residue [Greene, *J. Math. Phys.* 20, 1183 (1979)] calculated at the magnetic axis (determines stability).

2. The user must include

`use oculus, only : magneticaxis, ga00aa`

`type(magneticaxis) :: axis`

in their source that calls `ga00aa`, where `axis` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `axis`, is arbitrary.

3. Required inputs

axis%Nfp : integer ;

- i. the toroidal periodicity of the magnetic field, e.g. Nfp=1;

axis%Ntor : integer ;

- i. the desired Fourier resolution of the magnetic axis,
- ii. if it is not required to have a Fourier decomposition of the magnetic axis, or the magnetic field is axisymmetric, choose Ntor=0;

axis%R : real ;

- i. guess for the R location of the magnetic axis on the $\phi = 0$ plane;

axis%Z : real ;

- i. guess for the Z location of the magnetic axis on the $\phi = 0$ plane;

axis%maxits : integer ;

- i. max. iterations allowed in search;
- ii. e.g. maxits=16;

axis%tol : real ;

- i. required accuracy to which the position of the magnetic axis on the $\phi = 0$ plane is required
- ii. e.g. tol=1.0e-06;

axis%odetol : real ;

- i. o.d.e. integration tolerance;
- ii. e.g. odetol=1.0e-08;

ifail : integer ;

- i. if ifail.ge. 1 : there is no screen output, *except* if there is an input error;
- ii. if ifail.le. 0 : a one-line summary is provided, giving the (R, Z) location of the axis, t_{axis} , etc.
- iii. if ifail.le.-1 : the Fourier harmonics of the magnetic axis are displayed;
- iv. if ifail.le.-2 : the eigenvalues and eigenvectors of the tangent map at the axis are displayed;
- iv. if ifail.le.-3 : information detailing the progress of the iterative search is provided;
- iv. if ifail.le.-4 : C05NBF (which does not require derivatives) is used instead of C05PBF;

4. Execution

call ga00aa(axis, ifail)

5. Outputs

axis%R : real ;

- i. updated;

axis%Z : real ;

- i. updated;

axis%tangent(1:2,1:2) : real ;

- i. the tangent mapping at axis;
- ii. if the eigenvalues of the tangent map are imaginary, e.g. $\lambda \equiv \alpha + \beta i$, then the rotational-transform on axis satisfies $\tan(\|t\|) = \beta/\alpha$, where $\|t\| \equiv t \bmod 2\pi$.
- iii. if the eigenvalues of the tangent map are real, then the eigenvalues give the direction of the stable and unstable manifolds.

axis%wr(1:2) : real ;

axis%wi(1:2) : real ;

`axis%vr(1:2,1:2)` : real ;
`axis%vi(1:2,1:2)` : real ;
 i. the eigenvalues and eigenvectors of the tangent mapping at axis;

`axis%iota` : real ;
 i. rotational-transform on axis
 ii. will only be meaningful if axis is stable, which is indicated by both (i) the sign of residue, and (ii) whether the eigenvalues and eigenvectors are real or imaginary.

`axis%Lallocated` : integer ;
 i. if `Lallocated=1`, the `Ri`, `Zi`, `Rnc`, `Rns`, `Zns`, `Znc` have been allocated;
 ii. if `Lallocated=0`, the `Ri`, `Zi`, `Rnc`, `Rns`, `Zns`, `Znc` have **not** been allocated;

`axis%Ri(0:4*Ntor)` : real ;
 i. the magnetic axis, $R(i\Delta\varphi)$, for $i = 0, 4*Ntor$, where $\Delta\varphi = \Delta\phi/(4*Ntor)$;
 ii. `Ri` is allocated internally; if on input `Ri` is already allocated it will first be deallocated; similarly for `Zi`

`axis%Zi(0:4*Ntor)` : real ;
 i. the magnetic axis, $Z(i\Delta\varphi)$, for $i = 0, 4*Ntor$, where $\Delta\varphi = \Delta\phi/(4*Ntor)$;

`axis%Rnc(0:Ntor)` : real ;
 i. the Fourier harmonics, $R(\phi) = \sum_n [R_{n,c} \cos(-n\phi) + R_{n,s} \sin(-n\phi)]$;
 ii. `Rnc` is allocated internally; if on input `Rnc` is already allocated it will first be deallocated; similarly for `Zns`, `Rns` and `Znc`.

`axis%Zns(0:Ntor)` : real ;
 i. the Fourier harmonics, $Z(\phi) = \sum_n [Z_{n,c} \cos(-n\phi) + Z_{n,s} \sin(-n\phi)]$;

`axis%Rns(0:Ntor)` : real ;
 i. the Fourier harmonics, $R(\phi) = \sum_n [R_{n,c} \cos(-n\phi) + R_{n,s} \sin(-n\phi)]$;

`axis%Znc(0:Ntor)` : real ;
 i. the Fourier harmonics, $Z(\phi) = \sum_n [Z_{n,c} \cos(-n\phi) + Z_{n,s} \sin(-n\phi)]$;

`axis%error` : real ;
 i. the error, $\sqrt{\Delta R^2 + \Delta Z^2}$, where $\Delta R \equiv R(\Delta\phi) - R_0$ and $\Delta Z \equiv Z(\Delta\phi) - Z_0$.

`axis%its` : integer ;
 i. the number of iterations required;

`axis%residue` : real ;
 i. Greene's residue of the magnetic axis; [Greene, *J. Math. Phys.* **20**, 1183 (1979)];

`axis%rzf(0:2,0:31)` : real ;
 i. Information regarding the progress of the iterations;
 ii. $R_i = \text{axis\%rzf}(0,0:\text{axis\%its})$ are the R values used in the iterations;
 ii. $Z_i = \text{axis\%rzf}(1,0:\text{axis\%its})$ are the Z values used in the iterations;
 ii. $F_i = \text{axis\%rzf}(2,0:\text{axis\%its})$ are the $|f|$ at each iteration;

`ifail` : integer ;
 i. on output:
 `ifail=0` : normal execution;
 `ifail=1` : input error;
 `ifail=2` : the routine [NAG:C05PBF](#) failed to locate the zero of the function, perhaps because of a failure in integrating along the fieldlines;

`ifail=3` : the NAG routine `C06EAF` failed to construct the Fourier harmonics of the axis, for either R or Z ;

`ifail=4` : the NAG routine `F02EBF` failed to construct the eigenvalues/vectors of the tangent mapping;

6. Comments:

- * The NAG routine `NAG:C05PBF` is used for the nonlinear root find, and `tol` is given directly to `NAG:C05PBF`.
- * The NAG routine `D02BJF` is used for the o.d.e. integration, and `odetol` is supplied directly to `D02BJF`.
- * If a good initial guess is given, this number should be small, as Newton methods should converge rapidly; however, if there are multiple magnetic axes (as during a sawtooth event) then the Newton method may encounter problems; also, numerical errors in the magnetic field (perhaps $\nabla \cdot \mathbf{B}$ is not exactly zero) can cause the fieldline integration to be inaccurate, and so it may be difficult to find the solution to the desired accuracy.
- * Please also consider using `ec00aa`;

B. ho00aa : find the homoclinic points (of the stable/unstable manifold);

1. This subroutine performs two iterative searches.

* The first search is for the unstable fixed point, $\bar{\mathbf{x}}$. Iterative fieldline tracing methods are used, and the numerical method is identical to that used in ga00aa, see [Sec. VI A](#).

* The second search is to find the homoclinic points. Homoclinic points are defined as those points that approach the unstable fixed point forwards in time *and* backwards in time, i.e. $T^n(\mathbf{x}) \rightarrow \bar{\mathbf{x}}$ as $n \rightarrow \pm\infty$, where $T(\mathbf{x})$ is the image of \mathbf{x} under the Poincaré map, and $T^{-1}(\mathbf{x})$ is the pre-image, and where time is analogous to the toroidal angle.

* The tangent mapping at the fixed point is constructed. The unstable manifold is identified by the unstable eigenvector, \mathbf{v}_u , which is that eigenvector with a real eigenvalue, λ_u , with magnitude greater than unity. The stable manifold is identified by the stable eigenvector, \mathbf{v}_s , which is that eigenvector with a real eigenvalue, λ_s , with magnitude less than unity. Note that $\lambda_u\lambda_s = 1$.

* The numerical task is then to find (d_u, d_s) such that

$$\mathbf{f}(d_u, d_s) \equiv T^{+i}(\bar{\mathbf{x}} + d_u\mathbf{v}_u) - T^{-j}(\bar{\mathbf{x}} + d_s\mathbf{v}_s) = 0, \quad (14)$$

where the (d_u, d_s) must be sufficiently small so that the linear approximation is valid (required as the eigenvectors of the tangent map are used to identify the stable and unstable manifolds).

* There are a countable infinity of homoclinic points (and they come in families, and they are increasingly close together near the unstable fixed point). The integers i and j are used to identify which homoclinic points are located and are determined as part of the calculation. The solution for (d_u, d_s) depends on the i and j .

2. The user must include

use oculus, only : homoclinictangle, ho00aa

type(homoclinictangle) :: tangle

in their source that calls ho00aa, where `tangle` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `tangle`, is arbitrary.

