RAS Techniques and Instruments

RASTAI **2,** 637–648 (2023) https://doi.org/10.1093/rasti/rzad043
Advance Access publication 2023 September 1

A set of codes for numerical convection and geodynamo calculations

Steven J. Gibbons ,[1]★ Ashley P. Willis,[2] Chris Davies[3] and David Gubbins[3]

[1]*Norwegian Geotechnical Institute, Postboks 3930 Ullevål Stadion, 0484 Oslo, Norway*
[2]*School of Mathematics and Statistics, University of Sheffield, Hounsfield Road, Sheffield S3 7RH, UK*
[3]*School of Earth and Environment, University of Leeds, Leeds LS2 9JT, UK*

Accepted 2023 August 30. Received 2023 August 25; in original form 2023 July 3

ABSTRACT

We present a set of codes for calculating and displaying solutions to diverse problems within thermal convection and magnetic field generation in rotating fluid-filled spheres and spherical shells. There are diverse programs for the kinematic dynamo problem, the onset of thermal convection, and boundary-locked thermal convection, and time-stepping codes for non-magnetic convection and the dynamo with either homogeneous or spatially varying thermal boundary conditions. Where possible, all programs have been benchmarked against other codes and tested by reproducing previously published results. Each program comes with the complete source code, a pdf instruction manual, and at least one example run with a sample input file and all necessary files for describing an initial condition. The only prerequisite for running most of the codes is a FORTRAN compiler. The plotting programs require in addition the PGPLOT graphics library. All source code, examples, input files, solutions, and instructions are available for download from github and Zenodo.

Key words: Software – Numerical Methods – Dynamo theory – Convection – Core–mantle coupling – Earth's magnetic field.

1 INTRODUCTION

The processes in Earth's outer core that generate the geomagnetic field are best understood through numerical simulations. Over a quarter of a century has now passed since the first fully self-consistent simulations of the geodynamo in which the induction equation, the heat equation, and the Navier Stokes equation are solved simultaneously in rotating spherical geometry (e.g. Glatzmaier & Roberts 1995; Christensen, Olson & Glatzmaier 1998; Kono & Roberts 2002). Around this time, processing capabilities had improved to the extent that it was possible for the first time to run a complete simulation on very modest computational resources and ensemble calculations could be performed to investigate the effect of changing parameters (e.g. Christensen, Olson & Glatzmaier 1999). The Benchmark Dynamo project (Christensen et al. 2001) provided a first baseline dynamo model by which new codes could be validated. Subsequent studies (e.g. Jones et al. 2011; Jackson et al. 2014; Marti et al. 2014) have provided benchmark solutions for dynamo simulations for a broader range of models.

Many codes have since been developed that have simulated the processes at ever higher spatial and temporal resolution (e.g. Matsui et al. 2016) with an increasing focus on high performance computing. Two geodynamo codes, XSHELLS (Schaeffer et al. 2017) and PARODY-PDAF (Fournier, Nerger & Aubert 2013), were optimized further towards deployment on the next generation of supercomputers within the EU-funded ChEESE Center of Excellence (Folch et al. 2023). The challenge is approaching parameters relevant to Earth's core, for which the decreasing temporal and spatial scales make the computations increasingly expensive and demanding. Aubert (2023) presents recent progress in geodynamo modelling towards Earth's parameter regime. Like many previous studies, Aubert (2023) mitigates the numerical difficulties by applying a hyperdiffusive treatment to the smallest scales.

Since the full geodynamo problem couples all the governing equations, is strongly time-dependent, and acts at a cascade of different spatial and temporal scales, the physical mechanisms that result in features such as geomagnetic reversals and core–mantle coupling can be difficult to isolate among the full spectrum of processes occurring in a given simulation. Causality can often only be suggested in a statistical sense over long durations of numerical simulations. It can therefore be beneficial to understanding cause and effect by isolating components of the full geodynamo problem and performing extensive sensitivity studies on the more limited systems. The kinematic dynamo problem, for instance, explores the magnetic field generating properties of a given fluid flow. Gubbins & Sarson (1994), for example, demonstrated how virtual geomagnetic poles during a reversal in a kinematic model followed paths concentrated on longitudes where the magnetic flux was concentrated. Calculations in the absence of a magnetic field can help us to understand the force balance in the outer core (e.g. Gastine, Wicht & Aubert 2016; Long et al. 2020) and provide clues as to how the geodynamo will be influenced by a spatially varying heat-flux at the outer boundary. Gubbins & Gibbons (2004), for example, demonstrated how a heat-flux pattern at the core–mantle boundary (CMB) with anomalously low heat-flux below both Africa and the Pacific suppressed columnar convection only below the Pacific, due to the length scales of the heat-flux anomalies. The control of the CMB heat-flux has since been investigated extensively also on full dynamo simulations (e.g. Mound & Davies 2023). Properties observed in such more limited systems may of course not

★ E-mail: steven.gibbons@ngi.no

apply to the full magnetohydrodynamic system. However, a complete understanding of the parameter space defining a geodynamo simulation is necessary. For example, would thermal convection take place in the absence of a magnetic field? How would the length scale of convection change were the magnetic field to collapse?

This paper presents a set of codes that address a number of topics related to the geodynamo. The flagship codes were designed to perform complete geodynamo simulations, with special attention paid to the influence of a laterally varying heat-flux at the CMB. At the same time, it was desired that there should be codes utilizing the same basis of routines and the same numerical formulation that addressed a number of the simpler systems. The language of choice was Fortran 77. This was partly due to the fact that the legacy codes at the University of Leeds for the kinematic dynamo were FORTRAN-based and partly in order to exploit most easily the BLAS and LAPACK libraries and the then-recently released ARPACK code. While not exploiting an object-oriented language, a special effort was made to modularize the code into far smaller units than had been used previously. The common blocks which had characterized earlier codes were also avoided. On the one hand, the new code contained a vast number of functions and subroutines: many with exceptionally long argument lists. On the other hand, these routines could be unit-tested far more easily than before and it was far easier to generate a new code from these building blocks. Although Fortran 77 seems like an illogical choice in 2023, with more modern alternatives such as Python and Julia available, the presented codes compile as readily now as they did 20 yr ago with freely available compilers.

All codes address the problems of convection and/or magnetic field generation in fluid-filled rotating spheres and spherical shells. The problems fall into five broad categories:

(i) The kinematic dynamo (5 codes),
(ii) The onset of thermal convection (4 codes),
(iii) Boundary locked (steady-state) convection (3 codes),
(iv) The time-dependent non-magnetic convection problem, with or without spatially varying heat-flux at the boundaries (2 codes), and
(v) The time-dependent dynamo problem, with or without spatially varying heat-flux at the boundaries (5 codes).

The Boussinesq approximation applies to all models. An additional 10 codes generate and manipulate the files defining the temperature, flow, and magnetic field variables. For example, if we want to change the spatial resolution of a calculation, there are programs that interpolate the solutions from one spatial specification to another. All of these 29 codes, described briefly in Section 3, are written in near-standard Fortran 77 and compile readily using, for example, the gfortran compiler.

An additional set of 10 programs, utilizing the PGPLOT plotting library (Pearson 2011), generate postscript plots of the specified fields with, optionally, arrows to indicate the direction and strength of the fluid flow. The postscript files generated are vector graphics that are readily converted to other formats, such as pdf and png. There are both free and commercial tools that will perform this conversion and we have found that the free *psconvert* program of GMT will perform the job well (see Data Availability statement). Two programs plot on surfaces of constant radius, four programs plot meridian or equatorial sections, or sections of constant distance from the equatorial plane, and four programs plot spherical projections. These codes are discussed separately in Section 4.

