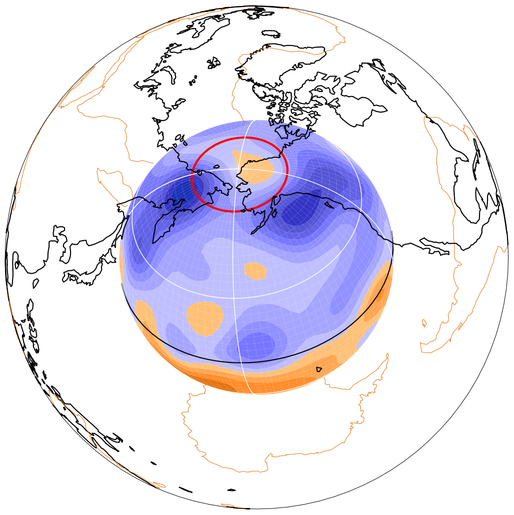
LEOPACK



Basic Equations and Numerical Representation

Steven J. Gibbons, Oslo

Original document: November 21st, 2001. Updated: November 20th, 2022.

1 Fundamentals

2 The basic equations

The programs in **LEOPACK** do not specifically refer to any physical quantities (e.g. kinematic viscosity, ν, magnetic diffusivity, η, or thermal expansivity, α) or indeed any dimensionless numbers (e.g. Rayleigh number, R, Ekman number, E, or Prandtl number, Pr). Instead, in order to accomodate the widest possible range of scalings, the programs multiply terms in the MHD equations by arbitrarily named constants which are described below and must be carefully specified by the user. The aim of this section is to explain how these scalings relate to physical quantities.

2.1 The heat equation

The equation defining the advection of heat is (see [GR87])

$$\frac{\partial T}{\partial t} + \boldsymbol{u}.\nabla T = \kappa \nabla^2 T + \frac{q}{C_p \rho} \tag{1}$$

where $\boldsymbol{u}$ is the fluid flow, T the temperature, κ the thermal diffusivity ($\mathrm{m^2 s^{-1}}$), q the rate of local heating ($\mathrm{Jm^{-3}s^{-1}}$), C_p the specific heat capacity ($\mathrm{Jkg^{-1}K^{-1}}$) and ρ the density ($\mathrm{kgm^{-3}}$).

The convection codes assume that the temperature, T, is expressed as follows:-

$$T(r,t,\theta,\phi) = T_0(r) + T_1(r,t,\theta,\phi) \tag{2}$$

The steady, basic state temperature distribution, $T_0(r)$, is given the form

$$T_0(r) = -\frac{1}{2} b_1 r^2 + \frac{b_2}{r} + b_3 \tag{3}$$

where b_1, b_2 and b_3 are constants. Its purpose is to define the temperature profile for the sphere or spherical shell, incorporating any internal heating sources. It satisfies

$$\nabla T_0 = -\left(b_1 r + b_2 r^{-2} \right) \boldsymbol{e}_r, \tag{4}$$

where $\boldsymbol{e}_r$ is the unit vector in the radial direction, and

$$\nabla^2 T_0 = -3 b_1. \tag{5}$$

If we substitute the definition (2) into Equation (1) and apply (4) and (5) we derive

$$\frac{\partial T_1}{\partial t} \quad = \quad \kappa \nabla^2 T_1 - 3\kappa b_1 + \frac{q}{C_p \rho} + \boldsymbol{u}.\left(b_1 r + b_2 r^{-2} \right) \boldsymbol{e}_r - \boldsymbol{u}.\nabla T_1 \tag{6}$$

It is now clear that the constant b_1 defines the sources of internal heating with

$$b_1 = \frac{q}{3C_p\rho\kappa}. \tag{7}$$

If there are no internal heating sources, then $q = 0$ and hence $b_1 = 0$. The constant b_2 is chosen appropriately for systems which have a simple temperature gradient from the inner to the outer boundary.

For numerical simplicity, it is best to solve for temperature functions with homogeneous boundary conditions. We therefore decompose T_1, the perturbation from the basic state temperature,

$$T_1(r,t,\theta,\phi) = \Theta(r,t,\theta,\phi) + \varepsilon T_\mathrm{a}(r,\theta,\phi). \tag{8}$$

Θ is the function which is solved for in all of the calculations. T_a is an additional temperature which is imposed if, for example, an inhomogeneous heat-flux at the outer boundary is required.

If we denote the radial component of the velocity u_r, then applying Equation (8) to Equation (6) gives us the heat equation as applied in all of the programs:

$$c_a\frac{\partial\Theta}{\partial t} \;=\; c_d\nabla^2(\Theta+\varepsilon T_\mathrm{a}) + b_1 u_r r + b_2\frac{u_r}{r^2} - c_c\boldsymbol{u}.\nabla(\Theta+\varepsilon T_\mathrm{a}) \tag{9}$$

The constants c_a, b_1, b_2, c_c and c_d are arbitrarily named, with no physical meaning attatched to them. Their use simply allows for any scaling to be applied to the equations. In the codes, c_a is stored in the double precision variable `CA`; and similarly with b_1 (`CB1`), b_2 (`CB2`), c_c (`CC`) and c_d (`CD`). Many of the codes are restricted to uniform thermal boundaries and so only work for $\varepsilon = 0$. Codes which are designed to implement inhomogeneous thermal boundaries usually denote ε with the double precision variable `SCAL`.

2.2 The momentum equation

In the Boussinesq approximation, all density variations except those with respect to the buoyancy force are considered to be negligible, and following the analysis of [GR87], the momentum equation is written

$$\frac{\partial\boldsymbol{u}}{\partial t} + \boldsymbol{u}.\nabla\boldsymbol{u} + 2\boldsymbol{\Omega}\times\boldsymbol{u} = -\nabla\tilde{\omega} + \frac{\delta\rho}{\rho_0}\boldsymbol{g} + \frac{\boldsymbol{J}\times\boldsymbol{B}}{\rho_0} + \nu\nabla^2\boldsymbol{u}, \tag{10}$$

where $\boldsymbol{J}$ and $\boldsymbol{B}$ are respectively the electric current and magnetic field. The scalar function $\tilde{\omega}$ combines the pressure, p, and the centrifugal force such that

$$\tilde{\omega} = \frac{p}{\rho} - \frac{1}{2}|\boldsymbol{\Omega}\times\boldsymbol{r}|^2,$$

and can be removed from the problem by taking the curl of Equation (10). The density variation $\delta\rho$ is expressed in terms of the thermal expansivity, α (K^{-1}), and T, the temperature perturbation from a well mixed state ($\rho = \rho_0$), to give

$$\frac{\delta\rho}{\rho_0} = -\alpha T. \tag{11}$$

The acceleration due to gravity, $\boldsymbol{g}$ is written in terms of the radial vector $\boldsymbol{r}$ as

$$\boldsymbol{g} = -\gamma\boldsymbol{r}, \tag{12}$$

for a constant γ, (s^{-2}). The linear dependence of $\boldsymbol{g}$ on r is a good approximation for the core (see for example [DA81] or [And89]), but would not be appropriate for the mantle. ν is the viscosity (m^{2}s^{-1}) and $\boldsymbol{\Omega} = \Omega\boldsymbol{k}$ is the rotation vector, in terms of the unit vector, $\boldsymbol{k}$. The electric current is related to the magnetic field by

$$\nabla \times \boldsymbol{B} = \mu\boldsymbol{J} \tag{13}$$

where μ is the magnetic permeability and assumed to equal μ_0, the magnetic permeability of free space everywhere.

In order to eliminate the pressure gradient from the momentum equation, we take the curl of Equation (10) and apply equations (11) and (12). If we denote the vorticity (the curl of $\boldsymbol{u}$) by $\boldsymbol{\omega}$, then our vorticity equation becomes

$$\begin{aligned} \frac{\partial\boldsymbol{\omega}}{\partial t} &= -\nabla \times (\boldsymbol{u}.\nabla\boldsymbol{u}) - 2\Omega\nabla \times (\boldsymbol{k} \times \boldsymbol{u}) \\ &\quad +\alpha\gamma\nabla \times (T\boldsymbol{r}) + \frac{1}{\rho\mu_0}\nabla \times [(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}] + \nu\nabla^2\boldsymbol{\omega}. \end{aligned} \tag{14}$$

It is assumed here that the kinematic viscosity, ν, is not a function of space. The basic state temperature, T_0, is a function of radius alone and therefore cannot contribute to the buoyancy term in the vorticity equation. Giving arbitrarily defined names to the scalings which multiply the terms in our equation (14), we write the curl of the momentum equation

$$\begin{aligned} c_e\frac{\partial\boldsymbol{\omega}}{\partial t} &= -c_f\nabla \times (\boldsymbol{u}.\nabla\boldsymbol{u}) - c_g\nabla \times (\boldsymbol{k} \times \boldsymbol{u}) \\ &\quad +c_h\nabla \times [(\Theta + \varepsilon T_{\mathrm{a}})\boldsymbol{r}] + c_j\nabla \times [(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}] + c_i\nabla^2\boldsymbol{\omega}. \end{aligned} \tag{15}$$

There are two vector quantities in the momentum equation, the velocity $\boldsymbol{u}$ and the magnetic field $\boldsymbol{B}$. $\boldsymbol{B}$ must always satisfy the solenoidal condition

$$\nabla.\boldsymbol{B} = 0 \tag{16}$$

and similarly, for a Boussinesq fluid, $\boldsymbol{u}$ must satisfy

$$\nabla.\boldsymbol{u} = 0. \tag{17}$$

We can therefore express both velocity and magnetic field in poloidal/toroidal decompositions

$$\boldsymbol{B} = \nabla \times \nabla \times \left[\quad {}^{P}B(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right] \ + \ \nabla \times \left[\quad {}^{T}B(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right] \tag{18}$$

and

$$\boldsymbol{u} = \nabla \times \nabla \times \left[\quad {}^{P}v(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right] \ + \ \nabla \times \left[\quad {}^{T}v(r,t,\theta,\phi)\ \boldsymbol{r} \quad \right]. \tag{19}$$

Note that these definitions are different from those of, for example Bullard and Gellman - [BG54], who use the unit radial vector, $\hat{\boldsymbol{r}}$, instead of $\boldsymbol{r}$.

2.3 The induction equation

The equation describing the evolution of a magnetic field, $\boldsymbol{B}$, in a conducting fluid with velocity $\boldsymbol{u}$ is derived from the pre-Maxwell equations

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \ , \quad \nabla \times \boldsymbol{B} = \mu \boldsymbol{J} \quad \text{and} \quad \nabla . \boldsymbol{B} = 0 \ , \tag{20}$$

and Ohm's law

$$\boldsymbol{J} = \sigma(\boldsymbol{E} + \boldsymbol{u} \times \boldsymbol{B}), \tag{21}$$

where $\boldsymbol{E}$ is the electric field and σ the electrical conductivity. The pre-Maxwell forms are used since the displacement current, $\partial \boldsymbol{E}/\partial t$, will be negligible for the relatively slow variations appropriate for the Earth. Assuming the electrical conductivity to be a constant, taking the curl of Equation (21) and applying the relations of (20) together with the vector identity

$$\nabla \times (\nabla \times \boldsymbol{V}) = \nabla(\nabla . \boldsymbol{V}) - \nabla^2 \boldsymbol{V},$$

gives the induction equation

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + \frac{1}{\mu_0 \sigma} \nabla^2 \boldsymbol{B}. \tag{22}$$

The generalised form of the induction equation, as used by the programs, is

$$c_k \frac{\partial \boldsymbol{B}}{\partial t} = c_m \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + c_l \nabla^2 \boldsymbol{B}. \tag{23}$$

As in equations (9) and (15), the constants c_k, c_m and c_l are arbitrarily named with no physical implications intended.

2.4 EXERCISE: Parameter scalings

The parameters c_a, b_1, b_2, c_c, c_d, c_e, c_f, c_g, c_h, c_i, c_j, c_k, c_l and c_m are all defined by equations (9), (15) and (23).