3. Required inputs

`tangle%Nfp` : integer ;

i. the toroidal periodicity of the magnetic field,

ii. e.g. `Nfp=1`;

`tangle%R` : real ;

i. guess for the R location of the unstable fixed point on the $\phi = 0$ plane;

`tangle%Z` : real ;

i. guess for the Z location of the unstable on the $\phi = 0$ plane;

`tangle%maxits` : integer ;

i. max. iterations allowed in search;

ii. e.g. `maxits=16`;

`tangle%xtol` : real ;

i. required accuracy to which the position of the unstable fixed point on the $\phi = 0$ plane is required

ii. e.g. `xtol=1.0e-06`;

`tangle%odetol` : real ;

i. o.d.e. integration tolerance;

ii. e.g. `odetol=1.0e-08`;

`tangle%dU` : real ;

i. guess for the displacement along the unstable manifold of a homoclinic point;

ii. e.g. `dU = 10-2`;

tangle%dS : real ;

- i. guess for the displacement along the stable manifold of a homoclinic point;
- ii. e.g. dS = 10^{-2} ;

tangle%htol : real ;

- i. required accuracy to which the homoclinic point is required;
- ii. e.g. htol=1.0e-05;

tangle%ltol : real ;

- i. required accuracy to which the linear approximation is required;
- ii. e.g. ltol=1.0e-03;

ifail : integer ;

4. Execution

call ho00aa(tangle, ifail)

5. Outputs

tangle%R : real ;

- i. updated;

tangle%Z : real ;

- i. updated;

tangle%tangent(1:2,1:2) : real ;

- i. the tangent mapping at unstable fixed point;
- ii. if the eigenvalues of the tangent map are imaginary, e.g. $\lambda \equiv \alpha + \beta i$, then the rotational-transform on tangle satisfies $\tan(\|\epsilon\|) = \beta/\alpha$, where $\|\epsilon\| \equiv \epsilon \pmod{2\pi}$.
- iii. if the eigenvalues of the tangent map are real, then the eigenvalues give the direction of the stable and unstable manifolds.

tangle%wr(1:2) : real ;

tangle%wi(1:2) : real ;

tangle%vr(1:2,1:2) : real ;

tangle%vi(1:2,1:2) : real ;

- i. the eigenvalues and eigenvectors of the tangent mapping at the fixed point;

tangle%error : real ;

- i. the error, $\sqrt{\Delta R^2 + \Delta Z^2}$, where $\Delta R \equiv R(\Delta\phi) - R_X$ and $\Delta Z \equiv Z(\Delta\phi) - Z_X$.

tangle%hits : integer ;

- i. the number of iterations required;

tangle%residue : real ;

- i. Greene's residue of the fixed point; [Greene, [J. Math. Phys. 20, 1183 \(1979\)](#)];

tangle%hits : integer ;

- i. the number of iterations required to locate the homoclinic points;

tangle%herror : real ;

- i. the error in locating the homoclinic points;

tangle%ilobe(1:2) : integer ;

- i. the i and j as defined above, Eq. (14);

`tangle%maxilobe` : integer ;

i. just for convenience; `maxilobe=max(ilobe(1:2))`;

`tangle%Lallocated` : integer ;

- i. if `Lallocated=1`, the `hpoints` array has been allocated;
- ii. if `Lallocated=0`, the `hpoints` array has **not** been allocated;

`tangle%hpoints(1:2,0:ltangle%maxilobe,1:2)` : real ;

- i. the locations of the homoclinic points;
- ii. the homoclinic points on the unstable branch are $R_{i,u} \equiv \text{hpoints}(1,i,1)$ and $Z_{i,u} \equiv \text{hpoints}(2,i,1)$, for $i = 0, \text{ilobe}(1)$;
- iii. the homoclinic points on the stable branch are $R_{i,s} \equiv \text{hpoints}(1,i,1)$ and $Z_{i,s} \equiv \text{hpoints}(2,i,2)$, for $i = 0, \text{ilobe}(2)$;
- iv. `hpoints` need not be allocated on input; if it is, it is immediately deallocated;

`tangle%lerror(1:2)` : real ;

- i. the error in the linear approximation;

`ifail` : integer ;

- i. on output:

`ifail=0` : normal execution;

`ifail=1` : input error;

`ifail=2` : the NAG routine C05PBF failed to locate the zero of the 'fixed-point' function, perhaps because of a failure in integrating along the fieldlines;

`ifail=3` : the fixed point is not 'unstable': the residue must be negative;

`ifail=4` : the NAG routine F02EBF failed to construct the eigenvalues/vectors of the tangent mapping;

`ifail=5` : the fixed point is not 'unstable': the eigenvalues must be real;

`ifail=6` : at least one eigenvector has illegal magnitude;

`ifail=7` : failed to fieldline integration to determine `ilobe`;

`ifail=8` : `ilobe(1).le.0` or `ilobe(2).eq.0` ;

`ifail=9` : failed to locate the homoclinic points;

6. Comments:

- * The NAG routine C05PBF is used for the nonlinear root find, and `tol` is given directly to C05PBF.
- * The NAG routine D02BJF is used for the o.d.e. integration, and `odetol` is supplied directly to D02BJF.
- * If a good initial guess is given, this number should be small, as Newton methods should converge rapidly; however, if there are multiple magnetic axes (as during a sawtooth event) then the Newton method may encounter problems; also, numerical errors in the magnetic field (perhaps $\nabla \cdot \mathbf{B}$ is not exactly zero) can cause the fieldline integration to be inaccurate, and so it may be difficult to find the solution to the desired accuracy.
- * Please also consider using `ec00aa` to find the unstable fixed point;

C. ec00aa : find action extremizing curves using global integration;

1. Find curves that extremize the action integral,

$$S[\mathcal{C}] \equiv \int_{\mathcal{C}} \mathbf{A} \cdot d\mathbf{l}. \quad (15)$$

Note: for a given vector potential, $\mathbf{B} = \nabla \times \mathbf{A}$, the action, S , is considered to be a function of the ‘trial-curve’, \mathcal{C} .

* From variational calculus, the variation in S due to variations, $\delta d\mathbf{l}$, in the curve is

$$\delta S = \int_{\mathcal{C}} \mathbf{B} \times d\mathbf{l} \cdot \delta d\mathbf{l}, \quad (16)$$

from which we see that curves that extremize S satisfy $\mathbf{B} \times d\mathbf{l} = 0$, i.e. the extremal curves are parallel to \mathbf{B} .

* This method of determining magnetic fieldlines is called Lagrangian or global integration.

* Working in cylindrical coordinates, an arbitrary trial curve is represented by

$$\mathbf{x}(\phi) = R(\phi)\mathbf{e}_R(\phi) + Z(\phi)\mathbf{e}_Z, \quad (17)$$

where $\mathbf{e}_R(\phi) \equiv \cos(\phi)\mathbf{i} + \sin(\phi)\mathbf{j}$ and $\mathbf{e}_Z \equiv \mathbf{k}$.

* The infinitesimal change in \mathbf{x} due to an infinitesimal increase in ϕ is

$$d\mathbf{l} = \left(\dot{R}\mathbf{e}_R + \mathbf{e}_\phi + \dot{Z}\mathbf{e}_Z \right) d\phi, \quad (18)$$

where the ‘dot’ denotes the derivative with respect to ϕ .

* It is assumed that the magnetic vector potential is in the form

$$\mathbf{A} \equiv A_R \nabla R + A_\phi \nabla \phi + A_Z \nabla Z. \quad (19)$$

Interestingly, and fortunately, only the curl of the vector potential will be required.

* The ‘Lagrangian’, i.e. the integrand $\mathbf{A} \cdot d\mathbf{l}/d\phi$, is $A_R \dot{R} + A_\phi + A_Z \dot{Z}$.

* A Fourier representation for $R(\phi)$ and $Z(\phi)$ is employed:

$$R \equiv \sum_{n=0}^N [R_{n,c} \cos(n\phi) + R_{n,s} \sin(n\phi)], \quad (20)$$

$$Z \equiv \sum_{n=0}^N [Z_{n,c} \cos(n\phi) + Z_{n,s} \sin(n\phi)]. \quad (21)$$

* Extremal curves are curves for which the derivative of S with respect to to the $R_{n,c}$, $R_{n,s}$ etc. are zero. We have, for example,

$$\frac{\partial S}{\partial R_{n,c}} = \int_0^{2\pi} \left(\frac{\partial A_R}{\partial R} \frac{\partial R}{\partial R_{n,c}} \dot{R} + A_R \frac{\partial}{\partial R_{n,c}} \dot{R} + \frac{\partial A_\phi}{\partial R} \frac{\partial R}{\partial R_{n,c}} + \frac{\partial A_Z}{\partial R} \frac{\partial R}{\partial R_{n,c}} \dot{Z} \right) d\phi. \quad (22)$$

* Consider the term

$$A_R \frac{\partial}{\partial R_{n,c}} \dot{R} = A_R \frac{\partial}{\partial R_{n,c}} \frac{d}{d\phi} R = A_R \frac{d}{d\phi} \frac{\partial}{\partial R_{n,c}} R = A_R \frac{d}{d\phi} \cos(n\phi) \quad (23)$$

Rather than writing $d_\phi \cos(n\phi) = -n \sin(n\phi)$, a very neat trick is to instead use integration-by-parts to write $A_R d_\phi \cos(n\phi) \equiv -d_\phi A_R \cos(n\phi)$. Note that d_ϕ is the total derivative with respect to ϕ , so that $d_\phi A_R \equiv \partial_R A_R \dot{R} + \partial_\phi A_R + \partial_Z A_R \dot{Z}$. Several terms cancel, and only the derivatives of \mathbf{A} are now required, rather than \mathbf{A} itself; in fact, it is the components of $\nabla \times \mathbf{A}$ that appear!

* The derivatives of the action with respect to the parameters defining the curve are

$$\frac{\partial S}{\partial R_{n,c}} = \int_0^{2\pi} \cos(n\phi) \left(B^Z - B^\phi \dot{Z} \right) R d\phi, \quad (24)$$

$$\frac{\partial S}{\partial R_{n,s}} = \int_0^{2\pi} \sin(n\phi) \left(B^Z - B^\phi \dot{Z} \right) R d\phi, \quad (25)$$

$$\frac{\partial S}{\partial Z_{n,c}} = \int_0^{2\pi} \cos(n\phi) \left(B^\phi \dot{R} - B^R \right) R d\phi, \quad (26)$$

$$\frac{\partial S}{\partial Z_{n,s}} = \int_0^{2\pi} \sin(n\phi) \left(B^\phi \dot{R} - B^R \right) R d\phi. \quad (27)$$

* Various numerical methods may be employed to find $F_R \equiv (B^Z - B^\phi \dot{Z}) R = 0$ and $F_Z \equiv (B^\phi \dot{R} - B^R) R = 0$.

2. The user must include

use oculus, only : extremizingcurve, ec00aa

type(extremizingcurve) :: curve

in their source that calls ec00aa. The variable name, curve, is arbitrary.

