Journal of Computational Physics 229 (2010) 7997–8010

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A comparison of high-order time integrators for thermal convection in rotating spherical shells

F. Garcia [a], M. Net [a], B. García-Archilla [b], J. Sánchez [a,*]

[a] Departamento de Física Aplicada, Universitat Politècnica de Catalunya, Jordi Girona Salgado s/n, Campus Nord. Mòdul B4, 08034 Barcelona, Spain
[b] Departamento de Matemática Aplicada II, Universidad de Sevilla, Escuela Superior de Ingenieros, Camino de los Descubrimientos, s/n, 41092 Sevilla, Spain

ARTICLE INFO

Article history:
Received 4 February 2010
Received in revised form 30 June 2010
Accepted 2 July 2010
Available online 8 July 2010

Keywords:
Time integration methods
Backward differentiation-extrapolation formulae
Krylov methods
Spectral methods
Spherical shells
Thermal convection

ABSTRACT

A numerical study of several time integration methods for solving the three-dimensional Boussinesq thermal convection equations in rotating spherical shells is presented. Implicit and semi-implicit time integration techniques based on backward differentiation and extrapolation formulae are considered. The use of Krylov techniques allows the implicit treatment of the Coriolis term with low storage requirements. The codes are validated with a known benchmark, and their efficiency is studied. The results show that the use of high-order methods, especially those with time step and order control, increase the efficiency of the time integration, and allows to obtain more accurate solutions.

1. Introduction

The study of the thermal convection in rotating spherical geometries is fundamental to explain many geophysical and astrophysical phenomena, such as the generation of the magnetic fields, or the differential rotation observed in the atmosphere of the major planets. The difficulties related to the experimental studies enhance the importance of three-dimensional numerical simulations in these fields. For this reason the development and improvement of the numerical techniques is basic for this research.

Most of the numerical papers dealing with spherical geometries (see [1–4], among others), employ second-order time integrators to find periodic, quasi-periodic, or even chaotic attractors by direct numerical simulation (DNS). The exclusive use of time integrations is not sufficient to establish the origin of the laminar flows and their dependence on the parameters, specially in the case of subcritical or multicritical transitions. In these situations pseudo-arclength continuation methods [5–7], and the linear stability analysis of the time dependent solutions [8,9] provide powerful instruments to clarify the dynamics. In addition, the computation of Liapunov exponents is used to assess if a system is chaotic. All these tools have been historically applied to low-dimensional systems, but the increase in computational power, and the advances in numerical linear algebra have motivated their use in high dimensional systems. However, these techniques require to find accurate solutions, and, in finding these, higher order time integration methods may prove advantageous. In fact, even if one is only interested in low precision time evolutions, high-order methods can sometimes be more efficient than low-order ones.

* Corresponding author. Tel.: +34 934017981; fax: +34 934016090.
E-mail address: sanchez@fa.upc.edu (J. Sánchez).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.07.004

For the spatial discretization of the equations on the sphere, many of the above mentioned codes use pseudo-spectral methods based on spherical harmonics basis functions, which provide high-accurate solutions with relatively few grid points [10]. These methods are based on transformations from the spectral to the physical space [11]. The calculation of the quadratic terms, appearing in the truncated equations, is performed in the physical space.

For time integration, some authors [12,2,13], among others, use semi-implicit integration methods, namely, they treat the linear terms (except the Coriolis term) implicitly with a Crank–Nicholson scheme, and the non-linear and the Coriolis terms explicitly with an Adams–Bashforth method [14]. In [2], a mixed Euler and integrating factor technique is also considered. Others, as in [15] use a modified second-order accurate Runge–Kutta method. In all these methods the linear systems of equations to be solved at every step can be separated into spherical harmonic components, which can be solved independently, so that only a set of small linear systems must be solved at each time step. At high rotation rates, though, when the Coriolis force dominates over the viscous forces, stability considerations lead to extremely small time steps. To avoid this situation, the Coriolis term can be treated implicitly [14,15]. Then the equations for a given spherical harmonic of order m are coupled for all degrees l. This implies that a large block-banded linear system has to be solved for each order m. As it was described in [15], solving them by means of direct factorization methods becomes prohibitive if relatively high resolutions are employed.

In this paper we study the efficiency of several time integration methods, with the Coriolis term treated either as semi-implicit or as fully implicit, giving rise to the different algorithms presented. We show that (preconditioned) Krylov or Arnoldi iterative methods can be used to solve the Coriolis-implicit linear system, reducing considerably the amount of memory required.

The time integration methods we use are based on the backward differentiation formulae (BDF) implemented both with constant stepsize and variable stepsize and variable order (VSVO). The BDF are widely used in very stiff problems [16]. Being fully implicit methods, they are usually implemented with a Newton–Krylov method for solving the non-linear equations [17]. In real geophysical and astrophysical flows, however, the non-linear terms are very costly to evaluate, and thus, in our time integration codes, we apply semi-implicit backward differentiation-extrapolation (IMEX-BDF) formulae, where the non-linear terms of the equations are treated explicitly.

The rest of the article is organized as follows. In Section 2, the formulation of the problem and the spatial discretization of the equations are introduced. In Section 3, the time discretization schemes and some details of their implementation are described. Section 4 contains a comparison with a previous benchmark [13]. In Section 5 the results of the study are presented, that is, the efficiency of the time integrators is compared and analyzed, the differences between the constant stepsize methods are shown, and the study of the implicit and semi-implicit variable stepsize and variable order methods is reported. Finally, the paper closes in Section 6 with a brief summary of the main conclusions.