We note of course that other codes are openly available to the community. XSHELLS is openly available as is the SINGE code (Vidal & Schaeffer 2015; Monville et al. 2019). The links to both of these codes are provided in the Data Availability section. Our aim with this paper is to explain the scope and limitations of the programs presented here such that a user can determine whether or not a given program is relevant to an application. The codes were used to generate results for a sequence of publications (e.g. Gibbons & Gubbins 2000; Gubbins et al. 2000a, b; Christensen et al. 2001; Gubbins & Gibbons 2002, 2004; Gibbons, Gubbins & Zhang 2007; Gubbins & Gibbons 2009) at a time when it was less usual than it is today to include or publish source code. This paper rectifies this retrospectively. The descriptions of the individual codes explain which codes were used for which studies.

Each of the programs has its own directory in the github/Zenodo distribution with a pdf instruction manual, the full source code, a Makefile, at least one input file for an example run, and any state files required as initial conditions. In addition, every subroutine has extensive comments in the source code, describing in detail the input and output parameters. This should make it far easier to recycle the code to form new applications. In this paper, we start by presenting the equations that are solved and provide a brief overview of how the fields are stored (Section 2). In Section 3, we present briefly each of the main programs and outline their applicability and the publications in which they have featured. Readers wishing to know more about the numerical methods employed in the various codes will find this information most comprehensively presented in the corresponding publications. In Section 4, we present briefly the graphics programs, together with sample outputs. This paper does not contain any operational instructions; a user manual is provided together with each of the codes. We finally provide some general considerations.

2 FORMULATION

This section gives a brief overview of the equations which are solved by the various programs and how the solutions are represented numerically. The quantities in the different codes are steered by seemingly arbitrarily named parameters. These are however consistent across the whole suite of programs and are detailed here for convenience.

2.1 The heat equation

The equation defining the advection of heat (see Gubbins & Roberts 1987) is

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

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

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

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

where t, r, θ, and ϕ are the time, radius, colatitude, and longitude, respectively. 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 basic state 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

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

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

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

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,

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

Θ is the function which is solved for in all of the calculations. $T_{\rm a}$ 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:

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

The constants c_a, b_1, b_2, c_c, and c_d are arbitrarily named, with no physical meaning attached 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 Gubbins & Roberts (1987), 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 (with units s^{-2}). The linear dependence of $\boldsymbol{g}$ on r is a good approximation for the core (see for example Dziewonski & Anderson 1981 or Anderson 1989), 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, $\boldsymbol{k}$, perpendicular to the equatorial plane and oriented upwards at the origin given counterclockwise rotation. 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

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

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

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

and

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

Note that these definitions are different from those of, for example Bullard & Gellman (1954), 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 generalized 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 Numerical representation

The parameters governing the scalings of the terms in the heat, momentum, and induction equations are specified in equations (9), (15), and (23), respectively. We here describe very briefly how the solution vectors representing the various fields are constructed and stored on file.

In equations (8), (18), and (19), we have five scalar functions of space and time, Θ, ${}^P v$, ${}^T v$, ${}^P B$, and ${}^T B$, which can all be expressed in the form

$$\begin{aligned} f(t, r, \theta, \phi) = f_0^{0c}(r, t) & \\ & + \sum_{l=1}^{L} f_l^{0c}(r, t) P_l^0(\cos\theta) \\ & + \sum_{l=1}^{L}\sum_{m=1}^{M(l)} f_l^{mc}(r, t) \cos m\phi\, P_l^m(\cos\theta) \\ & + \sum_{l=1}^{L}\sum_{m=1}^{M(l)} f_l^{ms}(r, t) \sin m\phi\, P_l^m(\cos\theta), \end{aligned} \tag{24}$$

where $P_l^m(\cos\theta)$ is an associated Legendre function, here satisfying the Schmidt quasi-normalization condition

$$\int_0^{\pi} \left[P_l^m(\cos\theta)\right]^2 \sin\theta d\theta = \frac{2(2-\delta_{m0})}{2l+1}. \tag{25}$$

The entire solution can be described completely in terms of the radial functions $f_l^{mc}(r, t)$ and $f_l^{ms}(r, t)$ where the scalar f can be each of Θ, ${}^P v$, ${}^T v$, ${}^P B$, and ${}^T B$. For a true 3D solution, the integer function M is given by

$$M(l) = l. \tag{26}$$

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 maximum value of m, $M_{\max}$ for instance, such that

$$M(l) = \min(l, M_{\max}). \tag{27}$$

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 Christensen et al. (2001) 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 3D solution for the model parameters described in Christensen et al. (2001) will integrate towards a solution which is zero for all m which are not multiples of m_0. Care must of course 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 Sarson & Jones 1999, 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. Improvements in computational power since the 1990s have of course limited the circumstances in which such measures are necessary, but exploiting symmetry considerations may still permit far higher resolution in radius or latitude than would be possible with full resolution in azimuth.

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). There are numerous additional symmetry considerations described by Gubbins & Zhang (1993) which can further reduce the number of radial functions in the expansions of the form in equation (24) needed to describe a completely viable physical solution.

If N_h gives the total number of radial functions of the form $f_l^{mc}(r, t)$ and $f_l^{ms}(r, t)$ (cf. equation 24) we need, and N_r gives the number of points in radius, then a vector of $N_h \times N_r$ numbers is sufficient to store our solution. In addition, we need a single array of N_r numbers giving the values of radius to use and a set of five integers describing the nature of each of the N_h radial functions. The first of these integers tells us which of Θ, ${}^P v$, ${}^T v$, ${}^P B$, or ${}^T B$ the radial function represents. The second gives l and the third gives m for a $\cos m\phi$ function or $-m$ for a $\sin m\phi$ function. The fourth and fifth integers specify the boundary condition to be applied to the radial function at the inner and outer boundary, respectively.

Each solution is saved in three files. These can have any name the user wishes but we usually label them `[stem].ints`, `[stem].vecs`, and `[stem].xarr`. The `ints` file has in the first row the number of radial functions (N_h) and this is followed by N_h rows each containing the five integers described above. The `xarr` file contains N_r numbers defining the radial node locations (which are in principle arbitrary, but must increase strictly) following a header line of two integers: `NR` and `IFORM` (where the latter describes the format of the numbers). The `vecs` file contains $N_h \times N_r$ numbers providing the values of the radial functions themselves following a header line of four integers: `IORD NR,NH` and `IFORM`. If `IORD` is 3 then the value of radial function i_h at node i_r is stored at location $(i_r - 1)N_h + i_h$ in the solution vector. (This ordering groups together the values of every spherical harmonic coefficient at a given radial grid node, as is needed for all programs that form matrices with interactions between different spherical harmonics. This is the case for the kinematic dynamo and boundary-locked convection codes that need to maintain a banded structure of large matrices.) If `IORD` is 4, it is stored at location $(i_h - 1)N_r + i_r$. (This ordering groups together the values of all radial grid nodes for a given spherical harmonic function. This is the ordering necessary for the time-stepping codes for which the matrices do not couple different radial functions.) It is possible to switch from one ordering to another using the *svpnsmap* auxiliary program. The number `IFORM` always takes the value 1 for the FORTRAN type format `5(1PD16.7)`; no other format was ever considered, although this could easily be implemented.

