

**global**

briefly

[called by: .]

[calls: .]

**contents**

<b>1</b>	<b>global</b>	<b>1</b>
1.1	input namelists	2
1.1.1	physicslist :	2
1.1.2	numericlist :	4
1.1.3	locallist :	5
1.1.4	globallist :	6
1.1.5	diagnosticslist :	7
1.1.6	screenlist :	8
1.2	input geometry	8
1.3	internal variables	9
1.3.1	Fourier representation	9
1.3.2	iRbc, iZbs, etc. : interface geometry	9
1.3.3	Fourier Transforms	9
1.3.4	DToocc, DToocs, DToosc, DTooss : volume-integrated Chebyshev-metrics	9
1.3.5	TTsscc, TTsscs, TTsssc, TTssss : volume-integrated Chebyshev-metrics	9
1.3.6	TDstcc, TDstcs, TDstsc, TDstss : volume-integrated Chebyshev-metrics	9
1.3.7	TDszcc, TDszcs, TDszsc, TDszss : volume-integrated Chebyshev-metrics	9
1.3.8	DDttcc, DDttcs, DDttsc, DDttss : volume-integrated Chebyshev-metrics	9
1.3.9	DDtzcc, DDtzcs, DDtzsc, DDtzss : volume-integrated Chebyshev-metrics	9
1.3.10	DDzzcc, DDzzcs, DDzzsc, DDzzss : volume-integrated Chebyshev-metrics	9
1.3.11	vector potential and the Beltrami linear system	9
1.3.12	dMA, dMB, dMC, dMD, dME, dMF : field matrices	10
1.3.13	construction of ‘force’	10
1.3.14	Btemn, Bzemn, Btomn, Bzomn : covariant field on interfaces	10
1.3.15	LGdof, NGdof : geometrical degrees-of-freedom;	10
1.3.16	parallel construction of derivative matrix	10
1.3.17	dmupfdx : derivatives of multiplier and poloidal flux with respect to geometry	10
1.3.18	trigonometric factors	11
1.3.19	LBBintegral, lABintegral : volume integrals	11
1.4	subroutine readin	11
1.4.1	machprec, vsmall, small, sqrtmachprec : machine precision	11
1.4.2	input file extension $\equiv$ command line argument	11
1.4.3	reading physicslist	11
1.4.4	reading numericlist	11
1.4.5	reading locallist	11
1.4.6	reading globallist	11
1.4.7	reading diagnosticslist	11
1.4.8	reading screenlist	11
1.4.9	Mvol : total number of volumes	11
1.4.10	mn, im(1:mn) and in(1:mn) : Fourier mode identification	12
1.4.11	halfmm(1:mn, regumm(1:mn)) : regularization factor	12
1.4.12	ime and ine : extended resolution Fourier mode identification	12
1.4.13	mns, ims and ins : Fourier mode identification for straight-fieldline angle	12
1.4.14	iRbc(1:mn,0:Mvol, iZbs(1:mn,0:Mvol, iRbs(1:mn,0:Mvol and iZbc(1:mn,0:Mvol : geometry	12
1.4.15	ajk : construction of coordinate axis	12
1.5	subroutine wrtend	12

## 1.1 input namelists

1. The input file, `ext.sp`, where `ext*100` is given as command line input, contains the following namelists and interface geometry.
2. The maximum value of `Nvol` is `MNvol=256`.
3. The maximum value of `Mpol` is `MNpol= 32`.
4. The maximum value of `Ntor` is `MNtor= 16`.
5. In the following, all default settings are shown.

### 1.1.1 physicslist :

1. The namelist `physicslist` controls the geometry, profiles, and numerical resolution.

`namelist/physicslist/`

- `Igeometry = 3 : integer` : selects Cartesian, cylindrical or toroidal geometry;
  - i. `Igeometry = 1` : Cartesian; geometry determined by  $R$ ;
  - i. `Igeometry = 2` : cylindrical; geometry determined by  $R$ ;
  - i. `Igeometry = 3` : toroidal; geometry determined by  $R$  and  $Z$ ;
- `Istellsym = 1 : integer` : stellarator symmetry is enforced if `Istellsym.eq.1`;
- `Lfreebound = 0 : integer` : compute vacuum field surrounding plasma;
- `phiedge = 1.0 : real` : total enclosed toroidal magnetic flux;
- `curtor = 0.0 : real` : total enclosed (toroidal) plasma current;
- `curpol = 0.0 : real` : total enclosed (poloidal) linking current;
- `gamma = 0.0 : real` : adiabatic index; cannot set  $|\gamma| = 1$ ;
- `Nfp = 1 : integer` : field periodicity;
  - i. all Fourier representations are of the form  $\cos(m\theta - nN\zeta)$ ,  $\sin(m\theta - nN\zeta)$ , where  $N \equiv \text{Nfp}$ ;
  - i. constraint : `Nfp.ge.1`;
- `Nvol = 1 : integer` : number of volumes;
  - i. each volume  $\mathcal{V}_l$  is bounded by the  $\mathcal{I}_{l-1}$  and  $\mathcal{I}_l$  interfaces;
  - i. note that in cylindrical or toroidal geometry,  $\mathcal{I}_0$  is the degenerate coordinate axis;
  - i. constraint : `Nvol.le.MNvol`;
- `Mpol = 1 : integer` : poloidal resolution;
- `Ntor = 0 : integer` : toroidal resolution;