3. Required inputs

curve%Nfp : integer ;

- i. the toroidal periodicity of the magnetic field,
- ii. e.g. Nfp=1;

curve%Ntor : integer ;

- i. the required Fourier resolution in the toroidal direction;
- ii. constraint: Ntor.ge.0;

curve%Rnc(0:Ntor) : real ;

curve%Rns(0:Ntor) : real ;

curve%Znc(0:Ntor) : real ;

curve%Zns(0:Ntor) : real ;

- i. an initial guess for the Fourier harmonics of the extremizing curve;
- ii. these are allocatable, and must be allocated *before* calling ec00aa;

curve%etol : real ;

- i. the accuracy to which the extremal curve is required;
- ii. e.g. etol=10⁻⁶;

curve%ftol : real ;

- i. the accuracy to which the extremal curve is required; only if gradient flow is used;
- ii. e.g. ftol=10⁻³;

curve%maxits : integer ;

- i. maximum number of iterations allowed;

curve%emethod : integer ;

- i. determines the numerical method used to locate extremal curves:
- ii. emethod = 0 : uses Newton method to find $F_R = 0$ and $F_Z = 0$;

curve%odetol : real ;

- i. the o.d.e. integration tolerance;
- ii. e.g. odetol=10⁻⁶;
- iii. only used if emethod=1,2,3,4;

curve%tauend : real ;

- i. the upper limit on the o.d.e. integration;
- ii. e.g. tauend=1.00;
- iii. only used if emethod=1,2,3,4;

curve%dtau : real ;

- i. the intermediate output on the o.d.e. integration;
- ii. e.g. dtau=0.05;

iii. only used if emethod=1,2,3,4;

ifail : integer ;

4. Execution

call ec00aa(curve, ifail)

5. Outputs

curve%Rnc(0:Ntor) : real ;

curve%Rns(0:Ntor) : real ;

curve%Znc(0:Ntor) : real ;

curve%Zns(0:Ntor) : real ;

i. updated;

curve%its : integer ;

i. required iterations;

curve%err : real ;

i. accuracy achieved;

ifail : integer ;

i. on output:

ifail=0 : normal execution;

ifail=1 : input error;

ifail=2 : the routine C05NBF failed to find a solution;

6. Comments:

* The NAG routine C05NBF is used;

D. tr00aa : measure rotational-transform;

1. Fieldline tracing methods are used to determine the relative rotational-transform of one fieldline about a given “reference” fieldline, which will usually be a magnetic axis.

* The equations governing the fieldlines are the same as that given in Eq. (8) and Eq. (9).

* The user must supply *two* starting points, (R_a, Z_a) and (R, Z) . The location of the magnetic axis, (R_a, Z_a) , can be obtained from a previous call to `ga00aa`, see [Sec. VI A](#). however, the values of (R_a, Z_a) and (R, Z) are completely arbitrary. What is really measured by this routine is average “linking” of one fieldline about the other.

* A poloidal angle, $\theta(\phi)$, is introduced as

$$\tan \theta(\phi) = \frac{\delta Z(\phi)}{\delta R(\phi)}, \quad (28)$$

where $\delta R(\phi) = R(\phi) - R_a(\phi)$ and $\delta Z(\phi) = Z(\phi) - Z_a(\phi)$.

* This angle varies with ϕ according to

$$\frac{d\theta}{d\phi} = \frac{\delta R(Z' - Z'_a) - \delta Z(R' - R'_a)}{\delta R^2 + \delta Z^2}, \quad (29)$$

where \prime denotes total derivative with respect to ϕ .

* The o.d.e. integration defined in Eq. (29) may be initialized with $\theta(0) = 0$, and after a sufficiently large distance, $\Delta\phi$, this angle satisfies $\Delta\theta \approx \epsilon\Delta\phi$, where ϵ is the rotational-transform.

* A more accurate calculation of ϵ is enabled by fitting a straight line to $\theta(\phi)$, rather than just subtracting the endpoints. This will be implemented in time, upon request, . . .

* Formally, the rotational-transform is defined as the limit

$$\epsilon \equiv \lim_{\Delta\phi \rightarrow \infty} \frac{\Delta\theta}{\Delta\phi}. \quad (30)$$

This limit *only* converges on regular fieldlines. For irregular, or chaotic, fieldlines, this limit does not converge and the rotational-transform is not defined!

* Note that Eq. (29) requires knowledge of $R_a(\phi)$ and $Z_a(\phi)$, and $R(\phi)$ and $Z(\phi)$, and these are obtained by integrating Eq. (8) and Eq. (9). So, in total there are 5 coupled o.d.e.s.

2. The user must include

```
use oculus, only : transformdata, tr00aa
```

```
type(transformdata) :: transform
```

in their source that calls `tr00aa`, where `transform` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `transform`, is arbitrary.

3. Required inputs

```
transform%Nfp          : integer ;
```

i. the toroidal periodicity of the magnetic field, e.g. `Nfp=1` for tokamaks, `Nfp=5` for LHD, . . . ;

```
transform%Ppts         : integer ;
```

i. the number of toroidal transits that the fieldlines will be followed, e.g. `Ppts>100`;

```
transform%odetol       : real    ;
```

i. o.d.e. integration tolerance; e.g. `odetol=1.0e-08`;

```
transform%Ra           : real    ;
```

```
transform%Za           : real    ;
```

```
transform%R            : real    ;
```

```
transform%Z            : real    ;
```

- i. starting points for fieldline integrations;
- ii. usually, (R_a, Z_a) will be the location of the magnetic axis on the $\phi = 0$ plane, and (R, Z) is arbitrary.
- iii. the starting points must be distinct: in particular, $\text{length} \equiv (R - R_a)^2 + (Z - Z_a)^2$ must exceed 10^{-12} .

4. Execution

```
call tr00aa( transform, ifail )
```

5. Outputs

```
transform%iota          : real    ;
```

- i. the “rotational-transform” of the fieldline starting at (R, Z) relative to the fieldline starting at (R_a, Z_a) .

```
transform%Lallocated    : integer ;
```

- i. if `Lallocated=1`, the `transform%RZ` array has been allocated;
- ii. if `Lallocated=0`, the `transform%RZ` array has **not** been allocated;

```
transform%RZ(1:2,0:Ppts) : real   ;
```

- i. the Poincaré plot data: $R_i \equiv \text{transform}\%RZ(1, i)$, $Z_i \equiv \text{transform}\%RZ(2, i)$
- ii. the `transform%RZ` need not be allocated on input; if it is, it is first deallocated;

```
ifail                   : integer ;
```

- i. on output:

```
ifail=0 : normal execution;
```

```
ifail=1 : input error;
```

```
ifail=2 : the NAG routine D02BJF failed to integrate along the fieldline. perhaps because the fieldline left the computational domain, . . .
```

E. pp00aa : fieldline tracing for Poincaré plot, calculate Lyapunov exponent;

1. This subroutine follows a magnetic fieldline from a given starting point, for a given number of toroidal periods, either forwards or backwards in the toroidal angle, ϕ . The intersection points of the fieldline with the Poincaré section $\phi = 0$ are returned.

* The (maximum) Lyapunov exponent, λ , may also be calculated [Benettin, Galgani & Strelcyn, *Phys. Rev. A*, 14:2338 (1976)]. This measures the average exponential rate of separation of a nearby trajectory,

$$|\delta\mathbf{x}(\phi)| = e^{\lambda\phi}|\delta\mathbf{x}(0)|, \quad (31)$$

as $\phi \rightarrow \infty$ and as $|\delta\mathbf{x}(0)| \rightarrow 0$.

* The limit $|\delta\mathbf{x}(0)| \rightarrow 0$ is best treated by linearizing the fieldline equations about the given reference fieldline, as is hereafter assumed.

* To calculate λ , assuming that $\delta\mathbf{x}$ lies in the tangent space and $|\delta\mathbf{x}(0)| = 1$, as

$$\lambda(\phi) = \frac{1}{\phi} \log(|\delta\mathbf{x}(\phi)|). \quad (32)$$

* In Eq. (32), λ has been expressed as a function of ϕ so that the user may examine whether the limit $\lim_{\phi \rightarrow \infty} \lambda(\phi)$ has converged.

* If $\lambda > 0$, the long-time trajectory of the fieldline is infinitesimally sensitive to the initial position, and this is a defining characteristic of chaos; however, not all fieldlines are chaotic. Consider a nearby fieldline that separates linearly, i.e. $|\delta\mathbf{x}(\phi)| = 1 + c\phi$ for some constant, c ; for example, a fieldline that lies on a nearby flux surface with slightly-different rotational-transform. The value of λ given by Eq. (32) gives

$$\lambda \approx \frac{\log c}{\phi} + \frac{\log \phi}{\phi}. \quad (33)$$

* To distinguish a weakly-exponentially-separating fieldline from a linearly-separating fieldline may require a very long integration in ϕ . It is recommended to plot $\log(|\lambda(\phi)|)$ against $\log(\phi)$ to determine if $\lambda(\phi)$ is approaching a non-zero value as $\phi \rightarrow \infty$; and it may be useful to compare this to what would be expected, Eq. (33), for linearly separating fieldlines.

* Also, there maybe nearby fieldlines that neither exponentially separate nor linearly separate, but instead oscillate about the reference fieldline. Such behavior is displayed by fieldlines in the vicinity of a stable fixed point.

2. The user must include

use oculus, only : poincaredata, pp00aa

type(poincaredata) :: poincare

in their source that calls pp00aa, where poincaredata is a derived type (i.e. structure) that contains both the required input and output information. The variable name, poincare, is arbitrary.