For dynamo calculations where we have a conducting inner core and/or a conducting layer at the base of the mantle, we represent the magnetic field in a separate set of files to the other functions (i.e. six files in total). The radii for the outer core nodes must correspond exactly in both sets of `xarr` files. The naming convention in the examples given is `intsm`, `vecsm`, and `xarrm` for the magnetic field and `intsv`, `vecsv`, and `xarrv` for the remaining functions. A full overview of the mathematical and numerical formalism behind the codes is contained in the file `DOC_fundamentals.pdf` document in the `additional_documentation` directory of the `LEOPACK-2022-revision` repository.

3 OVERVIEW OF MAIN PROGRAMS

This section provides a brief overview of the main codes arranged in groups of applications. An at-a-glance overview is provided in Fig. 1 while the lists below provide a somewhat expanded description of applicability.

3.1 The kinematic dynamo

All of these codes impose a prescribed and fixed velocity field and solve for the magnetic-field-generating properties of this flow. Only two flows have been implemented: the flow of Dudley & James (1989) in the code *djiepgrf* and the flow of Kumar & Roberts (1975) in all the other codes. All of the codes solve an eigenvalue system using the Implicitly Restarted Arnoldi Method (Arnoldi 1951) as implemented in the ARPACK software (Sorensen 1992; Lehoucq, Sorensen & Yang 1998). To specify a different form of flow would necessitate creating a new code from one of these programs as starting points.

The five codes are as follows:

(i) **djiepgrf**. Calculates the generally complex growth rates, σ, of a magnetic field subject to the flow of Dudley & James (1989) with different specifications of coefficients.

(ii) **kriepgrf**. Calculates the generally complex growth rates, σ, of a magnetic field subject to the flow of Kumar & Roberts (1975) with different specifications of coefficients.

(iii) **krcmrnif**. Calculates a critical magnetic Reynolds number, R_m^c, for which the real part of the growth rate, σ, is zero for the flow of Kumar & Roberts (1975). The procedure is iterative and not guaranteed to find a value of R_m^c.

(iv) **krddmcmrnif**. Calculates a critical magnetic Reynolds number, R_m^c, for which the real part of the growth rate, σ, is zero for the flow of Kumar & Roberts (1975) but where the scalings of the different components of the flow are defined by the D and M parameters (controlling the relative strengths of the differential rotation and meridian circulation, respectively) as introduced by Gubbins et al. (2000a). The procedure is iterative and not guaranteed to find a value of R_m^c. This code was subsequently used to map out the parameter spaces in the studies of Gubbins et al. (2000b), Gubbins & Gibbons (2002), and Gubbins & Gibbons (2009).

(v) **krssgeps**. Calculates a critical magnetic Reynolds number, R_m^c, for a strictly zero growth rate. This program solves a generalized eigenvalue problem. The flow is of the form specified by Kumar & Roberts (1975).

3.2 The onset of thermal convection

There are four codes for solving for the critical Rayleigh number for the onset of thermal convection in a rotating fluid-filled sphere or spherical shell. (The sphere is just a special case of the spherical shell with the inner radius set to zero.) The exact definition of the Rayleigh number can vary so, in these programs, we refer only to the parameter c_h specified in equation (15). All programs perform essentially the same task. *linons1* and *linons2* differ only in that *linons1* takes a single initial guess for c_h^c and it estimates an initial gradient of the growth rate as a function of c_h based on a small perturbation, whereas *linons2* solves for a value of c_h^c between two bounds within which c_h^c is known to be found. The codes *sbrlinons1* and *sbrlinonsd* apply a solid body rotation in order to solve for a system rotating at the same angular velocity as the convection rolls at onset. This was found to be necessary as the eigensolvers struggle to find the correct eigenvalues when the imaginary parts of the growth rates become too large. By imposing a solid body rotation, we solve in a rotating frame of reference and the imaginary part of the growth rate is defined by the difference between the drift rate and the imposed rotation. The growth rate can therefore become purely real if the solid body rotation corresponds exactly with the drift rate. *sbrlinons1* imposes a fixed solid body rotation and *sbrlinonsd* attempts to iterate to the solid body rotation which makes the relative drift rate zero.

Note that this problem decouples in the wavenumber, m, such that each run is performed for a specified m. Each set of parameters has a critical Rayleigh number for each wavenumber so the overall critical Rayleigh number will have an associated preferred wavenumber.

The four codes are as follows:

(i) **linons1**. Finds the critical Rayleigh number for the onset of thermal convection in a rotating fluid-filled sphere or spherical shell from an initial guess.

(ii) **linons2**. As for *linons1* but iterates between a lower and an upper bound.

(iii) **sbrlinons1**. As for *linons2* but imposes a fixed solid body rotation. (i.e. solves in a fixed rotating frame of reference.)

(iv) **sbrlinonsd**. As for *sbrlinons1* but attempts to solve for the solid body rotation for which the imaginary part of the growth rate vanishes for the onset of thermal convection. (i.e. solves in a rotating frame of reference that is modified on each iteration.)

These codes were validated by reproducing the results of Zhang & Busse (1987), in which a fixed temperature was specified at the outer boundary, and were used to calculate the critical Rayleigh numbers in the study of Gibbons et al. (2007), for which a fixed heat-flux was specified at the outer boundary.

3.3 Boundary-locked convection

When there is a spatially varying temperature or heat-flux as a boundary condition, convection will always occur. That is to say that there is no critical regime like there is for the onset of convection in the uniform boundary problem. Convection driven by the thermal boundary is even possible when the fluid is stably stratified. There are three codes for calculating boundary-locked convection as follows:

(i) **blscnlsc**. Solves iteratively for steady-state boundary-locked or boundary-driven convection. Note that there is no time-dependence to the solution but that we cannot tell, from this calculation alone, whether or not the solution is stable or not.

(ii) **blscnlsic**. As for *blscnlsc* but the resulting steady-state solution is also subject to an instability analysis. This is an eigenvalue problem and a growth rate with a positive real part indicates that a perturbation