Internally these “double” summations are written as a “single” summation, e.g.  $f = \sum_j f_j \cos(m_j\theta - n_j\zeta)$ .
- `Lrad = 4 : integer(MNvol+1)` : Chebyshev resolution in each volume;
  - i. constraint : `Lrad(1:Mvol).ge.2`;
- `Lconstraint = -1 : integer` : selects constraints; primarily used in `ma02aa` and `mp00ac`.
  - i. if `Lconstraint.eq.-1`, then in the plasma regions  $\Delta\psi_t$ ,  $\mu$  and  $\Delta\psi_p$  are *not* varied; and in the vacuum region (only for free-boundary)  $\Delta\psi_t$  and  $\Delta\psi_p$  are *not* varied, and  $\mu = 0$ .
  - ii. if `Lconstraint.eq.0`, then in the plasma regions  $\Delta\psi_t$ ,  $\mu$  and  $\Delta\psi_p$  are *not* varied; and in the vacuum region (only for free-boundary)  $\Delta\psi_t$  and  $\Delta\psi_p$  are varied to match the prescribed plasma current, `curtor`, and the “linking” current, `curpol`, and  $\mu = 0$ ;
  - iii. if `Lconstraint.eq.1`, then in the plasma regions  $\mu$  and  $\Delta\psi_p$  are adjusted in order to satisfy the inner and outer interface transform constraints (except in the simple torus, where the enclosed poloidal flux is irrelevant, and only  $\mu$  is varied to satisfy the outer interface transform constraint); and in the vacuum region  $\Delta\psi_t$  and  $\Delta\psi_p$  are varied to match the transform constraint on the boundary and to obtain the prescribed linking current, `curpol`, and  $\mu = 0$ .
  - iv. if `Lconstraint.eq.2`, under reconstruction.
- `tflux : real(1:MNvol+1)` : toroidal flux,  $\psi_t$ , enclosed by each interface;
  - i. For each of the plasma volumes, this is a constraint: `tflux` is not varied;
  - i. For the vacuum region (only if `Lfreebound = 1`), `tflux` may be allowed to vary to match constraints;
  - i. Note that `tflux` will be normalized so that `tflux(Nvol) = 1.0`, so that `tflux` is arbitrary up to a scale factor;

- i. see also **phiedge**;
- **pflux** : real(1:MNvol+1) : poloidal flux,  $\psi_p$ , enclosed by each interface;
- **helicity** : real(1:MNvol) : helicity,  $\mathcal{K}$ , in each volume,  $\mathcal{V}_i$ ;
  - i. on exit, **helicity** is set to the computed values of  $\mathcal{K} \equiv \int \mathbf{A} \cdot \mathbf{B} dv$ ;
- **pscale = 0.0** : real : pressure scale factor;
  - i. the initial pressure profile is given by **pscale** \* **press**;
- **pressure** : real(1:MNvol+1) : pressure in each volume;
  - i. the pressure is *not* held constant, but  $p_l V_l^\gamma = P_l$  is held constant, where  $P_l$  is determined by the initial pressures and the initial volumes,  $V_l$ ;
  - i. (Note that if **gamma** = 0.0, then  $p_l \equiv P_l$ .)
  - i. on output, the pressure is given by  $p_l = P_l / V_l^\gamma$ , where  $V_l$  is the final volume;
  - i. **pressure** is only used in calculation of interface force-balance;
- **Ladiabatic = 0** : integer : logical flag;
  - i. if **Ladiabatic** = 0, the adiabatic constants are determined by the initial pressure and volume;
  - i. if **Ladiabatic** = 1, the adiabatic constants are determined by the given input **adiabatic**;
- **adiabatic** : real(1:MNvol+1) : adiabatic constants in each volume;
  - i. the pressure is *not* held constant, but  $p_l V_l^\gamma = P_l \equiv \text{adiabatic}$  is constant,
  - i. note that if **gamma** = 0.0, then **pressure** = **adiabatic**;
  - i. **pressure** is only used in calculation of interface force-balance;
- **mu** : real(1:MNvol+1) : helicity-multiplier,  $\mu$ , in each volume;
- **pl = 0** : integer(0:MNvol) :
- **ql = 0** : integer(0:MNvol) :
- **pr = 0** : integer(0:MNvol) :
- **qr = 0** : integer(0:MNvol) :
  - i. “inside” interface rotational-transform is  $\epsilon = (p_l + \gamma p_r) / (q_l + \gamma q_r)$ , where  $\gamma$  is the golden mean,  $\gamma = (1 + \sqrt{5})/2$ ;
  - i. if both  $q_l = 0$  and  $q_r = 0$ , then the (inside) interface rotational-transform is defined by **iota**;
- **iota** : real(0:MNvol) : rotational-transform,  $\epsilon$ , on inner side of each interface;
  - i. only relevant if illogical input for **ql** and **qr** are provided;
- **lp = 0** : integer(0:MNvol) :
- **lq = 0** : integer(0:MNvol) :
- **rp = 0** : integer(0:MNvol) :
- **rq = 0** : integer(0:MNvol) :
  - “outer” interface rotational-transform is  $\epsilon = (p_l + \gamma p_r) / (q_l + \gamma q_r)$ , where  $\gamma$  is the golden mean,  $\gamma = (1 + \sqrt{5})/2$ ;
  - if both  $q_l = 0$  and  $q_r = 0$ , then the (outer) interface rotational-transform is defined by **oita**;
- **oita** : real(0:MNvol) : rotational-transform,  $\epsilon$ , on outer side of each interface;
  - only relevant if illogical input for **ql** and **qr** are provided;
- **mupftol = 1.0e-16** : real : accuracy to which  $\mu$  and  $\Delta\psi_p$  are required;
  - only relevant if constraints on transform, enclosed currents etc. are to be satisfied iteratively, see **Lconstraint**;
- **mupfits = 8** : integer : an upper limit on the transform/helicity constraint iterations;
- only relevant if constraints on transform, enclosed currents etc. are to be satisfied iteratively, see **Lconstraint**;
  - constraint: **mupfits** > 0;
- **Rac** : real( 0:MNtor ) : Fourier harmonics of axis ; stellarator symmetric;
- **Zas** : real( 0:MNtor ) : Fourier harmonics of axis ; stellarator symmetric;
- **Ras** : real( 0:MNtor ) : Fourier harmonics of axis ; non-stellarator symmetric;
- **Zac** : real( 0:MNtor ) : Fourier harmonics of axis ; non-stellarator symmetric;
- **Rbc** : real(-MNtor:MNtor, -MMpol:MMpol) : Fourier harmonics of boundary; stellarator symmetric;