3. Required inputs

poincare%Nfp : integer ;

i. the toroidal periodicity of the magnetic field,

ii. e.g. Nfp=1;

poincare%R : real ;

i. starting point;

poincare%Z : real ;

i. starting point;

poincare%Ppts : integer ;

i. toroidal periods / iterations;

poincare%idirection : integer ;

- i. `idirection= 1` : follow fieldline in increasing toroidal direction;
- i. `idirection=-1` : follow fieldline in decreasing toroidal direction;

`poincare%flparameter` : integer ;

- i. `flparameter=0` : use toroidal angle to parameterize distance along fieldline; i.e. divide equations by B^{ζ} ;
- i. `flparameter=1` : use length to parameterize distance along fieldline; i.e. divide equations by $|B|$; under construction;

`poincare%iLyapunov` : integer ;

- i. `iLyapunov = 0` : the Lyapunov exponent will not be calculated;
- ii. `iLyapunov = 1` : the Lyapunov exponent not be calculated; note that this will slow the fieldline integration, as additional o.d.e.s that define the tangent mapping need to be integrated;

`poincare%Ltol` : real ;

- i. tolerance required for calculation of Lyapunov exponent;
- ii. e.g. `Ltol=1.0e-03`;
- iii. under construction: I plan to automatically adjust the integration length (i.e. `Ppts`) to ensure that λ is converged to within `Ltol`.

`poincare%odetol` : real ;

- i. o.d.e. integration tolerance;
- ii. e.g. `odetol=1.0e-08`;

`ifail` : integer ;

4. Execution

call `pp00aa(poincare, ifail)`

5. Outputs

`poincare%RZ(1:2,0:Ppts)` : real ;

- i. Poincaré data;
- ii. $R_i = \text{RZ}(1,0:\text{Ppts})$; $Z_i = \text{RZ}(2,0:\text{Ppts})$;
- ili. `RZ` need not be allocated on input; if it is, it is immediately deallocated;

`poincare%Ly(1:Ppts)` : real ;

- i. “evolution” of Lyapunov exponent; i.e. the estimate of the Lyapunov exponent as a function of iteration;
- iv. `Ly` need not be allocated on input; if it is, it is immediately deallocated;

`poincare%Lallocated` : integer ;

- i. if `Lallocated=1`, the `RZ` and `Ly` arrays have been allocated;
- ii. if `Lallocated=0`, the `RZ` and `Ly` arrays have not been allocated;

`poincare%Lyapunov` : real ;

- i. the Lyapunov exponent;

`poincare%ipts` : integer ;

- i. toroidal periods actually followed: if an error is encountered (e.g. the fieldline leaves the computational domain, the fieldline integration is terminated);

i. on output:

- `ifail=0` : normal execution;
- `ifail=1` : input error;

6. Comments:

* The NAG routine [NAG:D02BJF](#) is used to perform the o.d.e. integration.

F. gc00aa : follow guiding center;

1. This is under construction. Contact shudson@pppl.gov.

G. rz00aa : construct cylindrical Fourier harmonics of flux surface using fieldline tracing;

1. This subroutine follows a magnetic fieldline from a given starting point, for a given number of toroidal periods, to construct the Fourier coefficients that satisfy

$$R = \sum_i R_i \cos(m_i \theta - n_i \zeta) \quad (34)$$

$$Z = \sum_i Z_i \sin(m_i \theta - n_i \zeta) \quad (35)$$

2. The user must include

use oculus, only : poincaredata, rz00aa

type(poincaredata) :: poincare

in their source that calls rz00aa, where poincaredata is a derived type (i.e. structure) that contains both the required input and output information. The variable name, poincare, is arbitrary.

3. Required inputs

poincare%Nfp : integer ;

- i. the toroidal periodicity of the magnetic field,
- ii. e.g. Nfp=1;

poincare%R : real ;

- i. starting point;

poincare%Z : real ;

- i. starting point;

poincare%Ppts : integer ;

- i. toroidal periods / iterations;

poincare%odetol : real ;

- i. o.d.e. integration tolerance;
- ii. e.g. odetol=1.0e-08;

ifail : integer ;

4. Execution

call rz00aa(poincare, ifail)

5. Outputs

- i. on output:

ifail=0 : normal execution;

ifail=1 : input error;

6. Comments:

*

H. ad00aa : anisotropic diffusion using locally-field-aligned coordinates;

1. This subroutine integrates in time the anisotropic-diffusion equation,

$$\frac{\partial p}{\partial t} = \nabla \cdot (\kappa_{\parallel} \nabla_{\parallel} p + \kappa_{\perp} \nabla_{\perp} p) + S, \quad (36)$$

where p is a scalar function of position, e.g. the pressure; S is a scalar function of position, e.g. a source; t is an arbitrary integration parameter, e.g. time; the directional derivative along the magnetic field is $\nabla_{\parallel} p \equiv \mathbf{b} \cdot \nabla p$; and $\nabla_{\perp} p \equiv \nabla p - \nabla_{\parallel} p$.

* This equation may be re-written as $\partial_t p = \nabla \cdot [(\kappa_{\parallel} - \kappa_{\perp}) \nabla_{\parallel} p + \kappa_{\perp} \nabla p] + S$, and hereafter will use $\bar{\kappa}_{\parallel} \equiv \kappa_{\parallel} - \kappa_{\perp}$.

* [Presently, $\bar{\kappa} = \kappa_{\parallel}$, i.e. the $\kappa_{\perp} \nabla_{\parallel} p$ term is ignored.]

* It is more transparent to write the anisotropic-diffusion equation as

$$\frac{\partial p}{\partial t} = \bar{\kappa}_{\parallel} \mathbf{B} \cdot \nabla \frac{1}{B^2} \mathbf{B} \cdot \nabla p + \kappa_{\perp} \nabla \cdot \nabla p + S. \quad (37)$$

* The operator $\mathbf{B} \cdot \nabla$ reduces to $B^{\phi} \partial_{\phi}$ if coordinates, (α, β, ϕ) , can be constructed so that $\mathbf{B} = \nabla \alpha \times \nabla \beta$ and $\nabla \alpha \times \nabla \beta \cdot \nabla \phi = B^{\phi}$. Such coordinates can always be constructed locally by fieldline integration, *provided* $B^{\phi} \neq 0$. The benefit of this approach is to minimize the “parallel-pollution” of the perpendicular diffusion.

* A regular, cylindrical grid is employed, $R \in [R_{min}, R_{max}]$, $\phi \in [0, 2\pi/N_P]$ and $Z \in [Z_{min}, Z_{max}]$, where $N_P \equiv \text{Nfp}$ is the field periodicity; with grid resolution N_R , N_{ϕ} and N_Z . The location of each grid point, $\mathbf{x}_{i,j,k} = R_i \mathbf{e}_R + Z_j \mathbf{e}_Z$ is given by $R_i \equiv R_{min} + i\Delta R$, $Z_j \equiv Z_{min} + j\Delta Z$ and $\phi_k = k\Delta\phi$; where $\Delta R = (R_{max} - R_{min})/N_R$, $\Delta Z = (Z_{max} - Z_{min})/N_Z$ and $\Delta\phi = (2\pi/N_P)/N_{\phi}$.

* A second-order discretization of the *second* parallel derivative is obtained by differencing the *first* parallel derivatives on the “forward half- $\Delta\phi$ ” and the “backward half- $\Delta\phi$ ” grids as follows:

$$\nabla_{\parallel}^2 p|_{i,j,k} \equiv \frac{B^{\phi}}{\Delta\phi} \Big|_0 \left(\frac{B^{\phi}}{B^2} \Big|_{+1/2} \frac{p_{+1} - p_0}{\Delta\phi} - \frac{B^{\phi}}{B^2} \Big|_{-1/2} \frac{p_0 - p_{-1}}{\Delta\phi} \right), \quad (38)$$

where $p_0 \equiv p(\mathbf{x}_{i,j,k}) = p_{i,j,k}$ and $p_{\pm 1} \equiv p(\mathcal{M}^{\pm 1}(\mathbf{x}_{i,j,k}))$, where $\mathcal{M}^1(\mathbf{x}_{i,j,k})$ is the image of $\mathbf{x}_{i,j,k}$ under the “forward” fieldline mapping from $\phi = k\Delta\phi$ to $\phi = (k+1)\Delta\phi$, and $\mathcal{M}^{-1}(\mathbf{x}_{i,j,k})$ is the image of $\mathbf{x}_{i,j,k}$ under the “backward” fieldline mapping from $\phi = k\Delta\phi$ to $\phi = (k-1)\Delta\phi$; and the factors B^{ϕ}/B^2 are evaluated on the forward and backward half- $\Delta\phi$ grids.

* [Presently, $B^{\phi} = 1$ and $B^{\phi}/B^2 = 1$, i.e. the “metric” information is ignored.]

* The $\mathcal{M}^{\pm 1}(\mathbf{x}_{i,j,k})$ do not generally coincide with grid points, so $p(\mathcal{M}^{\pm 1}(\mathbf{x}_{i,j,k}))$ must be determined by interpolation. Given that the $\mathcal{M}^{\pm 1}(\mathbf{x}_{i,j,k})$ do however lie on the toroidal planes $(k+1)\Delta\phi$ and $(k-1)\Delta\phi$, an interpolation in ϕ is not required. The simplest interpolation is the second-order, bi-linear interpolation, e.g.

$$p_{+1} \equiv (1-y) [(1-x) p_{I,J,k+1} + x p_{I+1,J,k+1}] + y [(1-x) p_{I,J+1,k+1} + x p_{I+1,J+1,k+1}], \quad (39)$$

where the I and J label the left-lower grid point, i.e. $R_I \leq \bar{R} < R_{I+1}$ and $Z_J \leq \bar{Z} < Z_{J+1}$; and $x \equiv (\bar{R} - R_I)/\Delta R$ and $y \equiv (\bar{Z} - Z_J)/\Delta Z$,

* The bi-linear interpolation is used near the computational boundary, and a fourth-order, bi-cubic interpolation is used in the interior domain.

* The fieldline tracing information is contained in the $I_{\pm 1,i,j,k}$, $J_{\pm 1,i,j,k}$, $x_{\pm 1,i,j,k}$ and the $y_{\pm 1,i,j,k}$ arrays; and the $B^{\phi}|_0$ and $B^{\phi}/B^2|_{\pm 1/2}$ information is contained in $B_{-1:1,,i,j,k}$.

* The discretization of the second, parallel derivative given in Eq. (38) is only possible if $B^{\phi} \neq 0$; as the equations defining the fieldline mapping are $d_{\phi} R \equiv B^R/B^{\phi}$ and $d_{\phi} Z \equiv B^Z/B^{\phi}$, and the Jacobian of the locally-field-aligned coordinates is $1/B^{\phi}$. An additional logical array, $F_{i,j,k}$ indicates whether the fieldline tracing construction of the locally-field-aligned coordinates was successful. For all points where the fieldline tracing was *not* successful the pressure at the “mapped” points is set to zero, i.e. $p_{\pm} = 0$.