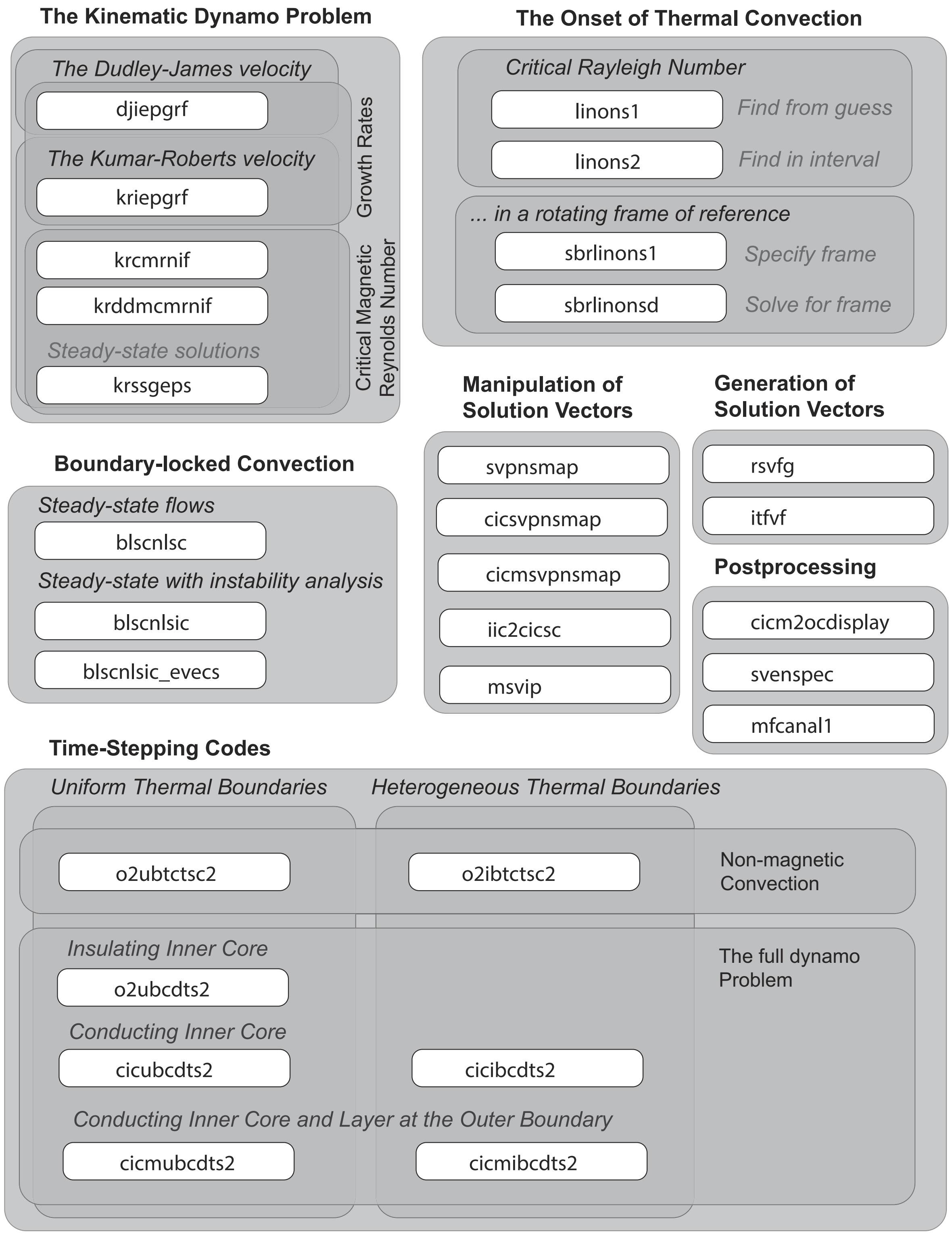


Figure 1. At-a-glance summary of the main programs listed in Section 3.

to the steady-state boundary-locked or boundary-driven flow would grow.

(iii) **blscnlsic_evecs**. As for *blscnlsic* but produces different output. The programs are essentially identical but *blscnlsic_evecs* writes out all of the eigenvectors from the instability analysis. This of course will consume a lot of disk space, so only use this program for spot-checking and displaying of the spatial form of the thermal instabilities.

These codes were validated by reproducing the results of Zhang & Gubbins (1993) and Zhang & Gubbins (1996), for which a fixed temperature outer-boundary-condition was imposed, and were used to calculate the boundary-locked solutions in the study of Gibbons et al. (2007), for which a fixed heat-flux outer-boundary-condition was imposed. An earlier version of the code *blscnlsc* had been used to calculate the solutions in the study of Gibbons & Gubbins (2000).

Note that all of these three programs use large matrices with cross-terms between different radial functions. The size of these matrices increases rapidly as the spatial resolution of the problem increases. You will rapidly reach a limit regarding memory demands and time-to-solution as the dimensions of the problem increase. However, any boundary-driven or boundary-locked flow should also result from the time-stepping code *o2ibtctsc2*. If you run *o2ibtctsc2* and *blscnlsc* with identical resolution and identical parameter settings then the time-stepping code should result in a steady-state solution identical to that obtained using *blscnlsc*. If the steady-state solution is unstable, as indicated by the program *blscnlsic*, then this should result in a time-dependent solution in *o2ibtctsc2*. The philosophy behind the two types of programs is rather different and the likeness of the two solutions should provide a degree of validation.

3.4 Time-stepping codes for non-magnetic convection

There are two codes which solve the heat and the vorticity equations in the absence of a magnetic field for general time-dependent temperature and flow fields. They differ only in whether we wish a spatially heterogeneous thermal boundary condition or not. The two codes are

(i) **o2ubtctsc2**. Time-stepping code for non-magnetic convection with a uniform temperature or heat-flux at the boundaries. A slightly earlier version of this code was used for the contribution labelled GJZ in the benchmark study (Case 0) of Christensen et al. (2001).

(ii) **o2ibtctsc2**. Time-stepping code for non-magnetic convection with the possibility of spatially varying temperature or heat-flux at the boundaries. This code was used to compute the time-dependent solutions in the studies of Gubbins & Gibbons (2004) and Gibbons et al. (2007). These studies considered only laterally varying heat-flux at the outer boundary, although the code also allows full flexibility in specifying laterally varying heat-flux at the boundary of the inner core.

3.5 Time-stepping codes for the full dynamo problem

There are five codes for solving the full set of equations for temperature, flow, and magnetic fields. They are summarized as follows:

(i) **o2ubcdts2**. Time-stepping code for convection-driven magnetic field generation with a uniform temperature or heat-flux at the boundaries and an insulating inner core. A slightly earlier version of this code was used for the contribution labelled GJZ in the benchmark study (Case 1) of Christensen et al. (2001).

(ii) **cicubcdts2**. Time-stepping code for convection-driven magnetic field generation with a uniform temperature or heat-flux at the boundaries and a conducting inner core that co-rotates with the mantle. The fluid must have a no-slip boundary condition at the inner boundary.

(iii) **cicibcdts2**. Time-stepping code for convection-driven magnetic field generation with the possibility of a spatially varying temperature or heat-flux at the boundaries and a conducting inner core that co-rotates with the mantle. The fluid must have a no-slip boundary condition at the inner boundary.

(iv) **cicmubcdts2**. Time-stepping code for convection-driven magnetic field generation with a uniform temperature or heat-flux at the boundaries and a conducting inner core that co-rotates with the mantle and, optionally, a conducting layer at the base of the mantle. The fluid must have a no-slip boundary condition at boundaries at which we solve for a magnetic field on either side.

(v) **cicmibcdts2**. Time-stepping code for convection-driven magnetic field generation with the possibility of a spatially varying temperature or heat-flux at the boundaries and a conducting inner core that co-rotates with the mantle and, optionally, a conducting layer at the base of the mantle. The fluid must have a no-slip boundary condition at boundaries at which we solve for a magnetic field on either side.

3.6 Auxiliary codes

The 10 codes listed here are miscellaneous tools for manipulating files acted on by the main programs:

(i) **svpnsmap**. Converts a set of solution vectors from one spatial mesh to another. We can change the locations of the radial grid nodes and change the set of radial functions present.

(ii) **cicsvpnsmap**. As for *svpnsmap* but operates on the double sets of solution vectors (i.e. six files) for the conducting inner core calculations.

(iii) **cicmsvpnsmap**. As for *cicsvpnsmap* but covers the cases where we also have a conducting layer at the base of the mantle.