- **Zbs** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of boundary; stellarator symmetric;
- **Rbs** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of boundary; non-stellarator symmetric;
- **Zbc** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of boundary; non-stellarator symmetric;
- **Rwc** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of wall ; stellarator symmetric;
- **Zws** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of wall ; stellarator symmetric;
- **Rws** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of wall ; non-stellarator symmetric;
- **Zwc** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of wall ; non-stellarator symmetric;
- **Vns** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of vacuum normal field at boundary;
- **Bns** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of plasma normal field at boundary;
- **Vnc** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of vacuum normal field at boundary;
- **Bnc** : `real(-MNtor:MNtor,-MMpol:MMpol)` : Fourier harmonics of plasma normal field at boundary;

### 1.1.2 numericlist :

1. The namelist **numericlist** controls internal resolution parameters that the user rarely needs to consider.

`namelist/numericlist/`

- **Linitialize = 0 : integer** : to initialize geometry using a regularization / extrapolation method;
  - if **Linitialize = -I**, where  $I$  is a positive integer, the geometry of the  $i = 1, N_V - I$  surfaces constructed by an extrapolation;
  - if **Linitialize = 0**, the geometry of the interior surfaces is provided after the namelists in the input file;
  - if **Linitialize = 1**, the interior surfaces will be initialized as  $R_{l,m,n} = R_{N,m,n} \psi_{t,l}^{m/2}$ , where  $R_{N,m,n}$  is the plasma boundary and  $\psi_{t,l}$  is the given toroidal flux enclosed by the  $l$ -th interface, normalized to the total enclosed toroidal flux; a similar extrapolation is used for  $Z_{l,m,n}$ ;
  - note that the Fourier harmonics of the boundary is *always* given by the **Rbc** and **Zbs** given in **physicslist**;
  - if **Linitialize = 2**, the interior surfaces *and the plasma boundary* will be initialized as  $R_{l,m,n} = R_{W,m,n} \psi_{t,l}^{m/2}$ , where  $R_{W,m,n}$  is the computational boundary and  $\psi_{t,l}$  is the given toroidal flux enclosed by the  $l$ -th interface, normalized to the total enclosed toroidal flux; a similar extrapolation is used for  $Z_{l,m,n}$ ;
  - note that, for free-boundary calculations, the Fourier harmonics of the computational boundary is *always* given by the **Rwc** and **Zws** given in **physicslist**;
  - if **Linitialize = 1, 2**, it is not required to provide the geometry of the interfaces after the namelists;
- **Lzerovac = 0 : integer** : to adjust vacuum field to cancel plasma field on computational boundary;
  - only relevant if **Lfreebound = 1**,
- **Ndiscrete = 2 : integer** :
  - resolution of the real space grid on which fast Fourier transforms are performed is given by **Ndiscrete\*MMpol\*4**;
  - constraint **Ndiscrete > 0**;
- **Nquad = -1 : integer** : the resolution of the Gaussian quadrature;
  - the resolution of the Gaussian quadrature,  $\int f(s)ds = \sum_k \omega_k f(s_k)$ , in each volume is given by **Iquad<sub>v</sub>**,
  - **Iquad<sub>v</sub>** is set in **preset**.
- **iMpol = -4 : integer** : Fourier resolution of straight-fieldline angle on interfaces;
  - the rotational-transform on the interfaces is determined by a transformation to the straight-fieldline angle, with poloidal resolution given by **iMpol**;
  - if **iMpol.le.0**, then **iMpol = Mpol - iMpol**;
- **iNtor = -4 : integer** : Fourier resolution of straight-fieldline angle on interfaces;
  - the rotational-transform on the interfaces is determined by a transformation to the straight-fieldline angle, with toroidal resolution given by **iNtor**;
  - if **iNtor.le.0**, then **iNtor = Ntor - iNtor**;
  - if **Ntor.eq.0**, then the toroidal resolution of the angle transformation is set **iNtor = 0**.
- **Lsparse = 0 : integer** : controls method used to solve for rotational-transform on interfaces;

- if **Lsparse = 0**, the transformation to the straight-fieldline angle is computed in Fourier space using a dense matrix solver, **NAG: F04AAF**;
- if **Lsparse = 1**, the transformation to the straight-fieldline angle is computed in real space using a dense matrix solver, **NAG: F04ATF**;
- if **Lsparse = 2**, the transformation to the straight-fieldline angle is computed in real space using a sparse matrix solver, **NAG: F11DEF**;
- if **Lsparse = 3**, the different methods for constructing the straight-fieldline angle are compared;
- **Lsvdiota = 0 : integer** : controls method used to solve for rotational-transform on interfaces; only relevant if **Lsparse = 0**;
  - if **Lsvdiota = 0**, use standard linear solver to construct straight fieldline angle transformation;
  - if **Lsvdiota = 1**, use SVD method to compute rotational-transform;
- **Imethod = 3 : integer** : controls iterative solution to sparse matrix arising in real-space transformation to the straight-fieldline angle; only relevant if **Lsparse.eq.2**; see **tr00ab** for details;
  - if **imethod = 1**, the method is RGMRES;
  - if **imethod = 2**, the method is CGS;
  - if **imethod = 3**, the method is BICGSTAB;
- **iorder = 2 : integer** : controls real-space grid resolution for constructing the straight-fieldline angle; only relevant if **Lsparse>0**; determines order of finite-difference approximation to the derivatives;
  - if **iorder = 2**,
  - if **iorder = 4**,
  - if **iorder = 6**,
- **Iprecon = 0 : integer** : controls iterative solution to sparse matrix arising in real-space transformation to the straight-fieldline angle; only relevant if **Lsparse.eq.2**; see **tr00ab** for details;
  - if **iprecon = 0**, the preconditioner is ‘N’;
  - if **iprecon = 1**, the preconditioner is ‘J’;
  - if **iprecon = 2**, the preconditioner is ‘S’;
- **iotatol = -1.0 : real** : tolerance required for iterative construction of straight-fieldline angle; only relevant if **Lsparse.ge.2**
- **Lextrap = 0 : integer** : geometry of innermost interface is defined by extrapolation;
- **Mregular = -1 : integer** : maximum regularization factor;
  - if **Mregular.ge.2**, then  $\text{regumm}_i = \text{Mregular} / 2$  where  $m_i > \text{Mregular}$