The numerical dynamo benchmark study, [CAC$^+$01], gives the following information for the Case 1 (insulating inner core) benchmark study.

The ratio of inner radius r_i to outer radius r_o is set to 0.35. Temperature is fixed to T_o and $T_\mathrm{o} + \Delta T$ on the outer and inner boundaries respectively. The Boussinesq approximation is used and gravity varies linearly with radius. The equations are scaled with $D = r_\mathrm{o} - r_\mathrm{i}$ as the fundamental length scale. The time-scale is D^2/ν, with ν the kinematic viscosity. ν/D is the scale for the velocity $\boldsymbol{u}$, and ΔT for the temperature T. The scaled temperature on the outer boundary is zero. Magnetic induction, $\boldsymbol{B}$, is scaled by $(\rho\mu\eta\Omega)^{1/2}$, where ρ is the density, μ the magnetic permeability, and Ω the basic rotation rate about the z-axis. The non-hydrostatic pressure is scaled by $\rho\nu\Omega$. The scaled equations are

$$E\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}.\nabla\boldsymbol{u} - \nabla^2\boldsymbol{u}\right) + 2\boldsymbol{k}\times\boldsymbol{u} + \nabla P = Ra\frac{\boldsymbol{r}}{r_\mathrm{o}}T + \frac{(\nabla\times\boldsymbol{B})\times\boldsymbol{B}}{Pm} \tag{24}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla\times(\boldsymbol{u}\times\boldsymbol{B}) + \frac{1}{Pm}\nabla^2\boldsymbol{B} \tag{25}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u}.\nabla T = \frac{1}{Pr}\nabla^2 T \tag{26}$$

$$\nabla.\boldsymbol{u} = \nabla.\boldsymbol{B} = 0 \tag{27}$$

Non-dimensional control parameters are the modified Rayleigh number

$$Ra = \frac{\alpha g_0 \Delta T D}{\nu\Omega} \tag{28}$$

where α is the thermal expansion coefficient and g_0 gravity at the outer radius, the Ekman number (note the non-standard definition!)

$$E = \frac{\nu}{\Omega D^2} \tag{29}$$

the Prandtl number

$$Pr = \frac{\nu}{\kappa} \tag{30}$$

where κ is the thermal diffusivity, and the magnetic Prandtl number

$$Pm = \frac{\nu}{\eta}. \tag{31}$$

3 Representation of spherical functions

The solutions to equations (9), (15) and (23) are all three dimensional functions of radius r, co-latitude θ and longitude ϕ. The temperature is obtained via the scalar function $\Theta(r,t,\theta,\phi)$ (see Equation 8). From equations (18) and (19), we see that both velocity and magnetic field are completely specified by the four scalar functions ${}^{P}B(r,t,\theta,\phi)$, ${}^{T}B(r,t,\theta,\phi)$, ${}^{P}v(r,t,\theta,\phi)$ and ${}^{T}v(r,t,\theta,\phi)$. Our solution at any given time, t, is then completely specified by five scalar functions. The dependence in the θ and ϕ directions is defined by expanding ${}^{P}v$, ${}^{T}v$, Θ, ${}^{P}B$ and ${}^{T}B$ in terms of spherical harmonics. The truncated expansions are as follows:

$$
\begin{aligned}
{}^{P}v(r,t,\theta,\phi) &= \sum_{l=1}^{L1}\sum_{m=0}^{M(l)} \left[{}^{P}v_l^{mc}(r,t)\cos m\phi + {}^{P}v_l^{ms}(r,t)\sin m\phi\right] P_l^m(\cos\theta) && (32)\\
{}^{T}v(r,t,\theta,\phi) &= \sum_{l=1}^{L2}\sum_{m=0}^{M(l)} \left[{}^{T}v_l^{mc}(r,t)\cos m\phi + {}^{T}v_l^{ms}(r,t)\sin m\phi\right] P_l^m(\cos\theta) && (33)\\
\Theta(r,t,\theta,\phi) &= \sum_{l=0}^{L3}\sum_{m=0}^{M(l)} \left[\Theta_l^{mc}(r,t)\cos m\phi + \Theta_l^{ms}(r,t)\sin m\phi\right] P_l^m(\cos\theta) && (34)\\
{}^{P}B(r,t,\theta,\phi) &= \sum_{l=1}^{L4}\sum_{m=0}^{M(l)} \left[{}^{P}B_l^{mc}(r,t)\cos m\phi + {}^{P}B_l^{ms}(r,t)\sin m\phi\right] P_l^m(\cos\theta) && (35)\\
{}^{T}B(r,t,\theta,\phi) &= \sum_{l=1}^{L5}\sum_{m=0}^{M(l)} \left[{}^{T}B_l^{mc}(r,t)\cos m\phi + {}^{T}B_l^{ms}(r,t)\sin m\phi\right] P_l^m(\cos\theta) && (36)
\end{aligned}
$$

In all applications dealt with here, the associated Legendre function $P_l^m(\cos\theta)$ satisfies the Schmidt quasi-normalisation condition

$$\int_0^\pi \left[P_l^m(\cos\theta)\right]^2 \sin\theta d\theta = \frac{2(2-\delta_{m0})}{2l+1}. \qquad (37)$$

(Further details on spherical harmonics are found in the document `DOC_sphervec.pdf` in the `additional_documentation` folder in the `LEOPACK-2022-revision` github repository. Note that only the temperature variable, Θ, has an $l=0$ term: this is absent from the other terms due to the solenoidal condition (16) and the incompressibility condition (17). In addition, the functions ${}^{P}v_l^{0s}$, ${}^{T}v_l^{0s}$, Θ_l^{0s}, ${}^{P}B_l^{0s}$ and ${}^{T}B_l^{0s}$ are all clearly non-existant. If `X` contains the double precision numerical representation of $\cos\theta$, then the value of $P_l^m(\cos\theta)$ is returned by the double precision function `SHMPLG( L, M, X )`. In more general practical situations, i.e. in the simulation codes, arrays of associated Legendre functions are calculated by the subroutine `SCHNLA` (see file `DOC_appkeysubs.pdf`).

The spherical harmonic expansions (32), (33), (34), (35) and (36) are ofcourse infinite and so the the truncation parameters $L1$, $L2$, $L3$, $L4$ and $L5$ should be made as large as is required to be able to adequately represent the actual solution.

Larger values for these parameters ofcourse lead to a higher computational cost and so a certain amount of judgement is required to ensure that convergence of the solution at different values of $L1$ etc. is acceptable.

3.1 Representation in longitude, ϕ

For a true three-dimensional solution, the integer function M is given by

$$M(l) = l. \tag{38}$$

However, there are many cases where it is valid to restrict the resolution in the ϕ direction. If the energy spectra in m decay much faster than in l, it may be appropriate to impose a maxmimum value of m, $M_{\max}$ for instance, such that

$$M(l) = min(l, M_{\max}). \tag{39}$$

This can lead to significant time savings by reducing the size of the (r, θ, ϕ) grid which needs transforming, and especially in reducing the time spent in the Fast Fourier Transforms. Also, we may impose a fundamental wavenumber, m_0, such that only m which are integer multiples of m_0 are included in the solution. In some instances, this may actually be valid for the physical solution which may display natural symmetry properties. An example of this is the dynamo benchmark solution of [CAC$^+$01] which displays a four-fold symmetry in ϕ and can therefore be represented by a spherical harmonic expansion containing only m which are integer multiples of 4. A fully three dimensional solution for the model parameters described in [CAC$^+$01] will integrate towards a solution which is zero for all m which are not multiples of m_0. Care must ofcourse be taken as the symmetry is likely to be broken when the physical parameters are changed and other symmetries are excited.

Even for cases where the physical solution is not exactly described by a limited set of wavenumbers, much can be learned from solutions with a reduced resolution in the ϕ-direction. Many authors (for example [SJ99] and references therein) have obtained great insights by restricting the solution to two azimuthal wavenumbers: $m = 0$ and one non-zero wavenumber. These have been termed 2.5D dynamos.

The transform routines are so written that if the integer variable `M0` contains an integer greater than 1, the non-linear terms are evaluated in a "half-space" which exploits this symmetry and thus makes great savings in CPU time. For example, if $M_{\max} = 24$ and $m_0 = 4$, then our solution vector will contain the wavenumbers 0, 4, 8, 12, 16, 20 and 24. The transform routines interpret this as evaluating the wavenumbers 0, 1, 2, 3, 4, 5 and 6 in the half-space $0 \leq \phi \leq 2\pi/m_0$ as opposed to the actual wavenumbers in the space $0 \leq \phi \leq 2\pi$. This reduces significantly the number of points in ϕ (N_ϕ) necessary to perform an accurate Fast Fourier Transform - FFT. (Note that this may or may not be a factor of m_0 due the requirement that N_ϕ is a power of 2. It may be more and may be less.)

The routine to use in order to choose an appropriate value of N_ϕ is `NPHPF`:

```
CALL NPHPF( MMAX, MO, NPHP, NPHMAX )
```

where `MMAX` is $M_{\max}$, the highest wavenumber in the spectral expansion, `MO` is m_0, `NPHMAX` is the limit allowed for N_ϕ in the program's array definitions, and N_ϕ is returned in the integer variable `NPHP` as the smallest number which is greater than $2M_{\max}/m_0$ and is a power of 2. The condition necessary for the FFT is that

$$f(\phi) = f(\phi + \frac{2\pi}{m_0}). \tag{40}$$

The point ϕ_i is given by

$$\phi_i = (i-1)\Delta\phi \tag{41}$$

where

$$\Delta\phi = \frac{2\pi}{N_\phi m_0}. \tag{42}$$

Typically, the wavenumbers are of the order 20 to 30 in the parameter range we have looked at. If $m_0 = M_{\max}$, as is the case in a 2.5D simulation, then $N_\phi = 4$ will always be sufficient, whichever non-zero wavenumber is used.

3.2 Representation in co-latitude, θ

The full three-dimensional solution includes all the terms in the truncated expansions (32), (33), (34), (35) and (36). We can make many useful time-savings by restricting the solution vectors to certain symmetries with respect to the equator.

For the full details of the symmetry arguments for solutions to the dynamo equations, I refer the reader to [GZ93]. A typical configuration is where the velocity, $\boldsymbol{u}$, is symmetric about the equator:

$$\boldsymbol{u}(r,\theta,\phi) = \boldsymbol{u}(r,\pi-\theta,\phi) \tag{43}$$

and the magnetic field, $\boldsymbol{B}$, is anti-symmetric about the equator:

$$\boldsymbol{B}(r,\theta,\phi) = -\boldsymbol{B}(r,\pi-\theta,\phi). \tag{44}$$

Note that the condition (43) implies that

$$\begin{array}{rcl} u_r(r,\theta,\phi) &=& u_r(r,\pi-\theta,\phi) \\ u_\theta(r,\theta,\phi) &=& -u_\theta(r,\pi-\theta,\phi) \\ u_\phi(r,\theta,\phi) &=& u_\phi(r,\pi-\theta,\phi) \end{array} \tag{45}$$

and that condition (44) implies that

$$\begin{array}{rcl} B_r(r,\theta,\phi) & = & -B_r(r,\pi-\theta,\phi) \\ B_\theta(r,\theta,\phi) & = & B_\theta(r,\pi-\theta,\phi) \\ B_\phi(r,\theta,\phi) & = & -B_\phi(r,\pi-\theta,\phi). \end{array} \tag{46}$$

The alternative situation,

$$\begin{array}{rcl} \boldsymbol{u}(r,\theta,\phi) & = & -\boldsymbol{u}(r,\pi-\theta,\phi) \\ \boldsymbol{B}(r,\theta,\phi) & = & \boldsymbol{B}(r,\pi-\theta,\phi) \end{array}$$

is not a valid solution to the full dynamo problem due to the non-linear terms in the momentum equation (see [GZ93]).