* All of these arrays, i.e. the I , J , x , y , B and F , are returned by **ad00aa**. Note that these arrays depend only on the magnetic field and the computational grid. If the user wishes to continue to relax the pressure *with the same magnetic field and with the same computational grid*, then this information can be passed back to **ad00aa** on a subsequent call, and this will eliminate the computational cost of re-constructing the locally-field-aligned coordinates. (Some computational savings can also be made if the user wishes to refine the grid in only the R and Z directions for the same magnetic field, but this option is not yet implemented.)

* If, however, on a subsequent call to `ad00aa`, the magnetic field has changed, the I , J , x , y , B and F arrays will not be consistent with the new magnetic field. The input flag `Lcontrol` determines how this information is to be used.

* The “non-directional diffusion” term, $\nabla \cdot \nabla p$, is given in cylindrical coordinates as

$$\nabla \cdot \nabla p = \frac{1}{R} \left[\frac{\partial}{\partial R} \left(R \frac{\partial p}{\partial R} \right) + \frac{\partial}{\partial \phi} \left(\frac{1}{R} \frac{\partial p}{\partial R} \right) + \frac{\partial}{\partial Z} \left(R \frac{\partial p}{\partial Z} \right) \right]. \quad (40)$$

* This is simplified to

$$\nabla \cdot \nabla p = \frac{\partial^2 p}{\partial R^2} + \frac{\partial^2 p}{\partial Z^2}, \quad (41)$$

i.e., metric information is ignored.

* A fourth-order discretization of Eq. (41) is employed.

* After constructing the parallel and perpendicular derivatives, the pressure is advanced explicitly via

$$p_{i,j,k}^{n+1} = p_{i,j,k}^n + \Delta t \left(\bar{\kappa}_{\parallel} \nabla_{\parallel}^2 p_{i,j,k}^n + \kappa_{\perp} \nabla \cdot \nabla p_{i,j,k}^n + S_{i,j,k} \right) \quad (42)$$

for $n = 1, N$ time-steps.

* It is possible, upon request, to implement an implicit time advance.

* The boundary condition is that $p = 0$ on the computational boundary defined by R_{min} , R_{max} , and Z_{min} , Z_{max} , and this boundary condition is enforced internally.

* The iterations will terminate if p becomes either too small or too large, which presumably indicate that the CFL condition on the explicit integration timestep has been violated. It is assumed that p will be order unity.

2. The user must include

use `oculus`, only : `pressurerelax`, `ad00aa`

`type(pressurerelax) :: pressure`

in their source that calls `ad00aa`, where `pressurerelax` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `pressure`, is arbitrary.

3. Required inputs

`pressure%Nfp` : integer ;

i. the toroidal periodicity of the magnetic field, e.g. `Nfp=1`;

`pressure%NR` : integer ;

`pressure%Np` : integer ;

`pressure%NZ` : integer ;

i. grid resolution in R ; constraint `NR` ≥ 4 ;

ii. grid resolution in ϕ ; constraint `Np` ≥ 1 ;

iii. grid resolution in Z ; constraint `NZ` ≥ 4 ;

`pressure%Rmin` : real ;

`pressure%Rmax` : real ;

`pressure%Zmin` : real ;

`pressure%Zmax` : real ;

i. calculation domain in R ; constraint `Rmax` $>$ `Rmin`; `Rmin` $>$ ϵ ;

ii. calculation domain in Z ; constraint `Zmax` $>$ `Zmin`;

`pressure%Lcontrol` : integer ;

- i. if `Lcontrol = 1`, : only construct the locally-field-aligned coordinates by fieldline tracing; the information describing the locally-field-aligned coordinates are returned in arrays `x`, `y`, `I`, `J`, `B` and `F`; if these arrays are allocated on input, they are first de-allocated; note that the pressure and source are not referenced, and so they need not be allocated;
- ii. if `Lcontrol = 2`, : relax pressure by integrating Eq. (42), assuming that the locally-field-aligned coordinate information is provided; the `x`, `y`, `I`, `J`, `B` and `F` arrays returned by a previous call to `ad00aa` must be provided; note that these arrays depend on the magnetic field and the computational grid, so this option should only be used if these have not changed since the last call to `ad00aa`; the pressure and the source must be allocated on input;
- iii. if `Lcontrol = 3`, : first construct the locally-field-aligned coordinates as for `Lcontrol=1`, then relax the pressure as for `Lcontrol=2`; only the pressure and source must be allocated on input;

`pressure%Lode` : integer ;

- i. `Lode = 0` : a single 4-th order, Runge-Kutta o.d.e. integration step will be performed to construct the field-aligned coordinates between adjacent toroidal planes; this will invariably be faster than using option `Lode=1` (i.e. require fewer evaluations of the magnetic field) but this option will only be reliable if $\Delta\phi$ is sufficiently small, i.e. `Np` is sufficiently large;
- ii. `Lode = 1` : the NAG routine `D02BFJ` will be used to more-accurately integrate along the magnetic field to construct the field-aligned coordinates;

`pressure%odetol` : real ;

- i. o.d.e. integration tolerance provided to `D02BFJ` for integrating along the magnetic field;
- ii. only used if `Lode = 1`;
- iii. the accuracy of the calculation is limited by N_R and N_Z , so `odetol` need not be too small; suggested `odetol=10-5`.

`pressure%p(0:NR,0:NZ,-1:Np)` : real ;

- i. initial state for the pressure, where $p(i, j, k) \equiv p(R_{min} + i\Delta R, k\Delta\phi, Z_{min} + j\Delta Z)$;
- ii. this must be allocated *before* calling `ad00aa` if `Lcontrol=2,3`;
- iii. a suitable initialization is

$$p = (1 - x^2)(1 - y^2), \quad (43)$$

where $x \equiv 2(R - R_{mid})/(R_{max} - R_{min})$ where $R_{mid} \equiv (R_{min} + R_{max})/2$; and similarly for y ; or, more simply, $p = 1$.

- iv. on exit, this array will be updated;

`pressure%s(0:NR,0:NZ, 0:Np-1)` : real ;

- i. source, where $s(i, j, k) \equiv s(R_{min} + i\Delta R, k\Delta\phi, Z_{min} + j\Delta Z)$;
- ii. this must be allocated *before* calling `ad00aa` if `Lcontrol=2,3`;

`pressure%Ntime` : integer ;

- i. number of time steps taken; only if `Lcontrol=2,3` ; e.g. `Ntime=10000`;

`pressure%dttime` : real ;

- i. `dttime` $\equiv \Delta t$ appearing in Eq. (42); only if `Lcontrol=2,3`;

`pressure%kpara` : real ;

`pressure%kperp` : real ;

- i. the parallel and perpendicular diffusion coefficients;

`ifail` : integer ;

- i. if `ifail = 1`, quiet mode;
- i. if `ifail = 0`, screen output is terse;
- i. if `ifail = -1`,

- i. if `ifail = -2`, some information regarding the relaxation will be shown at every 10000 timesteps;
- i. if `ifail = -3`, some information regarding the relaxation will be shown at every 1000 timesteps;
- i. if `ifail = -4`, some information regarding the relaxation will be shown at every 100 timesteps;
- i. if `ifail = -5`, some information regarding the relaxation will be shown at every 10 timesteps;
- i. if `ifail = -6`, some information regarding the relaxation will be shown at every 1 timesteps;

4. Execution

```
call ad00aa( pressure, ifail )
```

5. Outputs

```
pressure%x( 0:1,1:NR,1:NZ,0:Np-1) : real    ;
pressure%y( 0:1,1:NR,1:NZ,0:Np-1) : real    ;
pressure%I( 0:1,1:NR,1:NZ,0:Np-1) : integer ;
pressure%J( 0:1,1:NR,1:NZ,0:Np-1) : integer ;
pressure%B(-1:1,1:NR,1:NZ,0:Np-1) : real    ;
pressure%F(      1:NR,1:NZ,0:Np-1) : logical ;
```

- i. information describing the locally-field-aligned coordinates;
- ii. if on a subsequent call to `ad00aa` the user chooses to set `Lcontrol= 2`, then these arrays are required on input to the subsequent call of `ad00aa`;

i. on output:

```
ifail=0 : normal execution;
ifail=1 : input error;
ifail=2 : the pressure has exploded, probably because the CFL condition on the timestep has been violated;
```

6. Comments:

- * The steady state solution has $\mathcal{O}(p/S) \sim \mathcal{O}(\bar{\kappa}_{\parallel}/\kappa_{\perp})$, where S is the source; so that if $p \sim \mathcal{O}(\kappa_{\parallel})$ then $S \sim \mathcal{O}(\kappa_{\perp})$.
- * This subroutine is not yet parallelized, but it is easy to do so.

I. `bn00aa` : compute $(\mathbf{B} \cdot \mathbf{n})_{m,n}$ on a given toroidal surface;

- Given a toroidal, ‘control’ surface,

$$\mathbf{x}(\theta, \zeta) \equiv R(\theta, \zeta) \cos \zeta \mathbf{i} + R(\theta, \zeta) \sin \zeta \mathbf{j} + Z(\theta, \zeta) \mathbf{k}, \quad (44)$$

we may compute

$$B^n(\theta, \zeta) \equiv \mathbf{B} \cdot \mathbf{e}_\theta \times \mathbf{e}_\zeta. \quad (45)$$

- With the magnetic field given in cylindrical coordinates, $\mathbf{B} \equiv B^R \mathbf{e}_R + B^\phi \mathbf{e}_\phi + B^Z \mathbf{e}_Z$,

$$\mathbf{B} \cdot \mathbf{e}_\theta \times \mathbf{e}_\zeta = [B^R(-Z_\theta R) + B^\phi(Z_\theta R_\zeta - R_\theta Z_\zeta) + B^Z(R_\theta R)] R. \quad (46)$$

- The routine also returns the toroidal and poloidal currents, defined as surface integrals through appropriate surfaces,

$$I \equiv \int_S \mathbf{j} \cdot d\mathbf{s} = \int_{\partial S} \mathbf{B} \cdot d\mathbf{l} = \int_0^{2\pi} B_\theta d\theta, \quad (47)$$

$$G \equiv \int_S \mathbf{j} \cdot d\mathbf{s} = \int_{\partial S} \mathbf{B} \cdot d\mathbf{l} = \int_0^{2\pi} B_\zeta d\zeta. \quad (48)$$

These integrals are calculated using [NAG:D01AHF](#).