### 1.1.3 locallist :

1. The namelist **locallist** controls the construction of the Beltrami fields in each volume.

namelist/locallist/

- **LBeltrami = 4 integer**
  - if **LBeltrami = 1,3,5 or 7**, (SQP) then the Beltrami field in each volume is constructed by minimizing the magnetic energy with the constraint of fixed helicity; this is achieved by using sequential quadratic programming as provided by **NAG: E04UFF**; this approach has the benefit (in theory) of robustly constructing minimum energy solutions when multiple, i.e. bifurcated, solutions exist.
  - if **LBeltrami = 2,3,6 or 7**, (Newton) then the Beltrami fields are constructed by employing a standard Newton method for locating an extremum of  $F \equiv \int B^2 dv - \mu(\int \mathbf{A} \cdot \mathbf{B} dv - \mathcal{K})$ , where  $\mu$  is treated as an independent degree of freedom similar to the parameters describing the vector potential and  $\mathcal{K}$  is the required value of the helicity; this is the standard Lagrange multiplier approach for locating the constrained minimum; this method cannot distinguish saddle-type extrema from minima, and which solution that will be obtained depends on the initial guess;
  - if **LBeltrami = 4,5,6 or 7**, (linear) it is assumed that the Beltrami fields are parameterized by  $\mu$ ; in this case, it is only required to solve  $\nabla \times \mathbf{B} = \mu \mathbf{B}$  which reduces to a system of linear equations;  $\mu$  may or may not be adjusted iteratively, depending on **Lconstraint**, to satisfy either rotational-transform or helicity constraints;
  - for flexibility and comparison, each of the above methods can be employed; for example:
    - \* if **LBeltrami = 1**, only the SQP method will be employed;
    - \* if **LBeltrami = 2**, only the Newton method will be employed;
    - \* if **LBeltrami = 4**, only the linear method will be employed;
    - \* if **LBeltrami = 3**, the SQP and the Newton method are used;

- \* if **LBeltrami** = 5, the SQP and the linear method are used;
- \* if **LBeltrami** = 6, the Newton and the linear method are used;
- \* if **LBeltrami** = 7, all three methods will be employed;
- **Linitgues** = 1 integer controls how initial guess for Beltrami field is constructed;
  - only relevant for routines that require an initial guess for the Beltrami fields, such as the SQP and Newton methods, or the sparse linear solver;
  - if **Linitgues** = 0, the initial guess for the Beltrami field is trivial;
  - if **Linitgues** = 1, the initial guess for the Beltrami field is an integrable approximation;
  - if **Linitgues** = 2, the initial guess for the Beltrami field is read from file;
- **Lposdef** = 0 : integer : indicates whether the Beltrami/Vacuum linear system is positive definite;
  - if **Lposdef** = 0, the Beltrami/Vacuum linear system is assumed not positive definite;
  - if **Lposdef** = 1, the Beltrami/Vacuum linear system is assumed to be positive definite;
  - only relevant for the linear method of constructing the Beltrami fields;
  - note that with “new” method for implementing the flux constraints that the Beltrami matrix is never expected to be positive-definite; and so **Lposdef** should be redundant;

## 2. Comments:

- (a) The transformation to straight-fieldline coordinates is singular when the rotational-transform of the interfaces is rational; however, the rotational-transform is still well defined.

---

### 1.1.4 globallist :

1. The namelist **globallist** controls the search for global force-balance:

namelist/globallist/

- **Lfindzero** = 0 : integer : use Newton methods to find zero of force-balance, which is computed by **dforce**;
  - i. if **Lfindzero** = 0, then **dforce** is called once to compute the Beltrami fields consistent with the given geometry and constraints;
  - ii. if **Lfindzero** = 1, then call **NAG: C05NDF** (uses function values only), which iteratively calls **dforce**;
  - iii. if **Lfindzero** = 2, then call **NAG: C05PDF** (uses derivative information), which iteratively calls **dforce**;
- **escale** = 0.0 : real : controls the weight factor, **BBweight**, in the force-imbalance harmonics;
  - i.  $\text{BBweight}(i) \equiv \text{epsilon} \times \exp[-\text{escale} \times (m_i^2 + n_i^2)]$
  - ii. defined in **preset**; used in **dforce**;
  - iii. also see Eqn.(2) below;
- **epsilon** = 1.0 : real : weighting of force-imbalance;
  - i. used in **dforce**; also see Eqn.(2) below;
- **pcondense** = 2.0 : real : spectral condensation parameter;
  - i. used in **preset** to define  $\text{mmpp}(i) \equiv m_i^p$ , where  $p \equiv \text{pcondense}$ ;
  - ii. the angle freedom is exploited to minimize  $\text{epsilon} \sum_i m_i^p (R_i^2 + Z_i^2)$  with respect to tangential variations in the interface geometry;
  - iii. also see Eqn.(3) below;
- **epsilon** = 0.0 : real : weighting of spectral-width constraint;
  - i. used in **dforce**; also see Eqn.(3) below;
- **wpoloidal** = 1.0 : real : “star-like” poloidal angle constraint radial exponential factor; used in **preset** to construct **sweight**
- **upsilon** = 1.0 : real : weighting of “star-like” poloidal angle constraint; used in **preset** to construct **sweight**;
- **forcetol** = 1.0e-10 : real : required tolerance in force-balance error; only used as an initial check;
  - i. if the initially supplied interfaces are consistent with force-balance to within **forcetol**, then the geometry of the interfaces is not altered;
  - ii. if not, then the geometry of the interfaces is changed in order to bring the configuration into forcebalance so that the geometry of interfaces is within **c05xtol**, defined below, of the true solution;
  - iii. to force execution of either **NAG: C05NDF** or **NAG: C05PDF**, regardless of the initial force imbalance, set **forcetol** < 0;