(iv) **iic2cicsc**. Converts a set of files from the insulating inner core code (*o2ubcdts2*) to the six-file conducting inner core format for one of the other magnetic field time-stepping codes.

(v) **msvip**. Combines multiple sets of solution vector files. Useful when displaying solutions with inhomogeneous thermal boundary conditions.

(vi) **rsvfg**. Generates a random initial solution vector (a set of three files).

(vii) **itfvf**. Generates a set of files for imposing a spatially varying thermal boundary condition.

(viii) **cicm2ocdisplay**. Combines the magnetic field and flow/temperature files to a single solution vector specification (i.e. 3 files) for the outer core only. This is to generate temporary files for plotting since the constant radius plotting programs only take in single sets of files.

(ix) **svenspec**. Calculates the kinetic and magnetic energy as a function of l and m.

(x) **mfcanal1**. Calculates a number of properties of the magnetic field. The properties calculated were specifically chosen for one particular investigation. If more different properties were to be required, this would necessitate a new code. However, this program may provide a suitable starting point.

4 OVERVIEW OF GRAPHICS PROGRAMS

There are 10 programs which provide postscript displays of the temperature, flow, and magnetic fields generated. An at-a-glance overview is provided in Fig. 2. There are three types of projection which we will consider separately: rectangular plots of fields for constant radius (two codes), spherical shell sections in polar coordinates (four codes), and quasi-3D images of fields on the surface of a sphere (four codes).

Should the user prefer to use alternative software for making the plots [for example, the Generic Mapping Tools (GMT): Wessel et al. 2019], the programs in the gprograms directory could easily be adapted to simply write out the functions to be plotted to ASCII files which could be interpreted by other plotting programs. Subroutines such as `CONSTANT_R_RECT_EVAL`, `EQ_SEC_POLAR_EVAL`, and

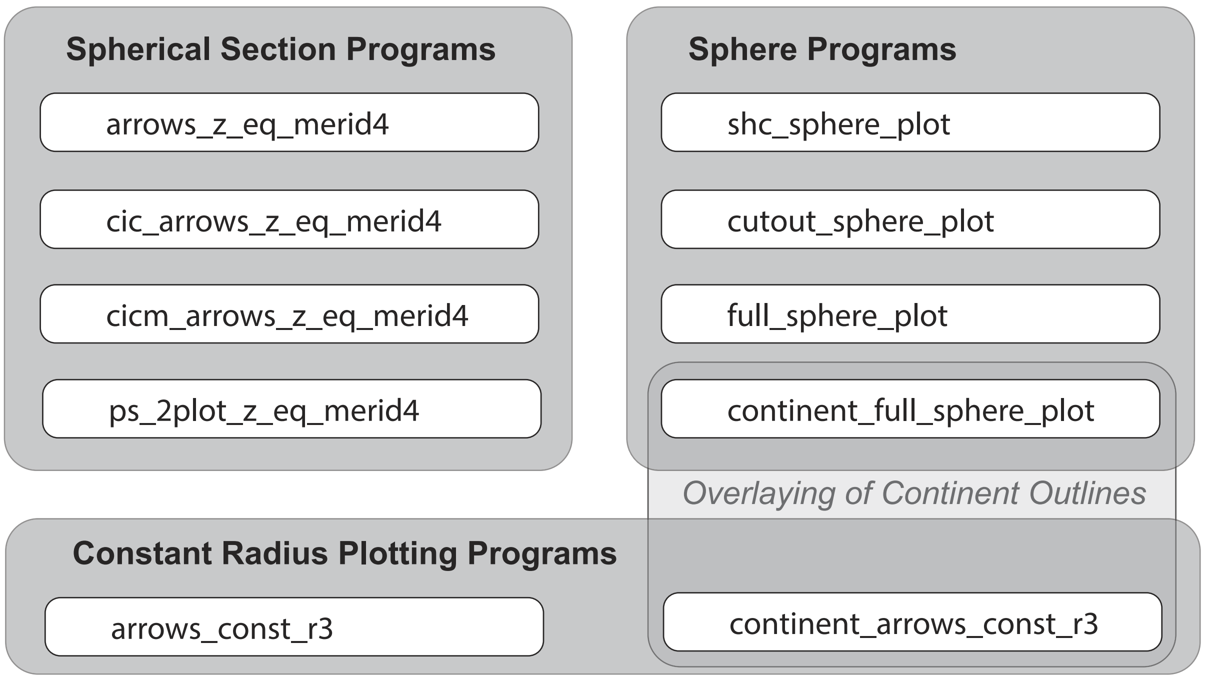


Figure 2. At-a-glance summary of the graphics programs listed in Section 4.

MERID_SEC_POLAR_EVAL look after all the interpolation of radial functions and the evaluation of the correct spherical harmonic terms. These routines return simple 2D arrays with the requested functions. (The coordinates in r, θ, and ϕ are always equally spaced between the requested limits.)

4.1 Programs for plotting on fixed-radius projections

Two codes are provided for plotting functions and flow arrows on surfaces of constant radius. They are:

(i) **arrows_const_r3**. Colour or contour plots of scalar functions (optionally with flow arrows) on a surface for a specified constant radius. Sample output in Fig. 3(a).

(ii) **continent_arrows_const_r3**. As for *arrows_const_r3* except that a simple outline of the continents is drawn on top. Sample output in Fig. 3(b).

4.2 Programs for plotting on spherical sections

Four codes are provided for plotting scalar functions and, optionally, flow arrows in equatorial sections, meridian sections, or sections of constant distance from the equatorial plane (constant z).

(i) **arrows_z_eq_merid4**. Plots section of scalar and, optionally, flow arrows for spherical shells (i.e. inner core is not included). Sample output in Fig. 4(a).

(ii) **cic_arrows_z_eq_merid4**. As for *arrows_z_eq_merid4* except that it plots solutions from the conducting inner core codes. Sample output in Fig. 4(b).

(iii) **cicm_arrows_z_eq_merid4**. As for *cicm_arrows_z_eq_merid4* except that it plots solutions with a conducting layer at the base of the mantle. Sample output in Fig. 4(c).

(iv) **ps_2plot_z_eq_merid4**. Plots two hemispherical sections side by side. There are many examples in the study of Gubbins & Gibbons (2002).

4.3 Programs for plotting on the surface of a sphere

There are four programs which plot scalar fields on the surface of a sphere.

(i) **shc_sphere_plot**. Plots a scalar function on a spherical surface specified by a file of spherical harmonic coefficients. Sample output in Fig. 5(a).

(ii) **cutout_sphere_plot**. Plots a scalar function from a standard 3-file solution vector on a spherical surface with a cut out section as displayed in Fig. 5(b).

(iii) **full_sphere_plot**. Plots a scalar function from a standard 3-file solution vector on a spherical surface. Sample output in Fig. 5(c).