In terms of our poloidal and toroidal scalars, the condition (43) implies that the expansion (32) only contains terms with even $(l-m)$, and the expansion (33) only contains terms with odd $(l-m)$. Similarly, the condition (44) implies that the expansion (35) only contains terms with odd $(l-m)$, and the expansion (36) only contains terms with even $(l-m)$. In general, the temperature Θ should obey the same symmetry as is imposed upon the poloidal velocity scalar, ${}^Pv$. The heat source and buoyancy terms in equations (9) and (15) couple the spherical harmonics in the expansion (32) uniquely with harmonics in Equation (34). The one exception to this is the $l=m=0$ harmonic which only receives contributions from the non-linear term $\boldsymbol{u}.\nabla\Theta$.

3.3 Representation in radius, r

All of the codes presented here use finite differences to approximate radial derivatives. Let $f_\alpha(r)$ denote any of the radial functions in the expansions (32), (33), (34), (35) and (36). α is an integer which uniquely identifies which component of the expansions f refers to. α is an index which points to locations within property arrays - which will be covered in detail in the following section. (For example, $\alpha=7$ might mean that $f_\alpha(r)$ is the radial function ${}^PB_3^{2c}(r)$. The 7th component of each of the arrays described in Section 3.4 would provide information as to the identity of this radial function.)

The number of radial grid nodes is given by the integer number, `NR` (N_r). As a rule, I would never do a calculation with fewer than 25 grid nodes, and the actual number of grid nodes necessary to adequately resolve a solution varies enormously with the parameter regime being studied and so some experimentation is generally required to find a suitable number.

The radii at which the N_r grid nodes lie are stored in the double precision array `XARR`. The radius of the inner boundary, r_i, is stored in `XARR( 1 )`, and the radius at the outer boundary, r_o, is stored in `XARR( NR )`. More generally, the radius at grid node i, r_i, with

$$r_\text{i} \leq r_i \leq r_\text{o} \tag{47}$$

is stored in `XARR( i )`: the difference between $r_{\rm i}$ and r_i being appreciated. There is, in principle, no restriction on the values of r_i which are permissible, other than the condition that $j_1 < j_2$ ought to imply that $r_{j_1} < r_{j_2}$.

The user may provide any values to the array `XARR`. Throughout my use of the codes, I have used three subroutines to set the elements of `XARR`:-

- `ESNAAS` Equally spaced radial nodes.
- `ZCPAAS` Zeroes of Chebyshev polynomial T_{N_r-2} transformed from the interval $(-1,1)$ to the interval $(r_{\rm i}, r_{\rm o})$. (Adapted from Daniele Funaro's routine ZECHGA.) This concentrates radial grid nodes close to the inner and outer boundaries.
- `ZCPAA2` Non-negative zeroes of Chebyshev polynomial T_{2N_r-2} transformed from the interval $(0,1)$ to the interval $(r_{\rm i}, r_{\rm o})$. (Adapted from Daniele Funaro's routine ZECHGA.) This concentrates radial grid nodes close to the outer boundary.

Figure (1) shows the different kinds of spacings possible with these three routines. The routine `ZCPAA2` was developed for the treatment of cases with no inner core: additional resolution close to the origin was never found to be very helpful.

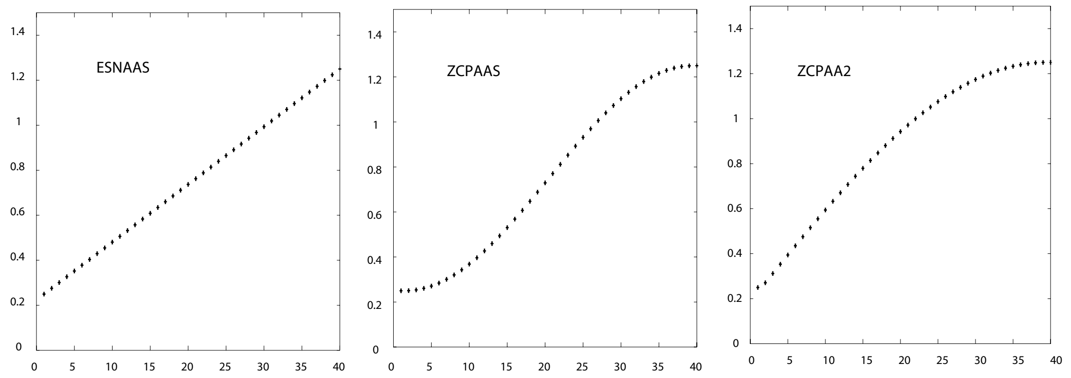


Figure 1: Distributions of radial grid nodes with $r_{\rm i} = 0.25$, $r_{\rm o} = 1.25$ and $N_r = 40$. Subroutine used to generate r values as shown.

Ofcourse, there may be circumstances where completely different definitions give optimal results. For example, Emmanuel Dormy devised a scheme where nodes close to the boundaries follow a geometric progression whilst those between the boundaries are equally spaced. Full details are found in [Dor97].

The radial spacings array, `XARR`, is saved to file using the routine `XARRWT`:

```
CALL XARRWT( NR, XARR, LU, FNAME, IFORM )
```

LU is the logical file unit (essentially an arbitrary integer: positive, neither 5 or 6, and not being used to label any other files). FNAME is a character string which is the filename: the convention in the work here is to terminate the filenames with the characters ".xarr". Finally, the integer flag IFORM specifies an output format for the NR data which will be written. The only value for IFORM I have ever applied is IFORM = 1 which writes the data in the format (5(1PD16.7)). If other formats are required for these files, additional options for IFORM should be made. Any changes made to the routine XARRWT should be well documented and accompanied by an appropriate change to the routine XARRRD. Please inform me of any changes made to this code and I can keep my copies up to date with changes if it is useful to do so.

An example .xarr file is as follows:

```
40    1
5.3846154D-01   5.3888866D-01   5.4230125D-01   5.4910312D-01   5.5924780D-01
5.7266599D-01   5.8926605D-01   6.0893456D-01   6.3153718D-01   6.5691950D-01
6.8490814D-01   7.1531191D-01   7.4792312D-01   7.8251901D-01   8.1886323D-01
8.5670754D-01   8.9579341D-01   9.3585385D-01   9.7661521D-01   1.0177990D+00
1.0591240D+00   1.1003078D+00   1.1410692D+00   1.1811296D+00   1.2202155D+00
1.2580598D+00   1.2944040D+00   1.3289999D+00   1.3616111D+00   1.3920149D+00
1.4200035D+00   1.4453859D+00   1.4679885D+00   1.4876570D+00   1.5042570D+00
1.5176752D+00   1.5278199D+00   1.5346218D+00   1.5380344D+00   1.5384615D+00
```

The first line contains two numbers, NR and IFORM, and the following lines contain the radius values in the format defined by IFORM. This corresponds to 40 grid nodes, with Chebyshev spacing (ZCPAAS) for $r_\mathrm{i}/r_\mathrm{o} = 0.35$ and $r_\mathrm{o} - r_\mathrm{i} = 1.0$.

Radial spacings files are read into programs using the routine XARRRD:

```
CALL XARRRD( NR, NRMAX, XARR, LU, FNAME )
```

where NRMAX is the maximum number of radial grid nodes as defined in the array declarations. XARRRD reads the top line of the .xarr file first and will abort if (a) the file describes more grid nodes than the program has available or (b) the data format is not recognised. NR will overwrite any value previously held.

3.4 Describing the solution vector(s)

A solution vector contains N_h spherical harmonic radial functions, each of which has a value at every one of N_r radial grid nodes. Its total length is therefore $N_h \times N_r$. They can consist of any set of spherical harmonics - it is the user's responsibility to ensure that the solution vector consists of meaningful definitions! We shall denote the j^{th} spherical harmonic radial function $f_j(r)$, and the value of this function at the i^{th} radial grid node, $f_j(r_i)$.

For each j, with $1 \leq j \leq N_h$, $f_j(r)$ is one of the radial functions in the expansions (32), (33), (34), (35) and (36). Which one of these functions is stored in $f_j(r)$ is determined by the values of the j^{th} elements in the integer arrays MHT, MHL and MHM.

The array MHT defines what type of function is stored in the $f_j(r)$ functions.

MHT(j) = 1 $\rightarrow f_j(r)$ is poloidal velocity (${}^P v$).
MHT(j) = 2 $\rightarrow f_j(r)$ is toroidal velocity (${}^T v$).
MHT(j) = 3 $\rightarrow f_j(r)$ is temperature (Θ).
MHT(j) = 4 $\rightarrow f_j(r)$ is poloidal magnetic field (${}^P B$).
MHT(j) = 5 $\rightarrow f_j(r)$ is toroidal magnetic field (${}^T B$).

The array MHL gives the spherical harmonic degree, l, of the $P_l^m(\cos\theta)$ which the radial functions multiply.

If radial function j multiplies $\cos m\phi$, then MHM(j) = M. If radial function j multiplies $\sin m\phi$, then MHM(j) = -M.

One final array is necessary to describe the arrangement of the solution vector. The integer array INARR contains 3 elements. INARR(2) = NR and INARR(3) = NH: the integer function INDFUN also gives full details about INARR. INARR(1) is set to the integer flag IFORMF which indicates the order in which the elements are stored. Depending upon the problem being attempted, it may be best to store the $f_j(r_i)$ in one of two different configurations. Given that all the codes use finite difference schemes for radial derivatives, the radial derivatives of the function $f_j(r)$ at radius r_i will depend upon $f_j(r)$ at only a limited number of grid nodes. (We will see later that for the universal parameter NBN - the number of bounding nodes - the radial derivatives of $f_j(r)$ at r_i will depend upon the values $f_j(r_k)$ such that $i - \mathrm{NBN} \leq k \leq i + \mathrm{NBN}$.) The goal of arranging the solution vector is to minimize the size of banded matrices arising.

If the system of equations being solved in the linear discretisation couples the radial functions f_{j_1} and f_{j_2} with j_1 and j_2 distinct, then the bandwidth for the equation defining $f_j(r_i)$ must cover all of the f_l with $1 \leq l \leq N_h$ for each of the grid nodes r_k from $r_{i-\mathrm{NBN}}$ to $r_{i+\mathrm{NBN}}$. We therefore need to store the solution vector as displayed in Figure (2 a) and set the integer flag IFORMF = 3. This configuration is necessary in the eigenvalue problems for thermal, shear or magnetic instabilities, and in the programs which use Newton-Raphson iteration to solve for steady, boundary-locked flows.

If the system of equations being solved in the linear discretisation does not couple distinct radial functions, then we only require a bandwidth of 2*NBN + 1. We then arrange the solution vector as displayed in Figure (2 b) and set IFORMF = 4. This configuration is necessary when time-stepping as, in these codes, only the diffusion terms are treated implicitly.

If the double precision array VEC now contains our solution vector, then our solution is now completely described by the arrays INARR, VEC, XARR, MHT, MHL and MHM.

Example:-

If

```
 IND       = INDFUN( i, j, INARR )
 MHT( j ) = 2
 MHL( j ) = 6
 MHM( j ) = -3
```

then `VEC( IND )` contains the value ${}^{T}v_6^{3s}(r_i)$ from Equation (33), with r_i stored in `XARR( i )`.

Solution vectors are written out to file using the routine `SVFWT`:

```
CALL SVFWT( INARR, LU, IFORM, SV, FNAME )
```

where `SV` is the double precision array of length `INARR( 2 )*INARR( 3 )` which stores the solution vector. Other parameters are dealt with in the coverage of the subroutine `XARRWT` in Section (3.3).