- **c05xmax** = 1.0e-06 : real : required tolerance in position,  $\mathbf{x} \equiv \{R_{i,v}, Z_{i,v}\}$ ;
- **c05xtol** = 1.0e-12 : real : required tolerance in position,  $\mathbf{x} \equiv \{R_{i,v}, Z_{i,v}\}$ ;
  - used by both **NAG: C05NDF** and **NAG: C05PDF**; see the NAG documents for further details on how the error is defined;
  - constraint **c05xtol.gt.0.0**;
- **c05factor** = 1.0e-02 : real : used to control initial step size in **NAG: C05NDF** and **NAG: C05PDF**;
  - constraint **c05factor.gt.0.0**;
  - only relevant if **Lfindzero.gt.0**;
- **LreadGF** = T : logical : read  $\nabla_{\mathbf{x}}\mathbf{F}$  from file .GF;
  - only used if **Lfindzero** = 2;
  - only used in **newton**;
- **mfreeits** = 0 : integer : maximum allowed free-boundary iterations;
  - only used if **Lfreebound** = 1;
  - only used in **xspech**;
- **bnstol** = 1.0e-06 : redundant;
- **bnsblend** = 0.666 : redundant;
- **gBntol** = 1.0e-06 : real : required tolerance in free-boundary iterations;
  - only used if **Lfreebound** = 1;
  - only used in **xspech**;
- **gBnbld** = 0.666 : real : normal blend;
  - The “new” magnetic field at the computational boundary produced by the plasma currents is updated using a Picard scheme:
 
$$(\mathbf{B} \cdot \mathbf{n})^{j+1} = \mathbf{gBnbld} \times (\mathbf{B} \cdot \mathbf{n})^j + (1 - \mathbf{gBnbld}) \times (\mathbf{B} \cdot \mathbf{n})^*, \quad (1)$$
 where  $j$  labels free-boundary iterations, and  $(\mathbf{B} \cdot \mathbf{n})^*$  is computed by virtual casing.
  - only used if **Lfreebound** = 1;
  - only used in **xspech**;
- **vcasingeps** = 1.0e-12 : real : regularization of Biot-Savart; see **bnorml**, **casing**;
- **vcasingtol** = 1.0e-08 : real : accuracy on virtual casing integral; see **bnorml**, **casing**;
- **vcasingits** = 8 : integer : minimum number of calls to adaptive virtual casing routine; see **casing**;
- **vcasingper** = 1 : integer : periods of integration in adaptive virtual casing routine; see **casing**;
- **mcasingcal** = 8 : integer : minimum number of calls to adaptive virtual casing routine; see **casing**;

## 2. Comments:

- (a) The “force” vector,  $\mathbf{F}$ , which is constructed in **dforce**, is a combination of pressure-imbalance Fourier harmonics,

$$F_{i,v} \equiv [[p + B^2/2]]_{i,v} \times \exp[-\mathbf{escale}(m_i^2 + n_i^2)] \times \mathbf{opsilon}, \quad (2)$$

and spectral-condensation constraints,  $I_{i,v}$ , and the “star-like” angle constraints,  $S_{i,v}$ , (see **lforce** for details)

$$F_{i,v} \equiv \mathbf{epsilon} \times I_{i,v} + \mathbf{upsilon} \times (\psi_v^\omega S_{i,v,1} - \psi_{v+1}^\omega S_{i,v+1,0}), \quad (3)$$

where  $\psi_v \equiv$  normalized toroidal flux, **tflux**, and  $\omega \equiv$  **wpoloidal**.

---

### 1.1.5 diagnosticslist :

- The namelist **diagnosticslist** controls post-processor diagnostics, such as Poincaré plot resolution, ..., **namelist/diagnosticslist/**
  - **odetol** = 1.0e-07 : real : o.d.e. integration tolerance for all field line tracing routines;
  - **absreq** = 1.0e-08 : real : redundant;
  - **relreq** = 1.0e-08 : real : redundant;
  - **absacc** = 1.0e-04 : real : redundant;
  - **epsr** = 1.0e-06 : real : redundant;



- **nPpts = 0** : integer : number of toroidal transits used (per trajectory) in following field lines for constructing Poincaré plots; if **nPpts < 1**, no Poincaré plot is constructed;
- **nPtrj = -1** : integer(1:MNvol+1) : number of trajectories in each annulus to be followed in constructing Poincaré plot;
  - if **nPtrj(1) < 0**, then **nPtrj(1) = Ni(1)**, where **Ni(1)** is the grid resolution used to construct the Beltrami field in volume *l*;
- **LHevalues = F** : logical : to compute eigenvalues of  $\nabla \mathbf{F}$ ;
- **LHevectors = F** : logical : to compute eigenvectors (and also eigenvalues) of  $\nabla \mathbf{F}$ ;
- **Lperturbed = 0** : integer : to compute linear, perturbed equilibrium;
- **dpp = 1** : integer : perturbed harmonic;
- **dqq = 1** : integer : perturbed harmonic;
- **Lcheck = 0** : integer : implement various checks;
  - if **Lcheck = 0**, no additional check on the calculation is performed;
  - if **Lcheck = 1**, the error in the current, i.e.  $\nabla \times \mathbf{B} - \mu \mathbf{B}$  is computed as a post-diagnostic;
  - if **Lcheck = 2**, the analytic derivatives of the interface transform w.r.t. the helicity multiplier,  $\mu$ , and the enclosed poloidal flux,  $\Delta\psi_p$ , are compared to a finite-difference estimate;
    - i. only if **Lconstraint.eq.1**;
    - ii. only for **dspec** executable, i.e. must compile with **DFLAGS = "-D DEBUG"**;
  - if **Lcheck = 3**, the analytic derivatives of the volume w.r.t. interface Fourier harmonic is compared to a finite-difference estimate;
    - i. must set **Lfindzero=2**,
    - ii. set **forcetol** sufficiently small and set **LreadGF = F**, so that the matrix of second derivatives is calculated,
    - iii. only for **dspec** executable, i.e. must compile with **DFLAGS = "-D DEBUG"**;
  - if **Lcheck = 4**, the analytic calculation of the derivatives of the magnetic field,  $B^2$ , at the interfaces is compared to a finite-difference estimate;
    - i. must set **Lfindzero=2**,
    - ii. set **forcetol** sufficiently small,
    - iii. set **LreadGF=F**,
    - iv. only for **dspec** executable, i.e. must compile with **DFLAGS = "-D DEBUG"**;
  - if **Lcheck = 5**, the analytic calculation of the matrix of the derivatives of the force imbalance is compared to a finite-difference estimate;
  - if **Lcheck = 6**, the virtual casing calculation is compared to **xdiagno**;
    - i. the input file for **xdiagno** is written by **bnorml**;
    - ii. this provides the Cartesian coordinates on the computational boundary where the virtual casing routine **casing** computes the magnetic field, with the values of the magnetic field being written to the screen for comparison;
    - iii. must set **Freebound=1, Lfindzero.gt.0, mfreeits.ne.0**;
    - iii. **xdiagno**; must be executed manually;
- **Ltiming = T** : logical : to check timing;
- **fudge = 1.0** : real : redundant;
- **scaling = 1.0** : real : redundant;

#### 1.1.6 screenlist :

1. The namelist **screenlist** controls screen output.

namelist/screenlist/

- Every subroutine, e.g. **xy00aa.h**, has its own write flag, **Wxy00aa**.