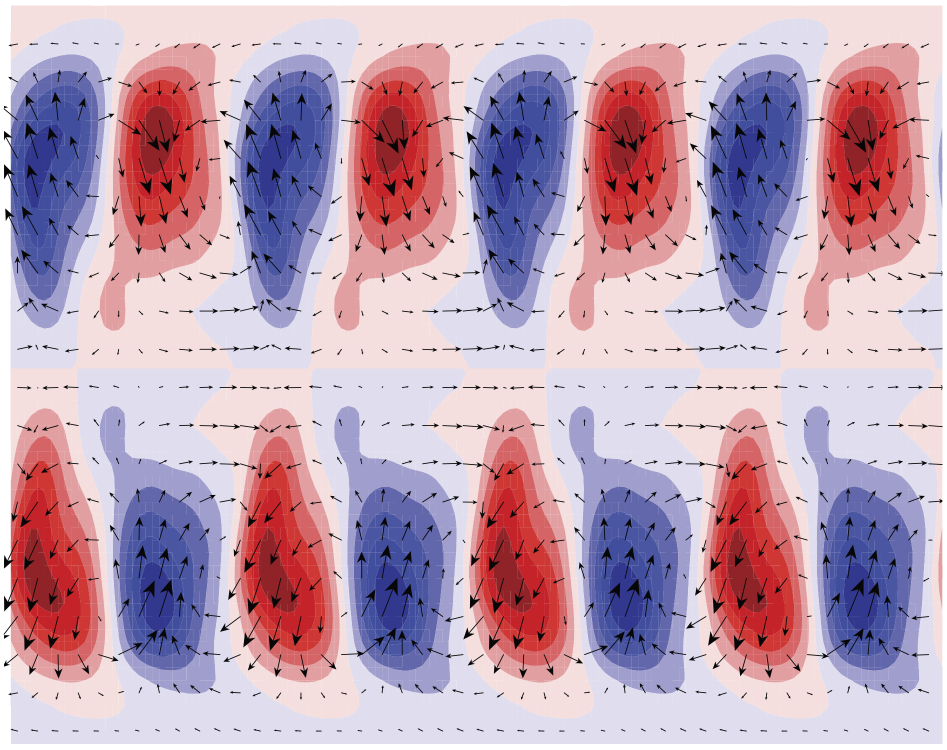
(iv) **continent_full_sphere_plot**. As *full_sphere_plot* except that a simple outline of the continents is drawn on top. Sample output in Fig. 5(d).

5 CONCLUSIONS

We present a set of codes for calculating and displaying solutions to diverse problems in convection and magnetic field generation in rotating fluid-filled spheres and spherical shells. The codes are freely available from both github and Zenodo. Each of the codes has a pdf user manual with an explanation of the parameters used and at least one set of input files for a sample run. (Some of the codes have several worked examples.) The main codes are purely written in FORTRAN with no external dependencies and compile under the free gfortran compiler (GNU Fortran version 9.4.0 on Ubuntu 20.04.1 was used for the most recent test of the code prior to submitting this work). The graphics codes require in addition the PGPLOT library (version 5.2 was used). All codes were tested thoroughly prior to uploading to github in late 2022, with every worked example being reproduced using newly compiled code.

These codes represent a relatively brief snapshot in the development of the dynamo simulation toolbox at the University of Leeds but are worth preserving and detailing as they

(a)



(b)

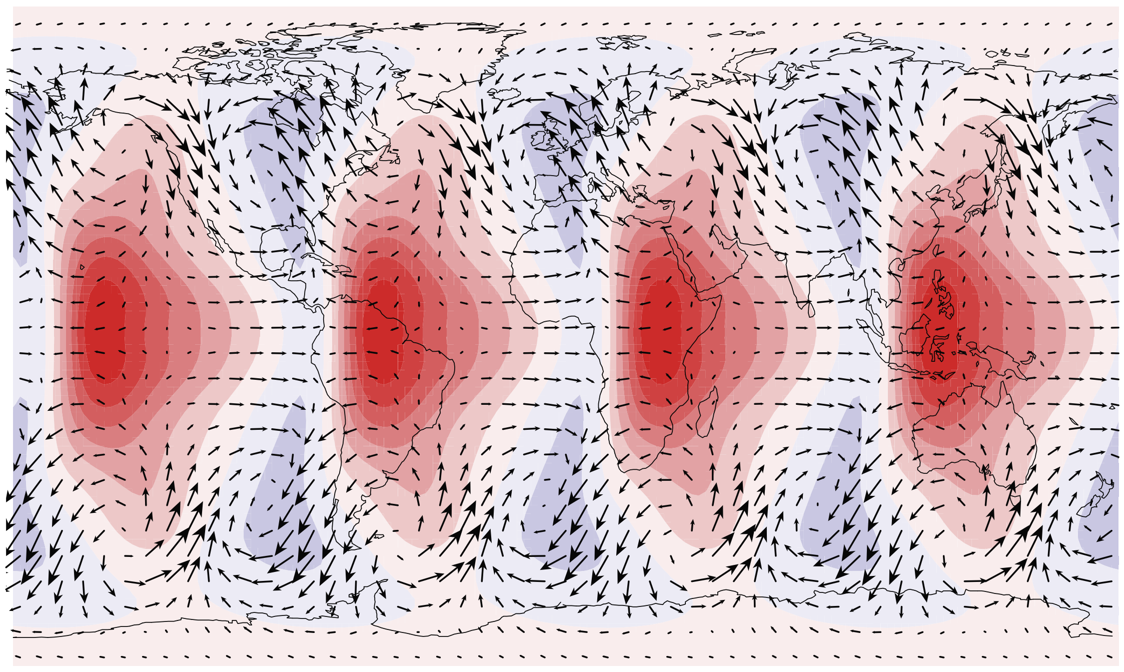


Figure 3. Example outputs from the constant radius plotting programs listed in Section 4.1 Panel (a) shows arrows of flow and coloured contours of v_θ for the Case 0 Dynamo Benchmark study of Christensen et al. (2001) generated using the solution vector and an input file in the directory GRAPHICS_arrows_const_r3 of the distribution. Panel (b) shows contours of the temperature perturbation together with outlines of continents and arrows of flow for the same solution from the directory GRAPHICS_continent_arrows_const_r3 of the distribution. All documentation for each program is contained within a pdf file in the appropriate directory.