7. The user must include

```
use oculus, only : bnormal, bn00aa
```

```
type(bnormal) :: bn
```

in their source that calls `bn00aa`, where `bn` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `bn`, is arbitrary.

3. Required inputs

```
bn%cmn                : integer ;
```

- i. total number of Fourier harmonics that describe the input control surface;

```
bn%cim(1:cmn)         : integer array ;
```

```
bn%cin(1:cmn)         : integer array ;
```

- i. Fourier mode identification;

```
bn%Rcc(1:cmn)         : real array;
```

```
bn%Rcs(1:cmn)         : real array;
```

```
bn%Zcc(1:cmn)         : real array;
```

```
bn%Zcs(1:cmn)         : real array;
```

- i. Fourier harmonics of control surface;

```
bn%Mpol               : integer;
```

```
bn%Ntor               : integer;
```

```
bn%Nfp                : integer;
```

- i. Poloidal and toroidal Fourier resolution, and field-periodicity;

```
bn%tol                : real;
```

- i. Relative accuracy required; `tol` \equiv EPSR provided to [NAG:D01AHF](#).

```
ifail                 : integer ;
```

- i. if `ifail` = 1, quiet mode;

- i. if `ifail` = 0, screen output is terse;

i. if ifail = -1,

4. Execution

```
call bn00aa( bn, ifail )
```

5. Outputs

```
bn%mn           : integer      ;
```

```
bn%im(1:mn)     : integer array;
```

```
bn%in(1:mn)     : integer array;
```

```
bn%gBc(1:mn)    : real      array;
```

```
bn%gBs(1:mn)    : real      array;
```

i. Fourier harmonics of normal field to control surface;

```
bn%Itor         : real;
```

```
bn%Gpol         : real;
```

i. enclosed currents;

i. on output:

```
ifail=0 : normal execution;
```

```
ifail=1 : input error;
```

```
ifail=2 : the integration of the toroidal current failed;
```

```
ifail=3 : the integration of the linking current failed;
```

VII. TOROIDAL-CYLINDRICAL COORDINATE-VECTOR TRANSFORMATION

1. Specializing to coordinate transformations of the form

$$\begin{aligned} R &= R(\rho, \theta, \zeta), \\ \phi &= \zeta, \\ Z &= Z(\rho, \theta, \zeta), \end{aligned} \quad (49)$$

where position is given $\mathbf{x} \equiv R \cos \zeta \mathbf{i} + R \sin \zeta \mathbf{j} + Z \mathbf{k}$, the induced vector transformation is

$$\begin{pmatrix} B^R \\ B^\phi \\ B^Z \end{pmatrix} = \begin{pmatrix} R_\rho & R_\theta & R_\zeta \\ 0 & 0 & 1 \\ Z_\rho & Z_\theta & Z_\zeta \end{pmatrix} \begin{pmatrix} B^\rho \\ B^\theta \\ B^\zeta \end{pmatrix}. \quad (50)$$

2. This matrix is invertible if $\Delta \equiv R_\theta Z_\rho - R_\rho Z_\theta \neq 0$, and the coordinate Jacobian is $\sqrt{g} \equiv R(R_\theta Z_\rho - R_\rho Z_\theta) = R\Delta$.

3. The coordinate transformation is written:

$$\begin{aligned} R &= \sum_{m,n} [R_{m,n}(0) + \lambda_{m,n}(\rho) X_{m,n}(\rho)] \cos(m\theta - n\zeta), \\ Z &= \sum_{m,n} [Z_{m,n}(0) + \lambda_{m,n}(\rho) Y_{m,n}(\rho)] \sin(m\theta - n\zeta); \end{aligned} \quad (51)$$

where the $X_{m,n}(\rho)$ and $Y_{m,n}(\rho)$ are cubic-splines (see `0culus:nr00aa` and `0culus:nr00ab` for details on the cubic-spline interpolation).

4. The regularization factors are given by

$$\lambda_{m,n}(\rho) = \begin{cases} 1 & , \text{ if } m = 0, \\ \bar{\rho}^{-\bar{m}} & , \text{ if } m > 0, \end{cases} \quad (52)$$

where e.g. $\bar{m} \equiv \min(m, 2)$; and $\bar{\rho} \equiv \rho/V$, where V is a normalization factor, e.g. $V \equiv \text{total volume}$. The following constraints must be enforced:

- i. for $m \neq 0$: $R_{m,n}(0) = 0$ and $Z_{m,n}(0) = 0$, and $X_{m,n}(0)$ and $Y_{m,n}(0)$ are arbitrary;
- ii. for $m = 0$: $X_{m,n}(0) = 0$ and $Y_{m,n}(0) = 0$;

The summation over m and n includes only the $\{(m, n) : m = 0; n = 0, N\}$ and $\{(m, n) : m = 1, M; n = -N, N\}$ harmonics, which may be called the ‘‘VMEC convention’’.

5. Eq. (51) is for ‘‘stellarator-symmetric’’ equilibria [Dewar & Hudson, [Physica D 112 \(1998\) 275](#)]. For arbitrary geometry, additional sine and cosine harmonics are added.

6. Inverting the vector transformation yields

$$\begin{pmatrix} \sqrt{g} B^\rho \\ \sqrt{g} B^\theta \\ \sqrt{g} B^\zeta \end{pmatrix} = \begin{pmatrix} -Z_\theta & R_\zeta Z_\theta - R_\theta Z_\zeta & +R_\theta \\ +Z_\rho & R_\rho Z_\zeta - R_\zeta Z_\rho & -R_\rho \\ 0 & R_\theta Z_\rho - R_\rho Z_\theta & 0 \end{pmatrix} \begin{pmatrix} RB^R \\ RB^\phi \\ RB^Z \end{pmatrix}. \quad (53)$$

7. The coordinates are ‘‘right-handed’’ if $\sqrt{g} > 0$, and ‘‘left-handed’’ if $\sqrt{g} < 0$. This can have important implications, particularly for `qf00aa` below. If the coordinate transformation is $R = R_{0,0} + \rho \cos \theta$ and $Z = +\rho \sin \theta$, then the coordinates are left-handed; and if the coordinate transformation is $R = R_{0,0} + \rho \cos \theta$ and $Z = -\rho \sin \theta$, then the coordinates are right-handed. Usually, the sign of Jacobian is opposite to the sign of $Z_{1,0}$.

A. bc00aa : interpolation of toroidal surfaces; construction;

1. Given a discrete set of (closed) toroidal surfaces, which can be described by a set of Fourier harmonics for R and Z in suitable angles, a continuous coordinate framework that is consistent with that described in [Sec. VII](#) can be constructed by a suitable interpolation.

- Most of the description in [Sec. VII](#) applies. Some loose ends are:
 - for $m \neq 0$, if $L_{rad} = 1$, then $X_{m,n}(0) = X_{m,n}(\rho_1)$, and similarly for $Y_{m,n}$ etc.;
 - for $m \neq 0$, if $L_{rad} > 1$, then $X_{m,n}(0) = (X_{m,n}(\rho_1)\rho_2 - X_{m,n}(\rho_2)\rho_1)/(\rho_2 - \rho_1)$, and similarly for $Y_{m,n}$ etc.;
 - for the cubic-spline interpolations the end-point derivatives are required, and these are approximated using one-sided, first-order differences, i.e. $X'_{m,n}(0) = (X_{m,n}(\rho_1) - X_{m,n}(0))/(\rho_1 - \rho_0)$, etc.
- The radial coordinate is the volume, which is calculated for each toroidal surface by the integral

$$V = \int_{\mathcal{V}} dv = \frac{1}{3} \int_{\mathcal{V}} \nabla \cdot \mathbf{x} dv = \frac{1}{3} \int_{\mathcal{S}} \mathbf{x} \cdot d\mathbf{s} = \frac{1}{3} \int_0^{2\pi} d\theta \int_0^{2\pi/N} d\zeta \quad |\mathbf{x} \cdot \mathbf{x}_\theta \times \mathbf{x}_\zeta|^s \quad (54)$$

where we have used $\nabla \cdot \mathbf{x} = 3$, and have assumed that the domain is periodic in the angles.

- Using $\mathbf{x} \cdot \mathbf{e}_\theta \times \mathbf{e}_\zeta = R(ZR_\theta - RZ_\theta)$,

$$\begin{aligned} V &= \frac{1}{3} \int_0^{2\pi} d\theta \int_0^{2\pi/N} d\zeta R(ZR_\theta - RZ_\theta) \\ &= \frac{1}{3} \sum_i \sum_j \sum_k R_{e,i} (Z_{e,j}R_{o,k} - R_{e,j}Z_{o,k}) (+m_k) \iint d\theta d\zeta \cos \alpha_i \cos \alpha_j \cos \alpha_k \\ &\quad + \frac{1}{3} \sum_i \sum_j \sum_k R_{e,i} (Z_{o,j}R_{e,k} - R_{o,j}Z_{e,k}) (-m_k) \iint d\theta d\zeta \cos \alpha_i \sin \alpha_j \sin \alpha_k \\ &\quad + \frac{1}{3} \sum_i \sum_j \sum_k R_{o,i} (Z_{e,j}R_{e,k} - R_{e,j}Z_{e,k}) (-m_k) \iint d\theta d\zeta \sin \alpha_i \cos \alpha_j \sin \alpha_k \\ &\quad + \frac{1}{3} \sum_i \sum_j \sum_k R_{o,i} (Z_{o,j}R_{o,k} - R_{o,j}Z_{o,k}) (+m_k) \iint d\theta d\zeta \sin \alpha_i \sin \alpha_j \cos \alpha_k \end{aligned} \quad (55)$$

where $\alpha_i \equiv m_i\theta - n_i\zeta$.