## 1.2 input geometry

1. The geometry of the *l*-th interface, for  $l = 0, N$  where  $N \equiv \mathbf{Nvol}$ , is described by a set of Fourier harmonics, using an arbitrary poloidal angle,

$$R_l(\theta, \zeta) = \sum_j R_{j,l} \cos(m_j \theta - n_j \zeta), \quad (4)$$

$$Z_l(\theta, \zeta) = \sum_j Z_{j,l} \sin(m_j \theta - n_j \zeta). \quad (5)$$



- These harmonics are read from the `ext.sp` file and come directly after the namelists described above. The required format is as follows:

$$\begin{array}{cccccccc}
m_1 & n_1 & R_{1,0} & Z_{1,0} & R_{1,1} & Z_{1,1} & \dots & R_{1,N} & Z_{1,N} \\
m_2 & n_2 & R_{2,0} & Z_{2,0} & R_{2,1} & Z_{2,1} & \dots & R_{2,N} & Z_{2,N} \\
\dots & & & & & & & & \\
m_j & n_j & R_{j,0} & Z_{j,0} & R_{j,1} & Z_{j,1} & \dots & R_{j,N} & Z_{j,N} \\
\dots & & & & & & & & 
\end{array} \tag{6}$$

- The coordinate axis corresponds to  $j = 0$  and the outermost boundary corresponds to  $j = \text{Nvol}$ .
- An arbitrary selection of harmonics may be included in any order, but only those within the range specified by `Mpol` and `Ntor` will be used.
- The geometry of *all* the interfaces, i.e.  $l = 0, N$ , including the degenerate ‘coordinate-axis’ interface, must be given.

### 1.3 internal variables

#### 1.3.1 Fourier representation

- Enhanced resolution is required for the metric elements,  $g_{ij}/\sqrt{g}$ , which is given by `mne`, `ime.`, and `ine`. The Fourier resolution here is determined by `lMpol=2*Mpol` and `lNtor=2*Ntor`.
- Enhanced resolution is required for the transformation to straight-field line angle on the interfaces, which is given by `mns`, `ims.`, and `ins`. The Fourier resolution here is determined by `iMpol` and `iNtor`.

#### 1.3.2 iRbc, iZbs, etc. : interface geometry

- The Fourier harmonics of the interfaces are contained in `iRbc(1:mn,0:Mvol)` and `iZbs(1:mn,0:Mvol)`, where `iRbc(1,j)`, `iZbs(1,j)` contains the Fourier harmonics,  $R_j, Z_j$ , of the  $l$ -th interface.

#### 1.3.3 Fourier Transforms

- The coordinate geometry and fields are mapped to/from Fourier space and real space using `NAG: C06FUF`.
- The resolution of the real space grid is given by `Nt=Ndiscrete*4*Mpol` and `Nz=Ndiscrete*4*Ntor`.
- Trigonometric information required for the fast Fourier transform’s is saved in `trigm(1:2*Nt)`, `trign(1:2*Nz)`, and `trigwk(1:2*Ntz)` where `Ntz=Nt*Nz`.
- Various workspace arrays are allocated. `sg(0:3,Ntz)`, which contains the Jacobian and its derivatives; and `guv(0:6,0:3,1:Ntz)`, which contains the metric elements and their derivatives.

#### 1.3.4 DToocc, DToocs, DToosc, DTooss : volume-integrated Chebyshev-metrics

#### 1.3.5 TTsscc, TTsscs, TTsssc, TTssss : volume-integrated Chebyshev-metrics

#### 1.3.6 TDstcc, TDstcs, TDstsc, TDstss : volume-integrated Chebyshev-metrics

#### 1.3.7 TDSzcc, TDSzcs, TDSzsc, TDSzss : volume-integrated Chebyshev-metrics

#### 1.3.8 DDttcc, DDttcs, DDttsc, DDttss : volume-integrated Chebyshev-metrics

#### 1.3.9 DDtzcc, DDtzcs, DDtzsc, DDtzss : volume-integrated Chebyshev-metrics

#### 1.3.10 DDzzcc, DDzzcs, DDzzsc, DDzzss : volume-integrated Chebyshev-metrics

- These are allocated in `dforce`, defined in `ma00aa`, and are used in `matrix` to construct the matrices.

#### 1.3.11 vector potential and the Beltrami linear system

- In each volume, the total degrees of freedom in the Beltrami linear system is `NAdof(1:Nvol)`. This depends on `Mpol`, `Ntor` and `Lrad(vvol)`.
- The covariant components of the vector potential are written as

$$A_\theta = \sum_i \sum_{l=0}^L A_{\theta,e,i,l} T_l(s) \cos \alpha_i + \sum_i \sum_{l=0}^L A_{\theta,o,i,l} T_l(s) \sin \alpha_i \tag{7}$$

$$A_\zeta = \sum_i \sum_{l=0}^L A_{\zeta,e,i,l} T_l(s) \cos \alpha_i + \sum_i \sum_{l=0}^L A_{\zeta,o,i,l} T_l(s) \sin \alpha_i, \tag{8}$$

where  $T_l(s)$  are the Chebyshev polynomials and  $\alpha_i \equiv m_i \theta - n_i \zeta$ .