An example of a solution vector file is as follows:

```
  4   40    2    1
 0.0000000D+00   8.0466569D-02   1.6041128D-01   2.3931566D-01   3.1666799D-01
 3.9196661D-01   4.6472317D-01   5.3446583D-01   6.0074226D-01   6.6312266D-01
 7.2120245D-01   7.7460496D-01   8.2298387D-01   8.6602540D-01   9.0345043D-01
 9.3501624D-01   9.6051811D-01   9.7979065D-01   9.9270887D-01   9.9918900D-01
 9.9918900D-01   9.9270887D-01   9.7979065D-01   9.6051811D-01   9.3501624D-01
 9.0345043D-01   8.6602540D-01   8.2298387D-01   7.7460496D-01   7.2120245D-01
 6.6312266D-01   6.0074226D-01   5.3446583D-01   4.6472317D-01   3.9196661D-01
 3.1666799D-01   2.3931566D-01   1.6041128D-01   8.0466569D-02   1.2246468D-16
 0.0000000D+00   2.6822190D-04   1.0694085D-03   2.3931566D-03   4.2222399D-03
 6.5327768D-03   9.2944634D-03   1.2470869D-02   1.6019794D-02   1.9893680D-02
 2.4040082D-02   2.8402182D-02   3.2919355D-02   3.7527767D-02   4.2161020D-02
 4.6750812D-02   5.1227633D-02   5.5521470D-02   5.9562532D-02   6.3281970D-02
 6.6612600D-02   6.9489621D-02   7.1851314D-02   7.3639722D-02   7.4801299D-02
 7.5287536D-02   7.5055535D-02   7.4068548D-02   7.2296463D-02   6.9716237D-02
 6.6312266D-02   6.2076701D-02   5.7009688D-02   5.1119549D-02   4.4422882D-02
 3.6944599D-02   2.8717880D-02   1.9784058D-02   1.0192432D-02   1.5920408D-17
```

which accompanies the radial spacings file:

```
  40    1
 0.0000000D+00   2.5641026D-02   5.1282051D-02   7.6923077D-02   1.0256410D-01
 1.2820513D-01   1.5384615D-01   1.7948718D-01   2.0512821D-01   2.3076923D-01
 2.5641026D-01   2.8205128D-01   3.0769231D-01   3.3333333D-01   3.5897436D-01
 3.8461538D-01   4.1025641D-01   4.3589744D-01   4.6153846D-01   4.8717949D-01
 5.1282051D-01   5.3846154D-01   5.6410256D-01   5.8974359D-01   6.1538462D-01
 6.4102564D-01   6.6666667D-01   6.9230769D-01   7.1794872D-01   7.4358974D-01
 7.6923077D-01   7.9487179D-01   8.2051282D-01   8.4615385D-01   8.7179487D-01
 8.9743590D-01   9.2307692D-01   9.4871795D-01   9.7435897D-01   1.0000000D+00
```

The first line of the `.vecs` file gives 4 integers which are `IFORMF`, `NR`, `NH` and `IFORM`.

- `INARR( 1 ) = 4` indicates that this solution vector is arranged as shown in Figure (2 b).

- `INARR( 2 ) = 40` indicates that there are 40 radial grid nodes.
- `INARR( 3 ) = 2` indicates that there are 2 spherical harmonic radial functions.
- `IFORM = 1` indicates that the `40*2` elements of the solution vector are stored in the `(5(1PD16.7))` format.

The `.xarr` file indicates that the 40 radial nodes are equally spaced between 0.0 and 1.0. There is not enough information from these two files to work out to which spherical harmonics the radial functions belong.

Reading a solution vector from a file requires the use of the subroutine `SVFRD`:

```
CALL SVFRD( INARR, LU, NRMAX, SV, FNAME )
```

On calling `SVFRD`, the number of spherical harmonic radial functions, N_h, must already be stored in `INARR( 3 )`: this ensures that the vector file being read is compatible with the corresponding integer arrays.

Notes:-

(a) The more recent time-stepping codes read in single solution vectors, for example in the double precision array `SV` (which has dimensions `NR*NH`) with integer descriptor arrays `MHT( NH )`, `MHL( NH )`, `MHM( NH )` and `INARR( 3 )`. They then calculate how many harmonics are poloidal velocity (`NH1`), toroidal velocity (`NH2`), temperature (`NH3`), poloidal magnetic field (`NH4`) and toroidal magnetic field (`NH5`): such that `NH1 + NH2 + NH3 + NH4 + NH5 = NH`. The solution vector is then redistributed over the 5 arrays `SV1`, `SV2`, `SV3`, `SV4` and `SV5`. For example, `SV1( NH1*NR )` is described by `MT1( NH1 )`, `ML1( NH1 )`, `MM1( NH1 )` and INARR1(3) with `INARR1( 1 ) = 4`, `INARR1( 2 ) = NR` and `INARR1( 3 ) = NH1`. Although `IFORMF` needs to be 4 in order for the time-stepping code to process it, the initial vector `SV` may be of any configuration (i.e. `INARR( 1 )` can be either 3 or 4), and the code will deal with it: the subroutine `IIASCE` sets the integer arrays `MT1`, `ML1` and `MM1` whilst the routine `MC2SCV` exchanges information between the vectors `SV` and `SV1`.

(b) The values 1 and 2 are no longer valid for `IFORMF` - or `INARR( 1 )` in current versions of the code.

3.5 Interpolation of radial functions

Section (3.4) showed us how the solution vectors are stored. In the majority of the codes, the radial functions and their derivatives are only ever referred to at the N_r discrete grid points as stored in the array `XARR`. However, it is highly likely that we will need to be able to interpolate radial functions between the designated grid points. The most basic interpolation is ofcourse a straight line, although this will often be too poor an approximation and, besides, we may also want information about higher derivatives.

Consider an aribtrary function, $f(x)$. We may expand f as a Taylor series about a point x_0:

$$\begin{aligned} f(r_0 + h) \quad = \quad & f^{(0)}(x_0) \quad + \quad hf^{(1)}(x_0) \quad + \quad \frac{h^2}{2!}f^{(2)}(x_0) \quad + \\ & \frac{h^3}{3!}f^{(3)}(x_0) \quad + \quad ..... \quad + \quad \frac{h^n}{n!}f^{(n)}(x_0) \quad + \quad ..... \end{aligned} \tag{48}$$

Here, $f^{(n)}(x_0)$ denotes the n^{th} derivative of f with respect to x, evaulated at the point x_0. Similarly, if we have N_n such distinct points, x_i, with $1 \leq i \leq N_n$ (uniqueness of the x_i is essential) and

$$h_i = x_i - x_0 \tag{49}$$

then

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N_n}) \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1N_n} \\ d_{21} & d_{22} & \cdots & d_{2N_n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N_n1} & d_{N_n2} & \cdots & d_{N_nN_n} \end{bmatrix} \begin{bmatrix} f^{(0)}(x_0) \\ f^{(1)}(x_0) \\ \vdots \\ f^{(N_n-1)}(x_0) \end{bmatrix} \tag{50}$$

to within an error related to N_n. The element d_{ij} of the matrix $\boldsymbol{D}$ is given by

$$d_{ij} = \frac{h_i^{j-1}}{(j-1)!}. \tag{51}$$

Clearly, the inverse of this matrix gives us an expression for the derivatives (up to $N_n - 1$) of f evaluated at $x = x_0$:

$$\begin{bmatrix} f^{(0)}(x_0) \\ f^{(1)}(x_0) \\ \vdots \\ f^{(N_n-1)}(x_0) \end{bmatrix} = [\boldsymbol{D}]^{-1} \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N_n}) \end{bmatrix} \tag{52}$$

Note that this expression does not depend upon the grid nodes being uniformly spaced. The subroutine responsible for calculating these coefficients is `GFDCFD` (see source code and document `DOC_appkeysubs.pdf`). All the finite difference calculations performed in these codes are based upon this principle. If $\{x_1,\ x_2,\ \cdots,\ x_{N_n}\}$ are chosen to be a subset of $\{r_1,\ r_2,\ \cdots,\ r_{N_r}\}$, and x_0 a radius r with $r_{\rm i} \leq r \leq r_{\rm o}$, then coefficients calculated by `GFDCFD` will allow f and its derivatives to be calculated at this arbitrary radius. The subroutine `SVRINT` is designed for this purpose.

3.5.1 EXAMPLE: The function RTPFCE

The double precision function `RTPFCE` returns the value of the temperature, or any one of the vector components v_r, v_θ, v_ϕ, B_r, B_θ or B_ϕ at a given r (`RAD`), θ (`THE`) or ϕ (`PHI`) within the sphere. It uses the `SVRINT` subroutine to interpolate the spherical harmonic radial functions at the appropriate value of r. For example:

```
BTHE = RTPFCE( 5, INARR, NNDS, IWORK, MHT, MHL, MHM, RAD,
                 THE, PHI, VEC, XARR, WORK1, WORK2, COEFM )
```

The source code documentation describes all of the inputs.

4 The finite difference scheme

4.1 General finite difference formulae

Whilst `SVRINT` will give us good approximations for values of radial functions and their derivatives at arbitrary radii, it is highly inefficient to perform such operations within a simulation code. Therefore, in all of the main programs, radial functions and their derivatives are only ever evaluated at the grid nodes, r_i, as stored in the `XARR` array. In other words, not only are the $\{x_1, x_2, \cdots, x_{N_n}\}$ in Equation (52) a subset of $\{r_1, r_2, \cdots, r_{N_r}\}$, but x_0 is also one of these grid points.

The subroutine `FDCMBD` (see source code and the document `DOC_appkeysubs.pdf`) creates an array of finite difference coefficients for the radial derivatives of functions by the principle described in Section (3.5). `FDCMBD` limits the subset of radial grid nodes to $\{r_{\rm lc}, \cdots, r_{\rm rc}\}$, where lc is given in the calling sequence to `FDCMBD` by the integer variable `NLMC`, and rc is given by `NRMC`. The finite difference coefficients are calculated such that a derivative can be calculated for any r_i within the range $\{r_{\rm ln}, \cdots, r_{\rm rn}\}$, where ln $\geq$ lc and rn $\leq$ rc. ln and rn are respectively specified within the code by `NLMN` and `NRMN`.

For a radial grid node r_i, with $i \geq (\text{lc} + \text{NBN})$ and $i \leq (\text{rc} - \text{NBN})$, the number of nodes in the stencil, N_n, will be $2 \times \text{NBN} + 1$. For grid nodes r_i with $\text{lc} \leq i < (\text{lc} + \text{NBN})$ or $(\text{rc} - \text{NBN}) < i \leq \text{r}$, then, for the same bandwidth NBN, there are fewer grid nodes whose values can be used to calculate derivatives and our stencil size, N_n, is therefore smaller. In general, the size of our stencil for a grid node r_i is given by

$$N_n = \min(\text{NBN}, i - \text{lc}) \quad + \quad \min(\text{NBN}, \text{rc} - i) \quad + \quad 1. \tag{53}$$

It must be ensured that the minimum value of N_n occuring in the range $\text{ln} \leq i \leq \text{rn}$ is atleast one greater than the highest derivative required: and ideally somewhat more in order that any kind of accuracy is achieved.

4.2 Finite differences including boundary conditions

We are generally using finite difference formulae to solve ordinary differential equations in radius and we therefore need a way of implementing boundary conditions.

If a radial function is subject to homogeneous boundary conditions, then the value of the function at the boundary can be determined by the values of the function at neighbouring radial grid nodes.