- Triple angle expansions are used to simplify the trigonometric terms as follows:

$$\begin{aligned} 4 \cos \alpha_i \cos \alpha_j \cos \alpha_k &= +\cos(\alpha_i + \alpha_j + \alpha_k) + \cos(\alpha_i + \alpha_j - \alpha_k) + \cos(\alpha_i - \alpha_j + \alpha_k) + \cos(\alpha_i - \alpha_j - \alpha_k) \\ 4 \cos \alpha_i \sin \alpha_j \sin \alpha_k &= -\cos(\alpha_i + \alpha_j + \alpha_k) + \cos(\alpha_i + \alpha_j - \alpha_k) + \cos(\alpha_i - \alpha_j + \alpha_k) - \cos(\alpha_i - \alpha_j - \alpha_k) \\ 4 \sin \alpha_i \cos \alpha_j \sin \alpha_k &= -\cos(\alpha_i + \alpha_j + \alpha_k) + \cos(\alpha_i + \alpha_j - \alpha_k) - \cos(\alpha_i - \alpha_j + \alpha_k) + \cos(\alpha_i - \alpha_j - \alpha_k) \\ 4 \sin \alpha_i \sin \alpha_j \cos \alpha_k &= -\cos(\alpha_i + \alpha_j + \alpha_k) - \cos(\alpha_i + \alpha_j - \alpha_k) + \cos(\alpha_i - \alpha_j + \alpha_k) + \cos(\alpha_i - \alpha_j - \alpha_k) \end{aligned} \quad (56)$$

- The cubic-spline interpolations are performed using `Oculus:nr00aa` and `Oculus:nr00ab`, which will be described elsewhere.

2. The user must include

use `oculus, only : coordinates, bc00aa`

`type(coordinates) :: rzmnn`

in their source that calls `bc00aa`, where `coordinates` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `rzmnn`, is arbitrary.

3. Required inputs

`rzmnn%mn` : integer ;

- i. Number of Fourier harmonics used to describe the toroidal surfaces;

`rzmnn%im(1:mn)` : integer ;

- i. Poloidal mode numbers: cannot be negative;

`rzmnn%in(1:mn)` : integer ;

i. Toroidal mode numbers: for $\text{im}(j)=0$ must have $\text{in}(j).\text{ge}.0$;

`rzmn%Lrad` : integer ;

i. Number of surfaces to be interpolated; must have $\text{Lrad}.\text{ge}.1$;

`rzmn%Rbc(0:Lrad,1:mn)` : integer ;

`rzmn%Rbs(0:Lrad,1:mn)` : integer ;

`rzmn%Zbc(0:Lrad,1:mn)` : integer ;

`rzmn%Zbs(0:Lrad,1:mn)` : integer ;

i. Fourier harmonics of the toroidal surfaces; these arrays must be allocated and assigned before calling `bc00aa`.

ii. The surfaces must be provided in increasing volume, and they must be “nested”.

iii. The coordinate transformation is given by Eq. (50), with the j -th coordinate surface being

$$R_j(\theta, \zeta) \equiv \sum_{i=1}^{\text{mn}} \text{Rbc}[j, i] \cos(\text{im}[i]\theta - \text{in}[i]\zeta), \quad (57)$$

$$Z_j(\theta, \zeta) \equiv \sum_{i=1}^{\text{mn}} \text{Zbs}[j, i] \sin(\text{im}[i]\theta - \text{in}[i]\zeta), \quad (58)$$

and similarly for the non-stellarator-symmetric terms.

iv. Note that the 0-th surface is the degenerate surface \equiv the coordinate axis, for which $\text{Rbc}[0, i] = 0$ and $\text{Zbs}[0, i] = 0$ for $\text{im}[i] \neq 0$.

`rzmn%mm` : integer ;

i. Maximum regularization factor; e.g. $\text{mm}=2$;

`ifail` : integer ;

i. if `ifail` = 1, quiet mode;

i. if `ifail` = 0, screen output is terse;

i. if `ifail` = -1,

4. Execution

`call bc00aa(rzmn, ifail)`

5. Outputs

i. on output:

`rzmn%ss(0:Lrad)` : real ;

i. Volume of each surface; will be used as radial coordinate; this is under construction: perhaps a factor of 2π is missing.

`rzmn%Xbc(0:Lrad,-1:2,1:mn)` : real ;

`rzmn%Xbs(0:Lrad,-1:2,1:mn)` : real ;

`rzmn%Ybc(0:Lrad,-1:2,1:mn)` : real ;

`rzmn%Ybs(0:Lrad,-1:2,1:mn)` : real ;

i. interpolating cubic-splines consistent with Eq. (51);

ii. if these are allocated on input they will first be de-allocated, and then re-allocated.

iii. note that the spline *extrapolation* past the outermost surface is particularly unreliable if (i) there are more than $\text{Lrad}=1$ surface to be interpolated; or (ii) the extrapolation is approaching the separatrix, and recall that a separatrix bounds all confined plasmas.

`ifail` : integer ;

`ifail=0` : normal execution;

`ifail=1` : input error;

`ifail=2` : the $m = 0$ harmonics are not zero at the origin;

`ifail=3` : volume is not an increasing function of the surface label;

6. Comments:

i. The subroutine `bc00ab` will, given the interpolation coefficients, calculate the coordinate transformation and derivatives as required.

B. bc00ab : interpolation of toroidal surfaces; evaluation;

1. This routine evaluates the coordinate transformation defined by bc00aa and provides the metric elements, Jacobian etc. It can be called after bc00aa.

2. **The user must include**

```
use oculus, only : coordinates, bc00ab
```

```
type(coordinates)  :: rzmn
```

in their source that calls bc00ab, where `coordinates` is a derived type (i.e. structure) that contains both the required input and output information. The variable name, `rzmn`, is arbitrary.

VIII. “TOROIDAL” SUBROUTINES

In this section are described subroutines that depend on the toroidal coordinates described in [Sec. VII](#).

A. aa00aa : construct vector potential in toroidal coordinates;

1. The vector potential, $\mathbf{B} = \nabla \times \mathbf{A}$, may be constructed in toroidal coordinates, (ρ, θ, ζ) , by radial integration.
2. The toroidal coordinates must be provided in a suitable format, and it is suggested that `bc00aa` be called prior to calling `aa00aa`. The magnetic field, $\mathbf{B} = B^R \mathbf{e}_R + B^\phi \mathbf{e}_\phi + B^Z \mathbf{e}_Z$, is assumed to be given in cylindrical coordinates; and the coordinate transformation, $\mathbf{x}(\rho, \theta, \zeta)$, as given in Eq. (49) will be assumed.
3. A general representation for the vector potential is $\mathbf{A} = \bar{A}_\rho \nabla \rho + \bar{A}_\theta \nabla \theta + \bar{A}_\zeta \nabla \zeta$. A gauge function, $g(\rho, \theta, \zeta)$, may be chosen to simplify this: choosing $\partial_\rho g = -\bar{A}_\rho$, the \bar{A}_ρ component is cancelled leaving $\mathbf{A} = A_\theta \nabla \theta + A_\zeta \nabla \zeta$. The equation $\mathbf{B} = \nabla \times \mathbf{A}$ becomes

$$\begin{aligned} \partial_\theta A_\zeta - \partial_\zeta A_\theta &= \sqrt{g} B^\rho, \\ -\partial_\rho A_\zeta &= \sqrt{g} B^\theta, \\ \partial_\rho A_\theta &= \sqrt{g} B^\zeta. \end{aligned} \quad (59)$$

4. The Fourier harmonics of the components of the vector potential can be determined by radial integration:

$$A_{\theta,m,n}(\rho) = A_{\theta,m,n}(0) + \int_0^\rho (\sqrt{g} B^\zeta)_{m,n}(\bar{\rho}) d\bar{\rho}, \quad (60)$$

$$A_{\zeta,m,n}(\rho) = A_{\zeta,m,n}(0) - \int_0^\rho (\sqrt{g} B^\theta)_{m,n}(\bar{\rho}) d\bar{\rho}. \quad (61)$$

5. Solving these two equations automatically satisfies the second and third equations in Eq. (59). The first equation in Eq. (59) is satisfied if and only if (i) $\nabla \cdot \mathbf{B} = 0$, and (ii) the ‘integration constants’ $A_\theta(0, \theta, \zeta)$ and $A_\zeta(0, \theta, \zeta)$ satisfy $\partial_\theta A_\zeta(0, \theta, \zeta) - \partial_\zeta A_\theta(0, \theta, \zeta) = (\sqrt{g} B^\rho)(0, \theta, \zeta)$. Note that $(\sqrt{g} B^\rho)(0, \theta, \zeta) = 0$, and this allows the remaining gauge freedom, namely $g(0, \theta, \zeta)$, to be used to set $A_\theta(0, \theta, \zeta) = 0$ and $A_\zeta(0, \theta, \zeta) = 0$.
6. The ‘background’ toroidal coordinates should usually be provided by `bc00aa`, and the domain of integration is $\rho \in [0, V]$, where $V \equiv \max(\rho)$.
7. The coordinate interpolation is evaluated using `nr00ab`, the spline construction to the Fourier harmonics of the vector potential are constructed using `nr00aa`, and the required FFTs are evaluated using `ft02aa`.
8. After calling `aa00aa`, the routine `aa00ab` may be called to evaluate the vector potential at an arbitrary point within the computational domain.
9. **The user must include**

use oculus, only : coordinates, vectorpotential, aa00aa

```
type(coordinates)      :: rzmnn
type(vectorpotential) :: Atzmnn
```

in their source that calls `aa00aa`, where `coordinates` and `vectorpotential` are derived types (i.e. structures) that contains both the required input and output information. The variable names, `rzmnn` and `Atzmnn`, are arbitrary.