3. The following internal arrays are declared in [preset](#)

$$\text{dAte}(0,i)\%s(1) \equiv A_{\theta,e,i,l}$$

$$\text{dAze}(0,i)\%s(1) \equiv A_{\zeta,e,i,l}$$

$$\text{dAto}(0,i)\%s(1) \equiv A_{\theta,o,i,l}$$

$$\text{dAzo}(0,i)\%s(1) \equiv A_{\zeta,o,i,l}$$

### 1.3.12 dma, dmb, dmc, dmd, dme, dmf : field matrices

1. The energy,  $W \equiv \int dv \mathbf{B} \cdot \mathbf{B}$ , and helicity,  $K \equiv \int dv \mathbf{A} \cdot \mathbf{B}$ , functionals may be written

$$W = \frac{1}{2} a_i A_{i,j} a_j + a_i B_{i,j} \psi_j + \frac{1}{2} \psi_i C_{i,j} \psi_j \quad (9)$$

$$K = \frac{1}{2} a_i D_{i,j} a_j + a_i E_{i,j} \psi_j + \frac{1}{2} \psi_i F_{i,j} \psi_j \quad (10)$$

where  $\mathbf{a} \equiv \{A_{\theta,e,i,l}, A_{\zeta,e,i,l}, A_{\theta,o,i,l}, A_{\zeta,o,i,l}, f_{e,i}, f_{o,i}\}$  contains the independent degrees of freedom and  $\boldsymbol{\psi} \equiv \{\Delta\psi_t, \Delta\psi_p\}$ .

2. These are allocated and deallocated in [dforce](#), assigned in [matrix](#), and used in [mp00ac](#) and [? df00aa](#).

### 1.3.13 construction of ‘force’

1. The force vector is comprised of `Bomn` and `Iomn`.

### 1.3.14 Btemn, Bzemn, Btomn, Bzomn : covariant field on interfaces

1. The covariant field:

### 1.3.15 LGdof, NGdof : geometrical degrees-of-freedom;

1. The geometrical degrees-of-freedom:

### 1.3.16 parallel construction of derivative matrix

1. The derivatives of force-balance,  $[[p + B^2/2]]$ , and the spectral constraints (see [sw03aa](#)), with respect to the interface geometry is constructed in parallel by [dforce](#).
2. force-balance across the  $l$ -th interface depends on the fields in the adjacent interfaces.

### 1.3.17 dmupfdx : derivatives of multiplier and poloidal flux with respect to geometry

1. The information in `dmupfdx` describes how the helicity multiplier,  $\mu$ , and the enclosed poloidal flux,  $\Delta\psi_p$ , must vary as the geometry is varied in order to satisfy the interface transform constraint.
2. The internal variable `dmupfdx(1:Mvol,1:2,1:LGdof,0:1)` is allocated/deallocated in [newton](#), and [hesian](#) if selected.
3. The magnetic field depends on the Fourier harmonics of both the inner and outer interface geometry (represented here as  $x_j$ ), the helicity multiplier, and the enclosed poloidal flux, i.e.  $\mathbf{B}_{\pm} = \mathbf{B}_{\pm}(x_j, \mu, \Delta\psi_p)$ , so that

$$\delta\mathbf{B}_{\pm} = \frac{\partial\mathbf{B}_{\pm}}{\partial x_j} \delta x_j + \frac{\partial\mathbf{B}_{\pm}}{\partial \mu} \delta \mu + \frac{\partial\mathbf{B}_{\pm}}{\partial \Delta\psi_p} \delta \Delta\psi_p. \quad (11)$$

4. This information is used to adjust the calculation of how force-balance, i.e.  $B^2$  at the interfaces, varies with geometry at fixed interface rotational transform. Given

$$B_{\pm}^2 = B_{\pm}^2(x_j, \mu, \Delta\psi_p), \quad (12)$$

we may derive

$$\frac{\partial B_{\pm}^2}{\partial x_j} = \frac{\partial B_{\pm}^2}{\partial x_j} + \frac{\partial B_{\pm}^2}{\partial \mu} \frac{\partial \mu}{\partial x_j} + \frac{\partial B_{\pm}^2}{\partial \Delta\psi_p} \frac{\partial \Delta\psi_p}{\partial x_j} \quad (13)$$

5. The constraint to be enforced is that  $\mu$  and  $\Delta\psi_p$  must generally vary as the geometry is varied if the value of the rotational-transform constraint on the inner/outer interface is to be preserved, i.e.

$$\left( \begin{array}{cc} \frac{\partial t_-}{\partial \mathbf{B}_-} \cdot \frac{\partial \mathbf{B}_-}{\partial \mu} & , \quad \frac{\partial t_-}{\partial \mathbf{B}_-} \cdot \frac{\partial \mathbf{B}_-}{\partial \Delta\psi_p} \\ \frac{\partial t_+}{\partial \mathbf{B}_+} \cdot \frac{\partial \mathbf{B}_+}{\partial \mu} & , \quad \frac{\partial t_+}{\partial \mathbf{B}_+} \cdot \frac{\partial \mathbf{B}_+}{\partial \Delta\psi_p} \end{array} \right) \left( \begin{array}{c} \frac{\partial \mu}{\partial x_j} \\ \frac{\partial \Delta\psi_p}{\partial x_j} \end{array} \right) = - \left( \begin{array}{cc} \frac{\partial t_-}{\partial \mathbf{B}_-} \cdot \frac{\partial x_j}{\partial \mathbf{B}_-} & \frac{\partial \mathbf{B}_-}{\partial x_j} \\ \frac{\partial t_+}{\partial \mathbf{B}_+} \cdot \frac{\partial x_j}{\partial \mathbf{B}_+} & \frac{\partial \mathbf{B}_+}{\partial x_j} \end{array} \right). \quad (14)$$

6. This  $2 \times 2$  linear equation is solved in [dforce](#); and the derivatives of the rotational-transform are given in `diotadxup`, see [preset](#).
7. A finite-difference estimate is computed if [Lcheck.eq.4](#).