The simplest case is that $f(r_\mathrm{i})$ ($f(r_1)$) or $f(r_\mathrm{o})$ ($f(r_{N_r})$) is equal to zero. However, we may also have a condition such as $df/dr = 0$. The value of f at the boundary (i.e. $f(r_1)$ or $f(r_{N_r})$) will not be zero but, with the help of a one-sided difference approximation for df/dr at the boundary, can be expressed in terms of $\{f(r_2), f(r_3), \cdots, f(r_{\mathrm{NBN}+1})\}$ for the inner boundary or $\{f(r_{N_r-1}), \cdots, f(r_{N_r-\mathrm{NBN}})\}$ at the outer boundary. In other words, we can replace Equation (52) with

$$\begin{bmatrix} f^{(0)}(x_0) \\ f^{(1)}(x_0) \\ \vdots \\ f^{(N_n-1)}(x_0) \end{bmatrix} = [\boldsymbol{D}]^{-1}\,[\boldsymbol{A}] \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N_n}) \end{bmatrix} \tag{54}$$

where that $\boldsymbol{A}$ matrix returns $f(x_1)$, $f(x_2)$ to $f(x_K)$ as a linear function of $f(x_2)$ to $f(x_K)$ only. $\boldsymbol{A}$ therefore has a column which is zero. Indeed, if there are two boundary conditions being imposed at a given boundary, then $\boldsymbol{A}$ returns $f(x_1)$, $f(x_2)$ to $f(x_K)$ as a linear function of $f(x_3)$ to $f(x_K)$ only. $\boldsymbol{A}$ therefore has two columns which are zero. If $f(r_1)$ is determined by the boundary conditions then we remove this node from the solution vector and simply solve for the values $\{f(r_2), f(r_3), \cdots, f(r_{\mathrm{NBN}+1})\}$. The elements of $\boldsymbol{A}$ modify the finite difference coefficients such that the boundary condition is implicit in the expressions for radial derivatives. In practice, this means that the the contributions from $f(r_1)$ are simply added to the coefficients for $\{r_2, r_3, \cdots, r_{\mathrm{NBN}+1}\}$, the exact manner of which is determined by the discretised boundary condition: the linear relationship between $f(r_1)$ and $f(r_2)$ to $f(r_{\mathrm{NBN}+1})$.

The nasty calculation of the $\boldsymbol{A}$ matrix in the **LEOPACK** codes is painlessly performed by the routine `LDGNMF` (see the file `DOC_appkeysubs.pdf`). This routine takes two integer flags which determine the boundary conditions to be applied at the inner and outer boundaries. The integer flags `IIBC` and `IOBC` can (currently) be set to the values as shown in tables (1) and (2).

The user never deals directly with **LDGNMF**; this routine is called from the most important of the finite difference coefficient routines, **SVFDCF** (see the file `DOC_appkeysubs.pdf`). Since the $\boldsymbol{A}$ matrix in Equation (54) is different for every boundary condition applied, a separate set of finite difference coefficients needs to be stored for every kind of radial function which satisfies a different boundary condition. The number of these sets, or *schemes*, has a (maximum) number `NDCS`

`IIBC`	Inner boundary condition.
1	None imposed
2	Function must vanish at boundary.
3	First derivative must vanish at boundary.
4	Both function and first derivative must vanish.
5	Both function and second derivative must vanish.
6	$rdf/dr - f(r) = 0$
7	$rdf/dr - lf(r) = 0$ with $l =$`L`.

Table 1: Current options for setting boundary conditions at the inner boundary: values of **IIBC**.

`IOBC`	Outer boundary condition.
1	None imposed
2	Function must vanish at boundary.
3	First derivative must vanish at boundary.
4	Both function and first derivative must vanish.
5	Both function and second derivative must vanish.
6	$rdf/dr - f(r) = 0$
7	$rdf/dr + (l+1)f(r) = 0$ with $l =$`L`.

Table 2: Current options for setting boundary conditions at the outer boundary: values of **IOBC**.

which must be declared at the outset. This is because the arrays of finite difference coefficients have `NDCS` as a dimension; not all of these schemes need be used in a calculation.

The properties of each of the schemes are stored in three integer arrays, `MHIBC`, `MHOBC` and `LARR`, each of dimension `NDCS`. For scheme `IS` with $1 \leq$ `IS` $\leq$ `NDCS`, `MHIBC( IS )` is the value of `IIBC` (see table 1) and `MHOBC( IS )` is the value of `IOBC` (see table 2). If `MHIBC( IS )` and `MHOBC( IS )` are set to 7, then the scheme denotes a poloidal magnetic field harmonic matching to a potential field. The spherical harmonic degrees, l, for those radial functions is stored in `LARR( IS )`. By setting `LARR( IS )` to `-1`, finite difference scheme `IS` is always ignored, and this is a way of dealing with finite difference schemes which are never referred to. For a spherical harmonic radial function number `IH`, which you will recall has its identity defined by the array elements `MHT( IH )`, `MHL( IH )` and `MHM( IH )`, the finite difference scheme is identified by the integer array element `MHP( IH )`.

The integer arrays `MHT`, `MHL` and `MHM`, along with the boundary conditions which the defined radial functions satisfy, are stored in a file with the suffix `.ints`.

The `.ints` file has a first line which indicates the number of harmonic radial functions, `NH`, stored in the corresponding `.vecs` file. The following `NH` lines contain five integer numbers; line number `IH+1` of the `.ints` file contains the numbers `MHT( IH )   MHL( IH )   MHM( IH )   IIBC   IOBC`.

Two examples are found in the directory

```
$LEOPACK_DIR/EXAMPLES/FUNDAMENTALS
```

The three files `case0.ints`, `case0.vecs` and `case0.xarr`, completely describe a solution to the non-magnetic (Case 0) benchmark from [CAC$^+$01]. The first line of the file `case0.ints` simply reads `503`. This is the number of spherical harmonic radial functions stored in the vector `case0.vecs`. This must be the same as the third number of the first line of `case0.vecs`. The time-stepping codes read in the `.ints` file first and if the first number in this file does not equal the third number in the first line of the `.vecs` file, the program knows that the `.ints` file and `.vecs` file cannot describe the same solution, and the program aborts with an error message. So we have `503` spherical harmonic radial functions. Line `2` of `case0.ints` reads

```
 1   2    0  4  4
```

This means that `MHT( 1 ) = 1`, `MHL( 1 ) = 2` and `MHM( 1 ) = 0`. We then know that the first of our radial functions corresponds to ${}^{P}v_2^{0c}(r)$ (c.f. Equation 32). We see also that `IIBC` and `IOBC` are both equal to `4`, and so we have the boundary conditions

$$ {}^{P}v_2^{0c}(r_{\rm i}) = {}^{P}v_2^{0c}(r_{\rm o}) = 0 \qquad (55)$$

and

$$\left.\frac{\partial[\ {}^{P}v_2^{0c}(r)\ ]}{\partial r}\right|_{r_{\rm i}} = \left.\frac{\partial[\ {}^{P}v_2^{0c}(r)\ ]}{\partial r}\right|_{r_{\rm o}} = 0 \qquad (56)$$

(c.f. tables 1 and 2). This means that the poloidal velocity, or atleast this particular spherical harmonic radial function, must satisfy rigid boundary conditions. Every line of `case0.ints` which begins with a number `1` indicates a poloidal velocity radial function. The values of `IIBC` and `IOBC` ought to be the same for each such function; i.e. if one line of `case0.ints` begins with `1` and ends with `4  4`, then every line of `case0.ints` which begins with `1` ought to end with `4  4`. Otherwise, we will have some poloidal velocity radial functions satisfying different boundary conditions to other poloidal velocity radial functions. This is never actually checked for in programs. The only safeguard against this is that if, when reading in a `.ints` file, the number of permitted finite difference schemes (`NDCS`) is exceeded then the program will abort with an error message. If line number `2` of `case0.ints` had read

```
 1    2     0   5   5
```

then we would be imposing the boundary conditions

$${}^{P}v_2^{0c}(r_{\rm i}) = {}^{P}v_2^{0c}(r_{\rm o}) = 0 \qquad (57)$$

and

$$\left.\frac{\partial^2[\ {}^{P}v_2^{0c}(r)\ ]}{\partial r^2}\right|_{r_{\rm i}} = \left.\frac{\partial^2[\ {}^{P}v_2^{0c}(r)\ ]}{\partial r^2}\right|_{r_{\rm o}} = 0. \qquad (58)$$

This would correspond to stress-free boundaries. Editting every line of `case0.ints` beginning with `1`, such that the `4  4` at the end of the line was changed to a `5  5`, would mean that when a time-stepping code, e.g. **o2ubtctsc2.exe**, next read in the three files `case0.ints`, `case0.vecs` and `case0.xarr`, the poloidal boundary conditions would be treated as stress-free. Line `92` of file `case0.ints` reads

```
 2    5    -4   2   2
```

which indicates that `MHT( 91 ) = 2`, `MHL( 91 ) = 5` and `MHM( 91 ) = -4`. We then know that radial function 91 corresponds to ${}^{T}v_5^{4s}(r)$ (c.f. Equation 33). The inner and outer boundary flags are both 2 which, by tables (1) and (2), means that this function vanishes at both boundaries. Similarly, all lines in the file `case0.ints` which begin with `2`, end with `2  2`. This is consistent with the rigid boundary condition

$${}^{T}v_5^{4s}(r_{\rm i}) = {}^{T}v_5^{4s}(r_{\rm o}) = 0. \qquad (59)$$

If every line of the file `case0.ints` which begins with `2` were to end in `6  6`, and every line of `case0.ints` beginning with `1` were to end in `5  5`, then a stress-free boundary condition would be applied to both the inner and outer boundaries. The last line of `case0.ints` reads

```
 3  36  -28  2  2
```

which means that `MHT( 503 ) = 3`, `MHL( 503 ) = 36` and `MHM( 503 ) = -28`. We then know that radial function 503 corresponds to $\Theta_{36}^{28s}(r)$ (c.f. Equation 34).

Question:

How would you convert the solution described by `case0.ints`, `case0.vecs` and `case0.xarr` to satisfy a fixed heat-flux at the outer boundary as opposed to a fixed temperature?

Answer:

For every line of the file `case0.ints` which begins with a number `3`, change the final two numbers from `2  2` to `2  3`. This makes the radial derivative of each Θ function zero at the outer boundary which, by Fourier's Law, means that there is no net heat-flux across the boundary contributed by the Θ functions. (The actual heat-flux at the boundary is then controlled by parameters b_1, b_2 and c_h as given in equations 3 and 15.) Ofcourse, we have not changed the actual solution vector, `case0.vecs`, in this operation and so for the change to actually be implemented, we will need to run the time-stepping code (for example **o2ubtctsc2.exe**) for a time for the solution vector to adjust. There might be some unpleasant numerical behaviour in a transition period (small timestep required), although I have yet to find a case where such a change simply failed to work.

Question:

What is the value of the radial function ${}^{T}v_{33}^{20c}(r)$ at a radius $r = 0.89579341$ in this solution?

Answer:

Firstly we need to find out the harmonic number, `IH`, corresponding to ${}^{T}v_{33}^{20c}(r)$ and the radial grid node number, `IR`, corresponding to $r = 0.89579341$. The definition of ${}^{T}v_{33}^{20c}(r)$ from Equation (33) tells us that `MHT( IH ) = 2` (toroidal velocity function), `MHL( IH ) = 33` (spherical harmonic degree, l) and `MHM( IH ) = 20` (since $m = 20$ and we have a $\cos m\phi$ dependence).

Examining the file `case0.ints` reveals that line number `415` reads

```
 2  33   20  2  2
```

and so we conclude that `IH = 414`.

Examining the first line of `case0.xarr` shows that we have `40` radial grid nodes which are stored in format `IFORM = 1` (i.e. `(5(1PD16.7))` format). There are therefore 5 numbers per line. $r = 0.89579341$ is found as the second number on line five - and hence the fourth line of data. It is therefore the radius at grid node `IR = (4-1)*5 + 2 = 17`.

We finally examine the file `case0.vecs` to find the value of radial function `414` at grid node `17`. The first line of `case0.vecs` reads

```
   4   40  503    1
```

which means that 503 radial functions (check against `case0.ints`), each represented at 40 grid nodes (check against `case0.xarr`) are stored in format`IFORM = 1` (i.e. `(5(1PD16.7))` format) with `INARR( 1 ) = 4`: see the integer function `INDFUN` and Figure (2). From `indfun.f`, the index in the solution vector for `IR = 17`, `NR = 40` and `IH = 414` is given by

```
INDEX = ( IH - 1 )*NR + IR
      = ( 414 - 1 )*40 + 17
      = 413*40 + 17
      = 16537
```

Our answer is therefore stored in the 16537th element of our vector. Now, performing an integer division of 16537 by 5 (the number of data per line) shows that

```
16537 = 3307*5 + 2
```

Since the first line of `case0.vecs` contains integer parameters, our value is found as the second element of line 3309 is `-2.2468647D-06`.