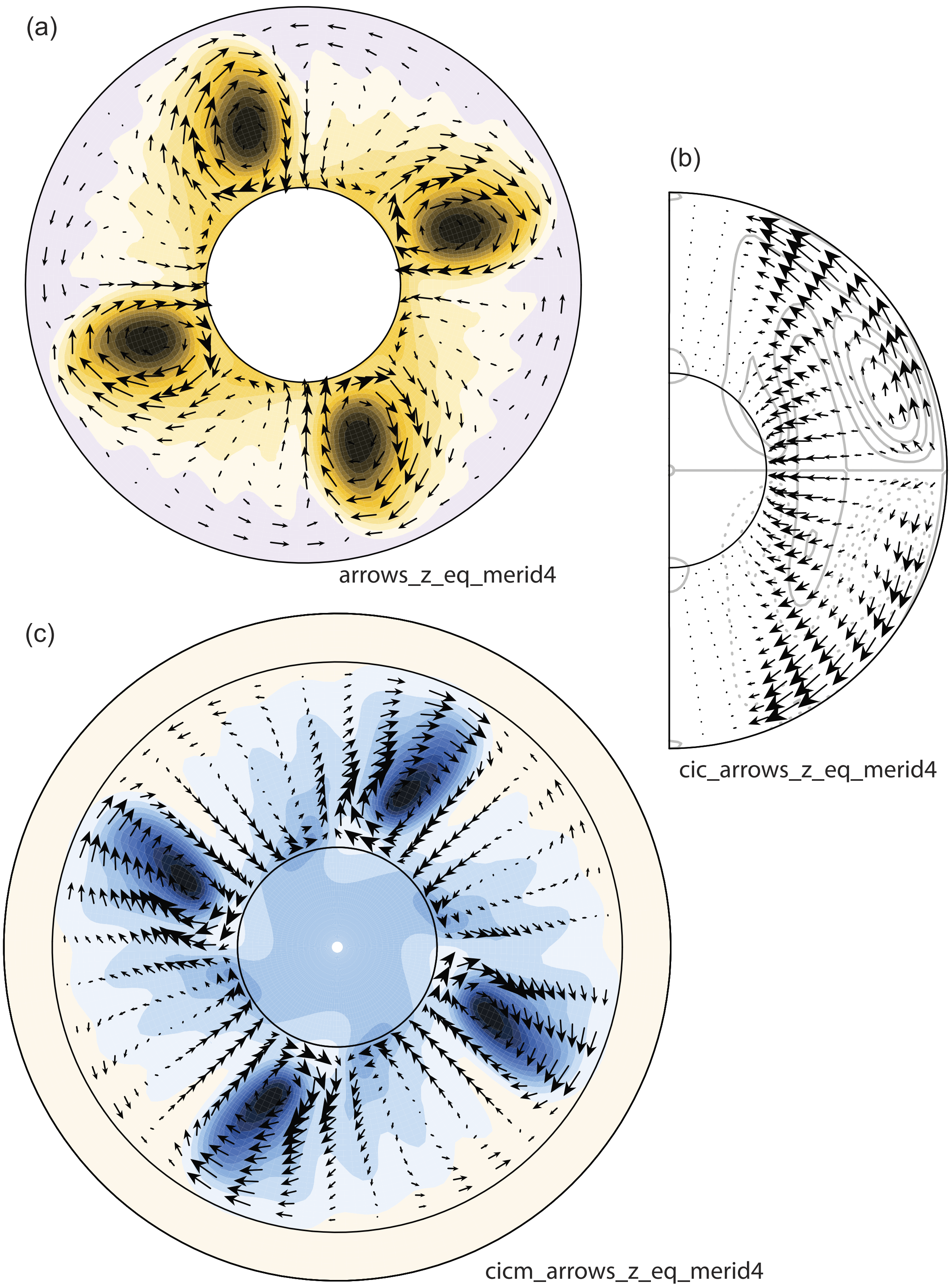


Figure 4. Example outputs from the spherical section plotting programs as indicated listed in Section 4.2 The plots in panels (a), (b), and (c) can be generated using the solution vectors and input files in the directories of the distribution GRAPHICS_arrows_z_eq_merid4, GRAPHICS_cic_arrows_z_eq_merid4, and GRAPHICS_cicm_arrows_z_eq_merid4, respectively. All documentation for each program is contained within a pdf file in the appropriate directory.

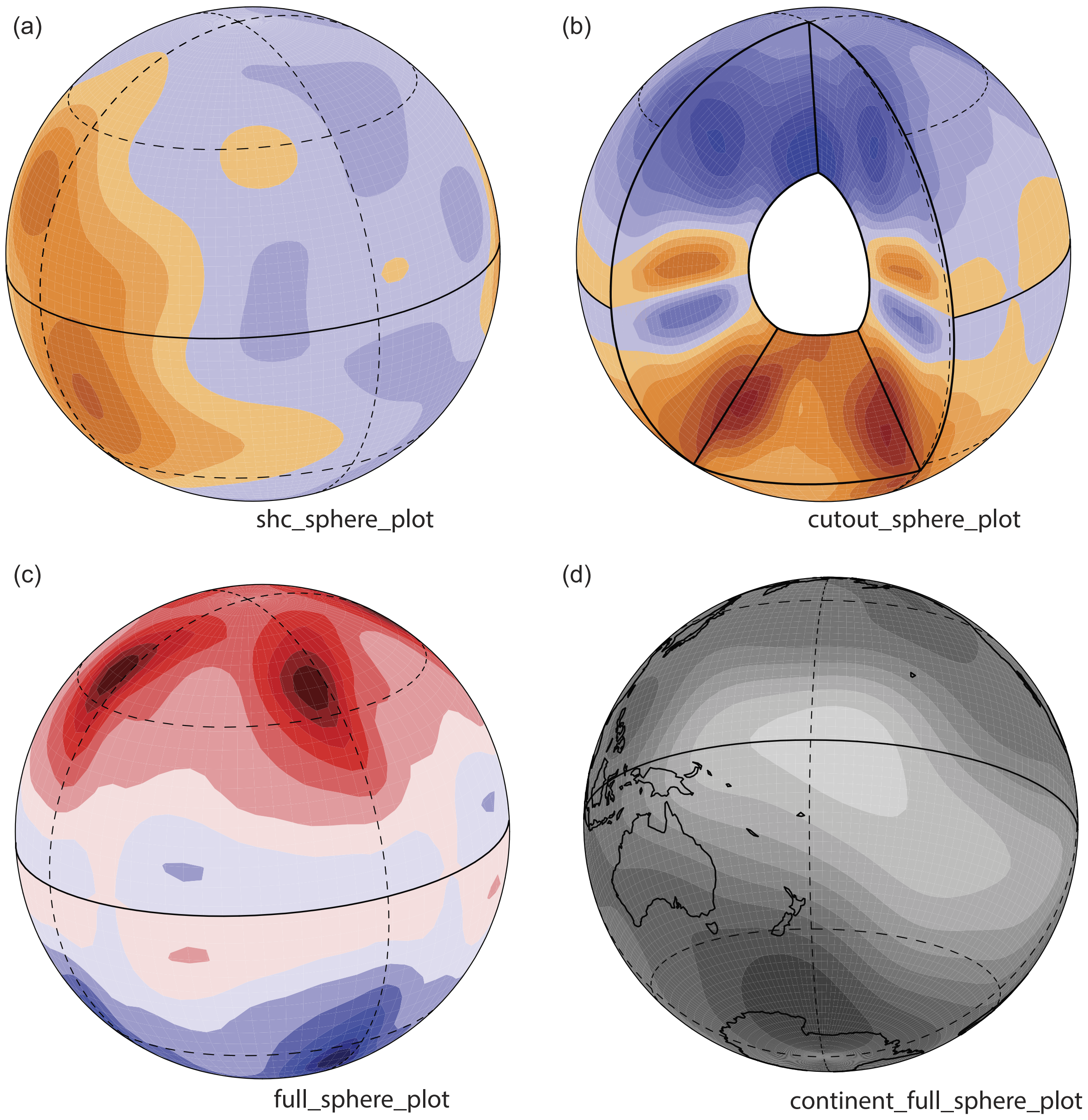


Figure 5. Example outputs from the spherical surface plotting programs listed in Section 4.3 The plots in panels (a), (b), (c), and (d) can be generated using the solution vectors and input files in the directories of the distribution GRAPHICS_shc_sphere_plot, GRAPHICS_cutout_sphere_plot, GRAPHICS_full_sphere_plot, and GRAPHICS_continent_full_sphere_plot, respectively. All documentation for each program is contained within a pdf file in the appropriate directory.

(i) were documented in unprecedented depth at this time,

(ii) formed a common basis for subsequent diverging developments and optimizations of the codes, and

(iii) consider many aspects of geodynamo simulation within a common framework.

The codes described in this paper (LEOPACK) were used for numerous publications as detailed. Subsequent studies by the Leeds group addressed more diverse problems and required both different approaches and codes with better scalability. The code used by Willis, Sreenivasan & Gubbins (2007) inherited much of the LEOPACK approach, including its fundamental spectral-finite difference discretization. The LEOPACK codes contributed both as a source of routines for certain inherited elements and, more widely, for checking the accuracy of the new code.