10. Required inputs

```
rzmnn          : structure ;
```

- i. the background toroidal coordinates, (ρ, θ, ζ) , that will be used;
- ii. it is assumed that before calling `aa00aa`, that the routine `bc00aa` has been called, and `rzmnn` should be unchanged;

```
Atzmnn%Nfp     : integer   ;
```

- i. the field periodicity, which must be the same as the field periodicity of the coordinates;

```
Atzmnn%Lrad    : integer   ;
```

- i. the required radial resolution;

```
Atzmnn%Mpol    : integer   ;
```

```
Atzmnn%Ntor    : integer   ;
```

i. the required Fourier resolution;

4. Execution

```
call aa00aa( rzmn, Atzmn, ifail )
```

11. Outputs

```
Atzmn%gBtc(0:Lrad,-1:2,1:amn) : real ;
```

```
Atzmn%gBts(0:Lrad,-1:2,1:amn) : real ;
```

```
Atzmn%gBzc(0:Lrad,-1:2,1:amn) : real ;
```

```
Atzmn%gBzs(0:Lrad,-1:2,1:amn) : real ;
```

i. the Fourier harmonics of the vector potential;

```
Atzmn%gBstz(1:Ntz,1:3,0:Lrad) : real ;
```

i. the vector potential on a regular grid in real space ;

B. aa00ba : evaluate vector potential (in toroidal coordinates);

1. Required inputs

`Atzmn` : structure ;

i. the vector potential returned by `aa00aa`;

`stz(1:3)` : real ;

i. the required position;

C. `ir00aa` : construct irrational flux-surface (incomplete);

1. Required inputs

`Atzmn` : structure ;

- i. the vector potential returned by `aa00aa`;

D. qf00aa : construct quadratic-flux minimizing surface using pseudo fieldline following algorithm;

1. Quadratic-flux minimizing (QFM) surfaces are constructed using the pseudo-fieldline integration algorithm. For a full description of QFM-surfaces please refer to the recent article by Hudson & Suzuki, [Phys. Plasmas, 21:102505, 2014](#) and references therein. Only a brief outline will be given here.
2. The “pseudo-field” is defined in toroidal coordinates (ρ, θ, ζ) as

$$\mathbf{B}_\nu \equiv \mathbf{B} - \nu B^\zeta \mathbf{e}_\rho, \quad (62)$$

where ν is constant along a pseudo fieldline but is otherwise a-priori unknown.

3. A (p, q) -QFM surface is a family of (p, q) -periodic fieldlines of \mathbf{B}_ν , along each of which the action-gradient is constant, i.e. $\nu = \nu(\alpha)$ where α labels pseudo fieldlines.
4. The task of finding these periodic pseudo fieldlines is equivalent to finding fixed points of the q -th return *pseudo*-map, P^q , which is constructed by integrating along the pseudo-field from an initial point, (θ_0, ρ_0) , on the Poincaré section $\zeta = 0$, around q toroidal periods to arrive at (θ_q, ρ_q) :

$$\begin{pmatrix} \theta_q \\ \rho_q \end{pmatrix} = P^q \begin{pmatrix} \nu \\ \rho_0 \end{pmatrix}, \quad (63)$$

where the dependence on θ_0 is suppressed.

5. Given that ν is constant along the pseudo-field but that the particular numerical value of ν is not yet known, it is required to find the particular pair (ν, ρ_0) that gives a periodic, integral curve of \mathbf{B}_ν at the prescribed angle, $\alpha \equiv \theta_0$.
6. The q -th return, pseudo tangent-map, ∇P^q , is defined by

$$\begin{pmatrix} \delta\theta_q \\ \delta\rho_q \end{pmatrix} = \nabla P^q \cdot \begin{pmatrix} \delta\nu \\ \delta\rho_0 \end{pmatrix}, \quad (64)$$

and can also be determined by pseudo fieldline integration over $\zeta \in [0, 2\pi q]$ by

$$\frac{d}{d\zeta} \nabla P^q = \begin{pmatrix} \partial_\theta \dot{\theta} & \partial_\rho \dot{\theta} \\ \partial_\theta \dot{\rho} & \partial_\rho \dot{\rho} \end{pmatrix} \cdot \nabla P^q + \begin{pmatrix} 0 & 0 \\ 1/\sqrt{g} B^\zeta & 0 \end{pmatrix},$$

with the initial condition

$$\nabla P^q = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \quad (65)$$

7. The pseudo tangent map allows an efficient Newton iterative algorithm for finding fixed points: a correction, $(\delta\nu, \delta\rho)$, to an initial guess for (ν, ρ) is determined by requiring the pseudo fieldline be periodic,

$$\begin{pmatrix} \theta_q \\ \rho_q \end{pmatrix} + \nabla P^q \cdot \begin{pmatrix} \delta\nu \\ \delta\rho_0 \end{pmatrix} = \begin{pmatrix} \theta_0 + 2\pi p \\ \rho_0 + \delta\rho \end{pmatrix}. \quad (66)$$

8. For the integrable case, for which there is a true periodic orbit for every value of the poloidal angle, the iterative solution will yield $\nu(\alpha) = 0$ for all α , and the pseudo field reduces to the true field; and similarly for the non-integrable case: where true periodic fieldlines exist, e.g. at α_X and α_O , the solution yields $\nu(\alpha_X) = 0$ and $\nu(\alpha_O) = 0$.
9. The pseudo fieldline with $\theta_0 = \alpha$ serves as an initial guess for the pseudo fieldline with $\theta_0 = \alpha + d\alpha$, and we may trace out the entire pseudo surface by varying θ_0 . Note that given that the (p, q) pseudo fieldlines have length equal to $2\pi q$, it is not required to construct the pseudo curves over the range $\alpha \in [0, 2\pi]$: periodicity means that it is only required to construct the curves over the range $\alpha \in [0, 2\pi/q]$.
10. At all angle locations where a periodic *true* fieldline does not exist, a periodic *pseudo* fieldline can still be constructed by suitably choosing ν . Intuitively, we may think of ν/\sqrt{g} as the amount of radial field that must be subtracted from the true field to ‘cancel’ the resonant effect of the perturbation, and by so doing create a rational ‘pseudo’ surface as a family of rational pseudo fieldlines. The function $\nu(\alpha)$ is sinusoidal.

11. **The user must include**

use oculus, only : coordinates, qfminsurface, qf00aa

```
type(coordinates)  :: rzmn
type(qfminsurface) :: qfms
```

in their source that calls qf00aa, where `coordinates` and `qfminsurface` are derived types (i.e. structures) that contains both the required input and output information. The variable names, `rzmn` and `qfms`, are arbitrary.

12. Required inputs

`rzmn` : structure ;

- i. the background toroidal coordinates, (ρ, θ, ζ) , that will be used;
- ii. it is assumed that before calling `qf00aa`, that the routine `bc00aa` has been called, and `rzmn` should be unchanged;

`qfms%Nfp` : integer ;

- i. the field periodicity;

`qfms%pp` : integer ;

`qfms%qq` : integer ;

- i. the rotational transform, $t \equiv pp/qq$;
- ii. the “poloidal” periodicity, `pp`, must be a multiple of the field periodicity, `Nfp`;
- iii. the integers `pp` and `qq` must be positive, and `qq` cannot be zero;
- iv. note that the sign of the rotational transform depends on the background coordinates; and if the background coordinates are such that the rotational transform is negative, then prior to calling `bc00aa` and `qf00aa` the “handedness” of the background coordinates must be changed;

`qfms%Np` : integer ;

- i. the required poloidal resolution, e.g. `Np=32`;

`qfms%Ntor` : integer ;

`qfms%Mpol` : integer ;

- i. the required Fourier resolution, e.g. `Ntor=12`, `Mpol=4`;

`qfms%odetol` : integer ;

- i. the o.d.e. integration tolerance, e.g. `odetol=10-8`,

`qfms%pqtol` : integer ;

- i. the required tolerance in locating periodic pseudo-fieldlines, e.g. `odetol=10-6`,

`qfms%rr` : real ;

`qfms%nu` : real ;

- i. initial guess for ρ and ν at $\theta_0 = 0$;
- ii. only used if `Lrestart=0`; (usually a suitable initial guess for ν is $\nu = 0$).

`qfms%Lrestart` : integer ;

- i. if `Lrestart=1`, an initial guess for both ρ_i and ν_i at each θ_i for $i = 0, Np$ is required on input;

`qfms%t(0:qNd,0:qNp)` : real ;

`qfms%r(0:qNd,0:qNp)` : real ;

`qfms%n(0:qNd,0:qNp)` : real ;

- i. only required on input if `Lrestart.eq.1`;
- ii. usually these will be provided by an earlier call to `qf00aa`;

`ifail` : integer ;

- i. if `ifail = 1`, quiet mode;
- i. if `ifail = 0`, screen output is terse;
- i. if `ifail = -1`,

4. Execution

call `qf00aa(rzmn, qfms, ifail)`

5. Outputs

qfms%qNd : integer

qNd = qq * max(Ntor,1);

qfms%qNp : integer

qNp = qq * Np ;

qfms%t(0:qNd,0:qNp) : real ;

qfms%r(0:qNd,0:qNp) : real ;

qfms%n(0:qNd,0:qNp) : real ;

qfms%mn : integer

i. mn = Ntor + 1 + Mpol * (2 * Ntor + 1)

qfms%im(1:mn) : integer

qfms%in(1:mn) : integer

i. mode identification;

qfms%Rbc(1:mn) :

qfms%Rbs(1:mn) :

qfms%Zbc(1:mn) :

qfms%Zbs(1:mn) :

i. Fourier harmonics of QFM-surface in straight pseudo fieldline angle;

qfms%ok : integer

i. error flag;

i. on output:

ifail=0 : normal execution;

ifail=1 : input error;

6. Comments:

- i. Presently, this routine uses the cylindrical field provided by `bfield`, and the coordinate transformation provided by `bc00aa`. Note that the coordinate transformation involves the (slow) Fourier summation of potentially many harmonics in the coordinate transform, and consequently `qf00aa` can be slow.
- ii. When `aa00aa` is complete, I will include an option to use the magnetic vector potential in toroidal coordinates, as this will eliminate the need for the coordinate transformation, and an exactly divergence-free numerical representation of the magnetic field will be enabled!

IX. MISCELLANEOUS/AUXILLIARY SUBROUTINES

oculus.h : last modified on 2016-11-15