Examining more closely the file `case0.ints` shows that for poloidal velocity, toroidal velocity and temperature, the highest spherical harmonic degree, l, is 36. In terms of the nomenclature in equations (32), (33) and (34), this means that $L1 = L2 = L3 = 36$. In addition, there are no wavenumbers higher than $m = 28$ and so $M_{\max} = 28$. Finally, all wavenumbers, m, are multiples of 4 and so $m_0 = 4$.

Input and output of integers files:

A `.ints` file is written by a call to the subroutine `HMFWT`: details on usage are found in the source code (or see the file `DOC_appkeysubs.pdf`).

Reading in a `.ints` file can be done in one of two ways, depending upon whether or not the boundary conditions of the radial functions are important. (They are not, for example, if we are simply plotting figures or estimating kinetic energy integrals.)

Assuming that the boundary conditions matter, as they most often will(!!), we read the indices and boundary conditions with the subroutine `HMFRD`: details on usage are found in the source code (or see the file `DOC_appkeysubs.pdf`).

If boundary conditions are irrelevant it is probably simplest to call the routine `BIHFRD`.

5 Time-stepping schemes

Our solution vector consists of the functions $\boldsymbol{u}$, Θ and $\boldsymbol{B}$ in the form described in the previous sections. We may therefore use $(\boldsymbol{u}, \Theta, \boldsymbol{B})$ to denote our solution vector generally, and $(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)$ to denote the solution vector at time $t = t_i$.

We may rewrite our MHD equations (9), (15) and (23) respectively

$$c_a \frac{\partial \Theta}{\partial t} = c_d \nabla^2 \Theta + F_\Theta(\boldsymbol{u}, \Theta, \boldsymbol{B}) \tag{60}$$

$$c_e \frac{\partial \boldsymbol{\omega}}{\partial t} = c_i \nabla^2 \boldsymbol{\omega} + \boldsymbol{F}_{\boldsymbol{\omega}}(\boldsymbol{u}, \Theta, \boldsymbol{B}) \tag{61}$$

and

$$c_k \frac{\partial \boldsymbol{B}}{\partial t} = c_l \nabla^2 \boldsymbol{B} + \boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}, \Theta, \boldsymbol{B}), \tag{62}$$

where the forcing terms, F_Θ, $\boldsymbol{F}_{\boldsymbol{\omega}}$ and $\boldsymbol{F}_{\boldsymbol{B}}$ are given respectively by

$$F_\Theta(\boldsymbol{u}, \Theta, \boldsymbol{B}) = c_d \varepsilon \nabla^2 T_{\mathrm{a}} + b_1 u_r r + b_2 \frac{u_r}{r^2} - c_c \boldsymbol{u}.\nabla(\Theta + \varepsilon T_{\mathrm{a}}), \tag{63}$$

$$\begin{aligned} \boldsymbol{F}_{\boldsymbol{\omega}}(\boldsymbol{u}, \Theta, \boldsymbol{B}) = & -c_f \nabla \times (\boldsymbol{u}.\nabla \boldsymbol{u}) - c_g \nabla \times (\boldsymbol{k} \times \boldsymbol{u}) \\ & + c_h \nabla \times [(\Theta + \varepsilon T_{\mathrm{a}})\boldsymbol{r}] + c_j \nabla \times [(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}] \end{aligned} \tag{64}$$

and

$$\boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}, \Theta, \boldsymbol{B}) = c_m \nabla \times (\boldsymbol{u} \times \boldsymbol{B}). \tag{65}$$

Over one time-step, the time derivatives $\partial\Theta/\partial t$, $\partial\boldsymbol{\omega}/\partial t$ and $\partial\boldsymbol{B}/\partial t$, are approximated by

$$\frac{\partial \Theta}{\partial t} = c \left(\frac{\partial \Theta}{\partial t}\right)^i + (1 - c)\left(\frac{\partial \Theta}{\partial t}\right)^{i+1} \tag{66}$$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = c \left(\frac{\partial \boldsymbol{\omega}}{\partial t}\right)^i + (1 - c)\left(\frac{\partial \boldsymbol{\omega}}{\partial t}\right)^{i+1} \tag{67}$$

and

$$\frac{\partial \boldsymbol{B}}{\partial t} = c\left(\frac{\partial \boldsymbol{B}}{\partial t}\right)^{i} + (1-c)\left(\frac{\partial \boldsymbol{B}}{\partial t}\right)^{i+1} \tag{68}$$

where the superscript i refers to quantities at the time $t = t_i$ and the superscript $i+1$ refers to quantities at the time $t = t_i + \Delta t$. The factor c must be between 0 and 1: $c = 0.5$ implies a Crank-Nicolson scheme, $c < 0.5$ implies a slightly more implicit scheme and $c > 0.5$ implies a slightly more explicit scheme. In the inputs to the codes in this collection, c is generally referred to as `CFAC`. Reintroducing equations (60), (61) and (62) gives

$$\begin{aligned} c_a\frac{\partial\Theta}{\partial t} &= c\left(c_a\frac{\partial\Theta}{\partial t}\right)^{i} + (1-c)\left(c_a\frac{\partial\Theta}{\partial t}\right)^{i+1} \\ c_a\left(\frac{\Theta^{i+1}-\Theta^{i}}{\Delta t}\right) &= c\left[c_d\nabla^2\Theta^i + F_\Theta(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right] \\ &+ (1-c)\left[c_d\nabla^2\Theta^{i+1} + F_\Theta(\boldsymbol{u}^{i+1},\Theta^{i+1},\boldsymbol{B}^{i+1})\right], \end{aligned} \tag{69}$$

$$\begin{aligned} c_e\frac{\partial\boldsymbol{\omega}}{\partial t} &= c\left(c_e\frac{\partial\boldsymbol{\omega}}{\partial t}\right)^{i} + (1-c)\left(c_e\frac{\partial\boldsymbol{\omega}}{\partial t}\right)^{i+1} \\ c_e\left(\frac{\boldsymbol{\omega}^{i+1}-\boldsymbol{\omega}^{i}}{\Delta t}\right) &= c\left[c_i\nabla^2\boldsymbol{\omega}^i + \boldsymbol{F}_{\boldsymbol{\omega}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right] \\ &+ (1-c)\left[c_i\nabla^2\boldsymbol{\omega}^{i+1} + \boldsymbol{F}_{\boldsymbol{\omega}}(\boldsymbol{u}^{i+1},\Theta^{i+1},\boldsymbol{B}^{i+1})\right] \end{aligned} \tag{70}$$

and

$$\begin{aligned} c_k\frac{\partial\boldsymbol{B}}{\partial t} &= c\left(c_k\frac{\partial\boldsymbol{B}}{\partial t}\right)^{i} + (1-c)\left(c_k\frac{\partial\boldsymbol{B}}{\partial t}\right)^{i+1} \\ c_k\left(\frac{\boldsymbol{B}^{i+1}-\boldsymbol{B}^{i}}{\Delta t}\right) &= c\left[c_l\nabla^2\boldsymbol{B}^i + \boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right] \\ &+ (1-c)\left[c_l\nabla^2\boldsymbol{B}^{i+1} + \boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}^{i+1},\Theta^{i+1},\boldsymbol{B}^{i+1})\right]. \end{aligned} \tag{71}$$

Since the forcing terms, F_Θ, $\boldsymbol{F}_{\boldsymbol{\omega}}$ and $\boldsymbol{F}_{\boldsymbol{B}}$ involve non-linear terms, it is not possible to simply rearrange equations (69), (70) and (71) to make Θ^{i+1}, $\boldsymbol{u}^{i+1}$ and $\boldsymbol{B}^{i+1}$ the subjects which may be solved for. We therefore need an iterative scheme. A predictor, $(\boldsymbol{u}^{i+1}_{(0)},\Theta^{i+1}_{(0)},\boldsymbol{B}^{i+1}_{(0)})$, for $(\boldsymbol{u}^{i+1},\Theta^{i+1},\boldsymbol{B}^{i+1})$ is first formed by solving the equations

$$\begin{aligned} \left[c_a + \Delta t c_d(c-1)\nabla^2\right]\Theta^{i+1}_{(0)} &= \left[c_a + \Delta t c_d c\nabla^2\right]\Theta^i \\ &+ \Delta t . F_\Theta(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i) \\ \left[c_e + \Delta t c_i(c-1)\nabla^2\right]\nabla\times\boldsymbol{u}^{i+1}_{(0)} &= \left[c_e + \Delta t c_i c\nabla^2\right]\nabla\times\boldsymbol{u}^i \\ &+ \Delta t . \boldsymbol{F}_{\boldsymbol{w}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i) \\ \left[c_k + \Delta t c_l(c-1)\nabla^2\right]\boldsymbol{B}^{i+1}_{(0)} &= \left[c_k + \Delta t c_l c\nabla^2\right]\boldsymbol{B}^i \\ &+ \Delta t . \boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i). \end{aligned} \tag{72}$$

Once we have our initial guess, we then iterate towards our solution, $(\boldsymbol{u}^{i+1}, \Theta^{i+1}, \boldsymbol{B}^{i+1})$, using

$$\begin{array}{rcl} \left[c_a + \Delta t c_d (c-1)\nabla^2\right] \Theta^{i+1}_{(j)} & = & \left[c_a + \Delta t c_d c \nabla^2\right] \Theta^i \\ & + & c\Delta t. \boldsymbol{F}_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i) \\ & + & (1-c)\Delta t. \boldsymbol{F}_\Theta(\boldsymbol{u}^{i+1}_{(j-1)}, \Theta^{i+1}_{(j-1)}, \boldsymbol{B}^{i+1}_{(j-1)}) \\ & & \\ \left[c_e + \Delta t c_i (c-1)\nabla^2\right] \nabla \times \boldsymbol{u}^{i+1}_{(j)} & = & \left[c_e + \Delta t c_i c \nabla^2\right] \nabla \times \boldsymbol{u}^i \\ & + & c\Delta t. \boldsymbol{F}_{\boldsymbol{\omega}}(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i) \\ & + & (1-c)\Delta t. \boldsymbol{F}_{\boldsymbol{\omega}}(\boldsymbol{u}^{i+1}_{(j-1)}, \Theta^{i+1}_{(j-1)}, \boldsymbol{B}^{i+1}_{(j-1)}) \\ & & \\ \left[c_k + \Delta t c_l (c-1)\nabla^2\right] \boldsymbol{B}^{i+1}_{(j)} & = & \left[c_k + \Delta t c_l c \nabla^2\right] \boldsymbol{B}^i \\ & + & c\Delta t. \boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i) \\ & + & (1-c)\Delta t. \boldsymbol{F}_{\boldsymbol{B}}(\boldsymbol{u}^{i+1}_{(j-1)}, \Theta^{i+1}_{(j-1)}, \boldsymbol{B}^{i+1}_{(j-1)}), \end{array} \tag{73}$$

terminating when we are satisfied that the j^{th} iteration, $(\boldsymbol{u}^{i+1}_{(j)}, \Theta^{i+1}_{(j)}, \boldsymbol{B}^{i+1}_{(j)})$, is a close enough approximation. The maximum number of iterations allowed in the time stepping codes is generally called `ITMX`. The iteration terminates when the numerical norm between successive iterations,

$$N_j = |(\boldsymbol{u}^{i+1}_{(j)}, \Theta^{i+1}_{(j)}, \boldsymbol{B}^{i+1}_{(j)}) - (\boldsymbol{u}^{i+1}_{(j-1)}, \Theta^{i+1}_{(j-1)}, \boldsymbol{B}^{i+1}_{(j-1)})| \tag{74}$$

is less than a given tolerance, usually specified in the time-stepping codes as `DTOL`. It is best to ensure that as few iterations as possible are required: $j = 1$ being the ideal. If more than 1 iteration is required per time-step then it is probably more effective to use a smaller, cheaper time-step than to labour over the evaluation of many iterations for a larger time-step.