A limitation is that the codes with conducting inner cores do not permit the inner core to rotate at a different rate to the outer boundary and do not permit stress-free boundary conditions. At the time of the benchmark study of Christensen et al. (2001), a conducting inner core had not been implemented at all and there is no *GJZ* contribution for the Case 2 dynamo simulation. Subsequent development of these codes at the University of Leeds reproduced the Case 2 benchmark with a completely different representation of the magnetic field. The co-rotating inner core limitation means that the user cannot, for example, study the torque balance on the inner core. A second limitation is that this initial set of codes was not parallelized.

Although the basis in Fortran 77 brings a number of disadvantages, such as the lack of dynamic memory allocation, it has helped the codes' longevity. The lack of dependence on external libraries, often subject to frequent updates, has meant that the code has required no modifications since its initialization. It is our hope that the codes are easy to navigate, compile, and run, and that they will provide a useful baseline comparison for subsequent code developments in geodynamo simulations.

ACKNOWLEDGEMENTS

We would like to thank Graeme Sarson whose PhD thesis provided many clear formulations that were included in various parts of the code documentation. We acknowledge the help of C. A. Jones and K. Zhang who supervised the development of earlier versions of the code at the University of Exeter between 1998 October and 2000 September.

CD acknowledges support from the Natural Environment Research Council (NERC) grant number NE/V010867/1.

We are grateful to two anonymous reviewers for very thorough reviews of this manuscript and for very helpful comments that have significantly improved the text.

DATA AVAILABILITY

The code and examples are all available from
`https://github.com/stevenjgibbons/LEOPACK-2022-revision`
and the repository is obtained by typing
```
git clone
https://github.com/stevenjgibbons/LEOPACK-2022-revision.git
```
A permanent zip file of the release v1.0.1 is found on Zenodo at
`https://doi.org/10.5281/zenodo.7932800`

The graphics programs require the PGPLOT library found at https://sites.astro.caltech.edu/~tjp/pgplot/ (last accessed 2023 August).

The XSHELLS code is available from `https://nschaeff.bitbucket.io/xshells/` (last accessed 2023 August).

The SINGE code is available from `https://bitbucket.org/vidalje/singe/src/master/` (last accessed 2023 August).

The psconvert tool is part of the free Generic Mapping Tools software (GMT: Wessel et al. 2019) and can be obtained from `https://www.generic-mapping-tools.org/` (last accessed 2023 August).

REFERENCES

Anderson D. L., 1989, Theory of the Earth. Blackwell Scientific Press, Oxford
Arnoldi W. E., 1951, Quart. J. Appl. Math., 9, 17
Aubert J., 2023, Geophys. J. Int., 235, 468
Bullard E. C., Gellman H., 1954, Phil. Trans. R. Soc. Lond. A, 247, 213
Christensen U., Olson P., Glatzmaier G. A., 1998, Geophys. Res. Lett., 25, 1565
Christensen U., Olson P., Glatzmaier G. A., 1999, Geophys. J. Int., 138, 393
Christensen U. R. et al., 2001, Phys. Earth Planet. Inter., 128, 25
Dudley M. L., James R. W., 1989, Proc. R. Soc. Lond. A, 425, 407
Dziewonski A. M., Anderson D. L., 1981, Phys. Earth planet. Inter., 25, 297
Folch A. et al., 2023, Future Gener. Comp. Syst., 146, 47
Fournier A., Nerger L., Aubert J., 2013, Geochem. Geophys. Geosyst., 14, 4035
Gastine T., Wicht J., Aubert J., 2016, J. Fluid Mech., 808, 690
Gibbons S. J., Gubbins D., 2000, Geophys. J. Int., 143, 631
Gibbons S. J., Gubbins D., Zhang K., 2007, Geophys. Astrophys. Fluid Dyn., 101, 347
Glatzmaier G. A., Roberts P. H., 1995, Nature, 377, 203
Gubbins D., Gibbons S., 2002, Geophys. Astrophys. Fluid. Dyn., 96, 481
Gubbins D., Gibbons S. J., 2004, Low Pacific Secular Variation in 'Timescales of the Paleomagnetic Field'. American Geophysical Union (AGU), Washington, p. 279
Gubbins D., Gibbons S. J., 2009, Geophys. J. Int., 177, 71
Gubbins D., Roberts P. H., 1987, in Jacobs J. A., ed., Geomagnetism Vol. II. Academic Press, Cambridge, MA, p. 1
Gubbins D., Sarson G., 1994, Nature, 368, 51
Gubbins D., Zhang K., 1993, Phys. Earth Planet. Inter., 75, 225
Gubbins D., Barber C. N., Gibbons S., Love J. J., 2000a, Proc. R. Soc. Lond. A, 456, 1333
Gubbins D., Barber C. N., Gibbons S., Love J. J., 2000b, Proc. R. Soc. Lond. A, 456, 1669
Jackson A. et al., 2014, Geophys. J. Int., 196, 712
Jones C., Boronski P., Brun A., Glatzmaier G., Gastine T., Miesch M., Wicht J., 2011, Icarus, 216, 120
Kono M., Roberts P. H., 2002, Rev. Geophys., 40, 4
Kumar S., Roberts P. H., 1975, Proc. R. Soc. Lond. A, 344, 235
Lehoucq R. B., Sorensen D. C., Yang C., 1998, ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods. SIAM, Philadelphia
Long R. S., Mound J. E., Davies C. J., Tobias S. M., 2020, J. Fluid Mech., 889, A7
Marti P. et al., 2014, Geophys. J. Int., 197, 119
Matsui H. et al., 2016, Geochem. Geophys. Geosyst., 17, 1586
Monville R., Vidal J., Cébron D., Schaeffer N., 2019, Geophys. J. Int., 219, S195
Mound J. E., Davies C. J., 2023, Nat. Geosci., 16, 380
Pearson T., 2011, Astrophysics Source Code Library, record ascl:1103.002
Sarson G. R., Jones C. A., 1999, Phys. Earth Planet. Inter., 111, 3
Schaeffer N., Jault D., Nataf H.-C., Fournier A., 2017, Geophys. J. Int., 211, 1
Sorensen D. C., 1992, SIAM J. Matrix Anal. Appl., 13, 357
Vidal J., Schaeffer N., 2015, Geophys. J. Int., 202, 2182
Wessel P., Luis J. F., Uieda L., Scharroo R., Wobbe F., Smith W. H. F., Tian D., 2019, Geochem. Geophys. Geosyst., 20, 5556
Willis A. P., Sreenivasan B., Gubbins D., 2007, Phys. Earth Planet. Inter., 165, 83
Zhang K., Busse F. H., 1987, Geophys. Astrophys. Fluid Dyn., 39, 119
Zhang K., Gubbins D., 1993, J. Fluid Mech., 250, 209
Zhang K., Gubbins D., 1996, Phys. Fluids, 8, 1141

SUPPORTING INFORMATION

Supplementary data are available at *RASTAI* online.

suppl_data

Please note: Oxford University Press is not responsible for the content or functionality of any supporting materials supplied by the authors. Any queries (other than missing material) should be directed to the corresponding author for the article.

This paper has been typeset from a TEX/LATEX file prepared by the author.