### 1.3.18 trigonometric factors

1. To facilitate construction of the metric integrals, various trigonometric identities are exploited.
2. The required information is saved in
3. The following are used for volume integrals (see [volume](#))

$$a_{i,j,k} = 4 m_k \oint \oint d\theta d\zeta \cos(\alpha_i) \cos(\alpha_j) \cos(\alpha_k) / (2\pi)^2, \quad (15)$$

$$b_{i,j,k} = 4 m_j \oint \oint d\theta d\zeta \cos(\alpha_i) \sin(\alpha_j) \sin(\alpha_k) / (2\pi)^2, \quad (16)$$

### 1.3.19 lBBintegral, lABintegral : volume integrals

1. The energy functional,  $F \equiv \sum_l F_l$ , where

$$F_l \equiv \left( \int_{V_l} \frac{p_l}{\gamma - 1} + \frac{B_l^2}{2} dv \right) = \frac{P_l}{\gamma - 1} V_l^{1-\gamma} + \int_{V_l} \frac{B_l^2}{2} dv, \quad (17)$$

where the second expression is derived using  $p_l V_l^\gamma = P_l$ , where  $P_l$  is the adiabatic-constant. In Eqn.(17), it is implicit that  $\mathbf{B}$  satisfies (i) the toroidal and poloidal flux constraints; (ii) the interface constraint,  $\mathbf{B} \cdot \nabla s = 0$ ; and (iii) the helicity constraint (or the transform constraint)

2. The derivatives of  $F_l$  with respect to the inner and outer adjacent interface geometry are stored in

`dFF(1:Nvol,0:1,0:mn+mn-1)`, where

$F_l \equiv \text{dFF}(1,0, 0)$

$\partial F_l / \partial R_{l-1,j} \equiv \text{dFF}(11,0, j)$

$\partial F_l / \partial Z_{l-1,j} \equiv \text{dFF}(11,0,mn\}j)$

$\partial F_l / \partial R_{l,j} \equiv \text{dFF}(11,1, j)$

$\partial F_l / \partial Z_{l,j} \equiv \text{dFF}(11,1,mn\}j)$

3. The volume integrals  $\int dv$ ,  $\int B^2 dv$  and  $\int \mathbf{A} \cdot \mathbf{B} dv$  in each volume are computed and saved in `volume(0:2,1:Nvol)`.

## 1.4 subroutine readin

1. The master node reads the input namelist and sets various internal variables. The relevant quantities are then broadcast.

### 1.4.1 machprec, vsmall, small, sqrtmachprec : machine precision

1. The machine precision is determined using [NAG: X02AJF](#).

### 1.4.2 input file extension $\equiv$ command line argument

1. The input file name, `ext`, is given as the first command line input, and the input file itself is `ext.sp`
2. Additional command line inputs recognized are:

(a) `-help`, `-h` ; will give help information to user; under construction;

(b) `-readin` ; will immediately set `Wreadin=T`; this may be over-ruled when `namelist/screenlist/` is read;

### 1.4.3 reading physicslist

1. The internal variable, `Mvol = Nvol + Lfreebound`, gives the number of computational domains.
2. The input value for the fluxes enclosed within each interface, `tflux(1:Mvol)` and `tflux(1:Mvol)`, are immediately normalized:

`tflux(1:Mvol)  $\rightarrow$  tflux(1:Mvol)/tflux(Nvol).`

`pflux(1:Mvol)  $\rightarrow$  pflux(1:Mvol)/tflux(Nvol).`

(The input  $\Phi_{edge} \equiv \text{phiedge}$  will provide the total toroidal flux; see [preset](#).)

### 1.4.4 reading numericlist

### 1.4.5 reading locallist

### 1.4.6 reading globallist

### 1.4.7 reading diagnosticslist

### 1.4.8 reading screenlist

### 1.4.9 Mvol : total number of volumes

1. The number of plasma volumes is `Mvol=Nvol+Lfreebound`;

#### 1.4.10 mn, im(1:mn) and in(1:mn) : Fourier mode identification

1. The Fourier description of even periodic functions is

$$f(\theta, \zeta) = \sum_{n=0}^N f_{0,n} \cos(-n\zeta) + \sum_{m=1}^M \sum_{n=-N}^N f_{m,n} \cos(m\theta - n\zeta), \quad (18)$$

where the resolution is given on input,  $M \equiv \text{Mpol}$  and  $N \equiv \text{Ntor}$ .

2. For convenience, the Fourier summations are written as

$$f(s, \theta, \zeta) = \sum_j f_j(s) \cos(m_j \theta - n_j \zeta), \quad (19)$$

for  $j = 1, \text{mn}$ , where  $\text{mn} = N + 1 + M(2N + 1)$ .

3. The integer arrays `im(1:mn)` and `in(1:mn)` contain the  $m_j$  and  $n_j$ .

#### 1.4.11 halfmm(1:mn, regumm(1:mn) : regularization factor

1. The “regularization” factor, `halfmm(1:mn) = im(1:mn) * half`, is real.
2. This is used in `lforce`, `bfield`, `stzxyz`, `coords`, `jo00aa`, `ma00aa`, `sc00aa` and `tr00ab`.

#### 1.4.12 ime and ine : extended resolution Fourier mode identification

1. The “extended” Fourier resolution is defined by `lMpol = 4 Mpol`, `lNtor = 4Ntor`.

#### 1.4.13 mns, ims and ins : Fourier mode identification for straight-fieldline angle

#### 1.4.14 iRbc(1:mn, 0:Mvol, iZbs(1:mn, 0:Mvol, iRbs(1:mn, 0:Mvol and iZbc(1:mn, 0:Mvol : geometry

1. `iRbc`, `iZbs`, `iRbs` and `iZbc` : Fourier harmonics of interface geometry;
2. `iVns`, `iVnc`, `iBns` and `iBns` : Fourier harmonics of normal field at computational boundary;

#### 1.4.15 ajk : construction of coordinate axis

1. This is only used in `rzaxis` to perform the poloidal integration and is defined quite simply:  
 $\text{ajk}[i] \equiv 2\pi$  if  $m_i = 0$ , and  
 $\text{ajk}[i] \equiv 0$  if  $m_i \neq 0$ .

### 1.5 subroutine wrtend

1. The restart file is written.