All the current time-stepping codes use a fixed time-step. A cleverer kind of code will judge from the changes in elements of the solution vector or forcing terms whether a new size for time-step is necessary. Applying this is quite a high priority if we are to look at anything but the simplest of time-dependences.

From our formalism in previous sections, we may define our solution vector, $(\boldsymbol{u}, \Theta, \boldsymbol{B})$, in terms of the five scalar quantities; ${}^Pv$, ${}^Tv$, Θ, ${}^PB$ and ${}^TB$. (See equations 18 and 19 for poloidal and toroidal scalar notation.) These functions can each be expanded in terms of spherical harmonics:

$${}^Pv(r,t,\theta,\phi) = \sum_\alpha \left[{}^Pv\right]_\alpha (r,t)\; Y_\alpha(\theta,\phi) \tag{75}$$

$${}^Tv(r,t,\theta,\phi) = \sum_\alpha \left[{}^Tv\right]_\alpha (r,t)\; Y_\alpha(\theta,\phi) \tag{76}$$

$$\Theta(r,t,\theta,\phi) = \sum_\alpha \left[\Theta\right]_\alpha (r,t)\; Y_\alpha(\theta,\phi) \tag{77}$$

$${}^PB(r,t,\theta,\phi) = \sum_\alpha \left[{}^PB\right]_\alpha (r,t)\; Y_\alpha(\theta,\phi) \tag{78}$$

$$ {}^{T}B(r,t,\theta,\phi) \;=\; \sum_{\alpha} \left[{}^{T}B\right]_{\alpha}(r,t)\; Y_{\alpha}(\theta,\phi) \tag{79}$$

c.f. equations (32), (33), (34), (35) and (36) with the spherical harmonic, $Y_\alpha(\theta,\phi)$ defined by

$$ Y_\alpha(\theta,\phi) = P_{l_\alpha}^{m_\alpha}(\cos\theta)\{{}^{cos}_{sin}\}_\alpha m_\alpha\phi, \tag{80}$$

(see the file `DOC_sphervec.pdf` for details). The forcing term $F_\Theta(\boldsymbol{u},\Theta,\boldsymbol{B})$ is a scalar and can also be expanded in spherical harmonics with

$$ F_\Theta(r,t,\theta,\phi) = \sum_\alpha \left[F_\Theta\right]_\alpha(r,t) Y_\alpha(\theta,\phi). \tag{81}$$

It is clear from the definitions of the vector forcing terms $\boldsymbol{F}_{\boldsymbol{\omega}}$ and $\boldsymbol{F}_{\boldsymbol{B}}$ that they are divergence free and can therefore also be expressed in poloidal and toroidal decompositions:

$$ \boldsymbol{F}_{\boldsymbol{\omega}} = \nabla\times\nabla\times\left[\,{}^{P}F_{\boldsymbol{\omega}}(r,t,\theta,\phi)\;\boldsymbol{r}\,\right] \;+\; \nabla\times\left[\,{}^{T}F_{\boldsymbol{\omega}}(r,t,\theta,\phi)\;\boldsymbol{r}\,\right] \tag{82}$$

and

$$ \boldsymbol{F}_{\boldsymbol{B}} = \nabla\times\nabla\times\left[\,{}^{P}F_{\boldsymbol{B}}(r,t,\theta,\phi)\;\boldsymbol{r}\,\right] \;+\; \nabla\times\left[\,{}^{T}F_{\boldsymbol{B}}(r,t,\theta,\phi)\;\boldsymbol{r}\,\right]. \tag{83}$$

The four scalars are all given the same kinds of expansions:

$$ {}^{P}F_{\boldsymbol{\omega}}(r,t,\theta,\phi) \;=\; \sum_\alpha \left[{}^{P}F_{\boldsymbol{\omega}}\right]_\alpha(r,t)Y_\alpha(\theta,\phi) \tag{84}$$

$$ {}^{T}F_{\boldsymbol{\omega}}(r,t,\theta,\phi) \;=\; \sum_\alpha \left[{}^{T}F_{\boldsymbol{\omega}}\right]_\alpha(r,t)Y_\alpha(\theta,\phi) \tag{85}$$

$$ {}^{P}F_{\boldsymbol{B}}(r,t,\theta,\phi) \;=\; \sum_\alpha \left[{}^{P}F_{\boldsymbol{B}}\right]_\alpha(r,t)Y_\alpha(\theta,\phi) \tag{86}$$

$$ {}^{T}F_{\boldsymbol{B}}(r,t,\theta,\phi) \;=\; \sum_\alpha \left[{}^{T}F_{\boldsymbol{B}}\right]_\alpha(r,t)Y_\alpha(\theta,\phi). \tag{87}$$

We introduce the notation

$$ \left[{}^{P}v^{i}\right]_\alpha(r) = \left[{}^{P}v\right]_\alpha(r,t_i), \tag{88}$$

and $\left[{}^{P}v^{i+1}_{(j)}\right]_\alpha(r)$ for the $j^{\rm th}$ iteration towards the value $\left[{}^{P}v\right]_\alpha(r,t_i+\Delta t)$. The same convention is applied to ${}^{T}v$, Θ, ${}^{P}B$, ${}^{T}B$, ${}^{P}F_{\boldsymbol{\omega}}$, ${}^{T}F_{\boldsymbol{\omega}}$, F_Θ, ${}^{P}F_{\boldsymbol{B}}$ and ${}^{T}F_{\boldsymbol{B}}$. Now we know (see file `DOC_sphervec.pdf`) that

$$ \nabla^2\sum_\alpha \Theta_\alpha(r,t)Y_\alpha = \sum_\alpha \mathcal{D}_{l_\alpha}\Theta_\alpha(r,t)Y_\alpha, \tag{89}$$

the $\mathcal{D}_l$ operator being defined by

$$\begin{aligned}\mathcal{D}_l f &= \frac{1}{r}\frac{d^2}{dr^2}(rf) - \frac{l(l+1)}{r^2}f & (90)\\ &= \frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{df}{dr}\right) - \frac{l(l+1)}{r^2}f & (91)\\ &= \frac{d^2f}{dr^2} + \frac{2}{r}\frac{df}{dr} - \frac{l(l+1)}{r^2}f. & (92)\end{aligned}$$

In other words, the ∇^2 operator does not couple distinct spherical harmonic radial functions. We may therefore consider the equation

$$\begin{aligned}\left[c_a + \Delta t c_d (c-1)\nabla^2\right]\Theta^{i+1}_{(0)} &= \quad \left[c_a + \Delta t c_d c \nabla^2\right]\Theta^i \\ &+ \quad \Delta t . F_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)\end{aligned} \qquad (93)$$

separately for each spherical harmonic radial function, α:

$$\begin{aligned}\left[c_a + \Delta t c_d (c-1)\nabla^2\right]\left[\Theta^{i+1}_{(0)}\right]_\alpha(r) &= \quad \left[c_a + \Delta t c_d c \nabla^2\right]\left[\Theta^i\right]_\alpha(r) \\ &+ \quad \Delta t . \left[F_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)\right]_\alpha(r),\end{aligned} \qquad (94)$$

where $\left[F_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)\right]_\alpha(r)$ has been evaluated using the spherical scalar transform from the non-linear and inhomogeneous terms in the heat equation. Equation (94) can further be written

$$\begin{aligned}\left[c_a + \Delta t c_d (c-1)\mathcal{D}_{l_\alpha}\right]\left[\Theta^{i+1}_{(0)}\right]_\alpha(r) &= \quad \left[c_a + \Delta t c_d c \mathcal{D}_{l_\alpha}\right]\left[\Theta^i\right]_\alpha(r) \\ &+ \quad \Delta t . \left[F_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)\right]_\alpha(r).\end{aligned} \qquad (95)$$

The terms $\left[c_a + \Delta t c_d (c-1)\mathcal{D}_{l_\alpha}\right]$ and $\left[c_a + \Delta t c_d c \mathcal{D}_{l_\alpha}\right]$ act as linear operators on the $\Theta_\alpha(r)$ radial functions. In our finite difference discretisation, Equation (95) becomes the matrix equation

$$\boldsymbol{A}_{3,l_\alpha}\left[\Theta^{i+1}_{(0)}\right]_\alpha(r) = \boldsymbol{B}_{3,l_\alpha}\left[\Theta^i\right]_\alpha(r) + \Delta t . \left[F_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)\right]_\alpha(r) \qquad (96)$$

where the matrices $\boldsymbol{A}_{3,l_\alpha}$ and $\boldsymbol{B}_{3,l_\alpha}$ are defined by

$$\boldsymbol{A}_{3,l} = \left[c_a + \Delta t c_d (c-1)\mathcal{D}_l\right] \qquad (97)$$

and

$$\boldsymbol{B}_{3,l} = \left[c_a + \Delta t c_d c \mathcal{D}_l\right]. \qquad (98)$$

Here, $\left[\Theta^{i+1}_{(0)}\right]_\alpha(r)$, $\left[\Theta^i\right]_\alpha(r)$ and $\left[F_\Theta(\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i)\right]_\alpha(r)$ are all vectors of length N_r, the number of radial grid nodes. Due to the short range finite difference interactions, the `NR*NR` matrices $\boldsymbol{A}_{3,l_\alpha}$ and $\boldsymbol{B}_{3,l_\alpha}$ are banded with a band width of `2*NBN + 1`. Since each spherical harmonic is decoupled at the linear algebra stage, we may simply construct a single solution vector for all of the NH3 temperature radial functions; this is a double precision vector of length NR*NH3 with the elements arranged in the IFORMF = 4 format as shown in Figure (2 b). This arrangement allows for all of the NH3 matrices $\boldsymbol{A}_{3,l_\alpha}$ to be stored in a banded matrix with length NR*NH3 and band width 2*NBN + 1. The $\boldsymbol{B}_{3,l_\alpha}$ matrices can be stored similarly. The arrays which stores all the matrices $\boldsymbol{A}_{3,l_\alpha}$ and $\boldsymbol{B}_{3,l_\alpha}$ are respectively termed AM3 and BM3 in the code and are both generated as a function of parameters CA, CD, CFAC and DELTAT by the subroutine TMTSMF.

The $\boldsymbol{A}_{3,l_\alpha}$ and $\boldsymbol{B}_{3,l_\alpha}$ matrices are precalculated before the time-step procedure is initiated. Aswell as forming the predictor (Equation 96), the same arrays AM3 and BM3 are also used to correct the solution vector by the iterative equation

$$\begin{aligned}\boldsymbol{A}_{3,l_\alpha}\left[\Theta^{i+1}_{(j)}\right]_\alpha(r) &= \boldsymbol{B}_{3,l_\alpha}\left[\Theta^{i}\right]_\alpha(r) + c.\Delta t.\left[F_\Theta(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r)\\ &+ (1-c).\Delta t.\left[F_\Theta(\boldsymbol{u}^{i+1}_{(j-1)},\Theta^{i+1}_{(j-1)},\boldsymbol{B}^{i+1}_{(j-1)})\right]_\alpha(r)\end{aligned} \tag{99}$$

We know that

$$\nabla^2\left[\nabla\times\nabla\times\left(\left[{}^PB\right]_\alpha(r)Y_\alpha\boldsymbol{r}\right)\right] = \nabla\times\nabla\times\left(\mathcal{D}_{l_\alpha}\left[{}^PB\right]_\alpha(r)Y_\alpha\boldsymbol{r}\right) \tag{100}$$

and

$$\nabla^2\left[\nabla\times\left(\left[{}^TB\right]_\alpha(r)Y_\alpha\boldsymbol{r}\right)\right] = \nabla\times\left(\mathcal{D}_{l_\alpha}\left[{}^TB\right]_\alpha(r)Y_\alpha\boldsymbol{r}\right) \tag{101}$$

(see file DOC_sphervec.pdf). Applying these results to Equation (72), the predictors for the poloidal and toroidal magnetic field scalars can then be written

$$\boldsymbol{A}_{4,l_\alpha}\left[{}^PB^{i+1}_{(0)}\right]_\alpha(r) = \boldsymbol{B}_{4,l_\alpha}\left[{}^PB^{i}\right]_\alpha(r) + \Delta t.\left[{}^PF_{\boldsymbol{B}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r) \tag{102}$$

and

$$\boldsymbol{A}_{5,l_\alpha}\left[{}^TB^{i+1}_{(0)}\right]_\alpha(r) = \boldsymbol{B}_{5,l_\alpha}\left[{}^TB^{i}\right]_\alpha(r) + \Delta t.\left[{}^TF_{\boldsymbol{B}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r). \tag{103}$$

The matrices $\boldsymbol{A}_{4,l_\alpha}$ and $\boldsymbol{B}_{4,l_\alpha}$ are defined by

$$\boldsymbol{A}_{4,l} = [c_k + \Delta t c_l(c-1)\mathcal{D}_l] \tag{104}$$

and

$$\boldsymbol{B}_{4,l} = [c_k + \Delta t c_l c\mathcal{D}_l] \tag{105}$$

and are returned in the double precision arrays AM4 and BM4 by the subroutine PFTSMF. Similarly, the matrices $\boldsymbol{A}_{5,l_\alpha}$ and $\boldsymbol{B}_{5,l_\alpha}$ are defined by

$$\boldsymbol{A}_{5,l} = [c_k + \Delta t c_l(c-1)\mathcal{D}_l] \tag{106}$$

and

$$\boldsymbol{B}_{5,l} = [c_k + \Delta t c_l c\mathcal{D}_l] \tag{107}$$

and are returned in the double precision arrays `AM5` and `BM5` by the subroutine `TFTSMF`. These arrays are also used for the iterative correction of the poloidal and toroidal magnetic field scalars:

$$\begin{aligned}\boldsymbol{A}_{4,l_\alpha}\left[{}^P B^{i+1}_{(j)}\right]_\alpha(r) &= \boldsymbol{B}_{4,l_\alpha}\left[{}^P B^{i}\right]_\alpha(r) + c.\Delta t.\left[{}^P F_{\boldsymbol{B}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r) \\ &+ (1-c).\Delta t.\left[{}^P F_{\boldsymbol{B}}(\boldsymbol{u}^{i+1}_{(j-1)},\Theta^{i+1}_{(j-1)},\boldsymbol{B}^{i+1}_{(j-1)})\right]_\alpha(r)\end{aligned} \tag{108}$$

and

$$\begin{aligned}\boldsymbol{A}_{5,l_\alpha}\left[{}^T B^{i+1}_{(j)}\right]_\alpha(r) &= \boldsymbol{B}_{5,l_\alpha}\left[{}^T B^{i}\right]_\alpha(r) + c.\Delta t.\left[{}^T F_{\boldsymbol{B}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r) \\ &+ (1-c).\Delta t.\left[{}^T F_{\boldsymbol{B}}(\boldsymbol{u}^{i+1}_{(j-1)},\Theta^{i+1}_{(j-1)},\boldsymbol{B}^{i+1}_{(j-1)})\right]_\alpha(r).\end{aligned} \tag{109}$$

From Equations (33) in the file `DOC_sphervec.pdf` the curl of a poloidal velocity vector is a toroidal vector:

$$\nabla\times\left[\ \nabla\times\nabla\times\left(\ \left[{}^P v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right)\ \right] = \nabla\times\left(\ -\mathcal{D}_{l_\alpha}\left[{}^P v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right) \tag{110}$$

and the curl of the toroidal velocity vector is (trivially!) a poloidal vector:

$$\nabla\times\left[\ \nabla\times\left(\ \left[{}^T v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right)\ \right] = \nabla\times\nabla\times\left(\ \left[{}^T v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right). \tag{111}$$

Applying equations (54) and (55) from the file `DOC_sphervec.pdf` reveals that

$$\nabla\times\nabla^2\left[\nabla\times\nabla\times\left(\ \left[{}^P v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right)\right] = \nabla\times\left(\ -\mathcal{D}^2_{l_\alpha}\left[{}^P v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right) \tag{112}$$

and

$$\nabla\times\nabla^2\left[\ \nabla\times\left(\ \left[{}^T v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right)\ \right] = \nabla\times\nabla\times\left(\ \mathcal{D}_{l_\alpha}\left[{}^T v\right]_\alpha(r)Y_\alpha\boldsymbol{r}\ \right), \tag{113}$$

where the $\mathcal{D}^2_l$ operator is defined by

$$\mathcal{D}^2_l f = f'''' + 4\frac{f'''}{r} - 2l(l+1)\frac{f''}{r^2} + (l+2)(l+1)l(l-1)\frac{f}{r^4}, \tag{114}$$

where $'$ denotes differentiation with respect to r.

Consequently, we form our velocity scalar function predictors using

$$\boldsymbol{A}_{1,l_\alpha}\left[{}^P v^{i+1}_{(0)}\right]_\alpha(r) = \boldsymbol{B}_{1,l_\alpha}\left[{}^P v^{i}\right]_\alpha(r) + \Delta t.\left[{}^T F_{\boldsymbol{\omega}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r) \tag{115}$$

and

$$\boldsymbol{A}_{2,l_\alpha}\left[{}^T v^{i+1}_{(0)}\right]_\alpha(r) = \boldsymbol{B}_{2,l_\alpha}\left[{}^T v^{i}\right]_\alpha(r) + \Delta t.\left[{}^P F_{\boldsymbol{\omega}}(\boldsymbol{u}^i,\Theta^i,\boldsymbol{B}^i)\right]_\alpha(r). \tag{116}$$

The matrices $\boldsymbol{A}_{1,l_\alpha}$ and $\boldsymbol{B}_{1,l_\alpha}$ are defined by

$$\boldsymbol{A}_{1,l} = \left[-c_e\mathcal{D}_l - \Delta t c_i(c-1)\mathcal{D}^2_l\right] \tag{117}$$

and

$$\boldsymbol{B}_{1,l} = \left[-c_e \mathcal{D}_l - \Delta t c_i c \mathcal{D}_l^2 \right] \tag{118}$$

and are returned in the double precision arrays `AM1` and `BM1` by the subroutine `PVTSMF`. Similarly, the matrices $\boldsymbol{A}_{2,l_\alpha}$ and $\boldsymbol{B}_{2,l_\alpha}$ are defined by

$$\boldsymbol{A}_{2,l} = [c_e + \Delta t c_i (c-1) \mathcal{D}_l] \tag{119}$$

and

$$\boldsymbol{B}_{2,l} = [c_e + \Delta t c_i c \mathcal{D}_l] \tag{120}$$

and are returned in the double precision arrays `AM2` and `BM2` by the subroutine `TVTSMF`. These arrays are also used for the iterative correction of the poloidal and toroidal magnetic field scalars:

$$\begin{array}{rcl} \boldsymbol{A}_{1,l_\alpha} \left[{}^P v_{(j)}^{i+1} \right]_\alpha (r) & = & \boldsymbol{B}_{1,l_\alpha} \left[{}^P v^i \right]_\alpha (r) + c.\Delta t. \left[{}^T F_{\boldsymbol{\omega}} (\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i) \right]_\alpha (r) \\ & + & (1-c).\Delta t. \left[{}^T F_{\boldsymbol{\omega}} (\boldsymbol{u}_{(j-1)}^{i+1}, \Theta_{(j-1)}^{i+1}, \boldsymbol{B}_{(j-1)}^{i+1}) \right]_\alpha (r) \end{array} \tag{121}$$

and

$$\begin{array}{rcl} \boldsymbol{A}_{2,l_\alpha} \left[{}^T v_{(j)}^{i+1} \right]_\alpha (r) & = & \boldsymbol{B}_{2,l_\alpha} \left[{}^T v^i \right]_\alpha (r) + c.\Delta t. \left[{}^P F_{\boldsymbol{\omega}} (\boldsymbol{u}^i, \Theta^i, \boldsymbol{B}^i) \right]_\alpha (r) \\ & + & (1-c).\Delta t. \left[{}^P F_{\boldsymbol{\omega}} (\boldsymbol{u}_{(j-1)}^{i+1}, \Theta_{(j-1)}^{i+1}, \boldsymbol{B}_{(j-1)}^{i+1}) \right]_\alpha (r). \end{array} \tag{122}$$

$$(a)\quad \begin{bmatrix} f_1(r_1) \\ f_2(r_1) \\ \vdots \\ f_j(r_1) \\ \vdots \\ f_{N_h}(r_1) \\ \cdots \\ f_1(r_2) \\ f_2(r_2) \\ \vdots \\ f_j(r_2) \\ \vdots \\ f_{N_h}(r_2) \\ \cdots \\ \\ \vdots \\ \\ \cdots \\ f_1(r_i) \\ f_2(r_i) \\ \vdots \\ f_j(r_i) \\ \vdots \\ f_{N_h}(r_i) \\ \cdots \\ \\ \vdots \\ \\ \cdots \\ f_1(r_{N_r}) \\ f_2(r_{N_r}) \\ \vdots \\ f_j(r_{N_r}) \\ \vdots \\ f_{N_h}(r_{N_r}) \end{bmatrix} \qquad (b)\quad \begin{bmatrix} f_1(r_1) \\ f_1(r_2) \\ \vdots \\ f_1(r_i) \\ \vdots \\ f_1(r_{N_r}) \\ \cdots \\ f_2(r_1) \\ f_2(r_2) \\ \vdots \\ f_2(r_i) \\ \vdots \\ f_2(r_{N_r}) \\ \cdots \\ \\ \vdots \\ \\ \cdots \\ f_j(r_1) \\ f_j(r_2) \\ \vdots \\ f_j(r_i) \\ \vdots \\ f_j(r_{N_r}) \\ \cdots \\ \\ \vdots \\ \\ \cdots \\ f_{N_h}(r_1) \\ f_{N_h}(r_2) \\ \vdots \\ f_{N_h}(r_i) \\ \vdots \\ f_{N_h}(r_{N_r}) \end{bmatrix}$$

Figure 2: Alternative arrangements of elements within the solution vector. (a) shows **IFORMF = 3** and (b) shows **IFORMF = 4**.

References

[And89] D. L. Anderson. *Theory of the Earth.* Blackwell Scientific Press, Oxford., 1989.

[BG54] E. C. Bullard and H. Gellman. Homogeneous dynamos and terrestrial magnetism. *Phil. Trans. R. Soc. Lond. A*, 247:213–278, 1954.

[CAC$^+$01] U. R. Christensen, J. Aubert, P. Cardin, E. Dormy, S. Gibbons, G. A. Glatzmaier, E. Grote, Y. Honkura, C. Jones, M. Kono, M. Matsushima, A. Sakuraba, F. Takahashi, A. Tilgner, J. Wicht, and K. Zhang. A numerical dynamo benchmark. *Phys. Earth Planet. Inter.*, 128:25–34, 2001.

[DA81] A. M. Dziewonski and D. L. Anderson. Preliminary reference earth model. *Phys. Earth Planet. Inter.*, 25:297–356, 1981.

[Dor97] E. Dormy. *Modeélisation Numérique de la Dynamo Terrestre.* PhD thesis, l'Institute de Physique du Globe de Paris, 1997.

[GR87] D. Gubbins and P. H. Roberts. Magnetohydrodynamics of the earth's core. In J. A. Jacobs, editor, *Geomagnetism Volume* II, pages 1–183. Academic Press, 1987.

[GZ93] D. Gubbins and K. Zhang. Symmetry properties of the dynamo equations for paleomagnetism and geomagnetism. *Phys. Earth Planet. Inter.*, 75:225–241, 1993.

[SJ99] G. R. Sarson and C. A. Jones. A convection driven geodynamo reversal model. *Phys. Earth Planet. Inter.*, 111:3–20, 1999.