2. Mathematical model and spatial discretization

We consider the thermal convection of a spherical fluid shell differentially heated, rotating about an axis of symmetry with constant angular velocity $\boldsymbol{\Omega} = \Omega\mathbf{k}$, and subject to radial gravity $\mathbf{g} = -\gamma\mathbf{r}$, where γ is a constant and $\mathbf{r}$ the position vector. The mass, momentum and energy equations are written, in a rotating frame of reference with angular velocity $\boldsymbol{\Omega}$, using the same formulation and non-dimensional units as in [18]. The units are the gap width, $d = r_o - r_i$, for the distance, $\nu^2/\gamma\alpha d^4$ for the temperature, and d^2/ν for the time, r_i and r_o being the inner and outer radii, respectively, ν the kinematic viscosity, and α the thermal expansion coefficient.

The velocity field is expressed in terms of toroidal, Ψ, and poloidal, Φ, potentials

$$\mathbf{v} = \nabla \times (\Psi\mathbf{r}) + \nabla \times \nabla \times (\Phi\mathbf{r}). \tag{1}$$

Consequently, the equations for both potentials, and the temperature perturbation, $\Theta = T - T_c$, from the conduction state $\mathbf{v} = \mathbf{0}$, $T = T_c(r)$, with $r = \|\mathbf{r}\|_2$, are

$$\left[\left(\partial_t - \nabla^2\right)L_2 - 2E^{-1}\partial_\varphi\right]\Psi = -2E^{-1}\mathcal{Q}\Phi - \mathbf{r}\cdot\nabla\times(\boldsymbol{\omega}\times\mathbf{v}), \tag{2}$$

$$\left[\left(\partial_t - \nabla^2\right)L_2 - 2E^{-1}\partial_\varphi\right]\nabla^2\Phi + L_2\Theta = 2E^{-1}\mathcal{Q}\Psi + \mathbf{r}\cdot\nabla\times\nabla\times(\boldsymbol{\omega}\times\mathbf{v}), \tag{3}$$

$$\left(\sigma\partial_t - \nabla^2\right)\Theta - R\eta(1-\eta)^{-2}r^{-3}L_2\Phi = -(\mathbf{v}\cdot\nabla)\Theta, \tag{4}$$

where $\boldsymbol{\omega} = \nabla\times\mathbf{v}$ is the vorticity.

The parameters of the problem are the Rayleigh number R, the Prandtl number σ, the Ekman number E, and the radius ratio η. They are defined by

$$R = \frac{\gamma\alpha\Delta T d^4}{\kappa\nu}, \quad E = \frac{\nu}{\Omega d^2}, \quad \sigma = \frac{\nu}{\kappa}, \quad \eta = \frac{r_i}{r_o}, \tag{5}$$

where κ is the thermal diffusivity, and ΔT the difference of temperature between the inner and outer boundaries.

The operators L_2 and $\mathcal{Q}$ are defined by $L_2 \equiv -r^2\nabla^2 + \partial_r(r^2\partial_r)$, $\mathcal{Q} \equiv r\cos\theta\nabla^2 - (L_2 + r\partial_r)(\cos\theta\partial_r - r^{-1}\sin\theta\partial_\theta)$, (r,θ,φ) being the spherical coordinates, with θ measuring the colatitude, and φ the longitude. In non-dimensional units the conduction state reads $T_c(r) = T_0 + R\eta/\sigma(1-\eta)^2 r$.

Non-slip perfect thermally conducting boundaries $\Phi = \partial_r\Phi = \Psi = \Theta = 0$ are used.

A standard treatment of the spatial dependence of the equations is used, so we comment only on the basic points (see e.g., [12,2], for details). The functions $X = (\Psi, \Phi, \Theta)$ are expanded in spherical harmonic series up to degree L, namely

$$X(t,r,\theta,\varphi) = \sum_{l=0}^{L}\sum_{m=-l}^{l} X_l^m(r,t)Y_l^m(\theta,\varphi), \tag{6}$$

with $\Psi_l^{-m} = \overline{\Psi_l^m}$, $\Phi_l^{-m} = \overline{\Phi_l^m}$, $\Theta_l^{-m} = \overline{\Theta_l^m}$, $\Psi_0^0 = \Phi_0^0 = 0$ to uniquely determine the two scalar potentials, and $Y_l^m(\theta,\varphi) = P_l^m(\cos\theta)e^{im\varphi}$, P_l^m being the normalized associated Legendre functions of degree l and order m. The Eqs. (2)–(4) written for the complex coefficients become

$$\partial_t\Psi_l^m = \mathcal{D}_l\Psi_l^m + \frac{1}{l(l+1)}\left[2E^{-1}(im\Psi_l^m - [\mathcal{Q}\Phi]_l^m) - [\mathbf{r}\cdot\nabla\times(\boldsymbol{\omega}\times\mathbf{v})]_l^m\right], \tag{7}$$

$$\partial_t\mathcal{D}_l\Phi_l^m = \mathcal{D}_l^2\Phi_l^m - \Theta_l^m + \frac{1}{l(l+1)}\left[2E^{-1}\left(im\mathcal{D}_l\Phi_l^m + [\mathcal{Q}\Psi]_l^m\right) + [\mathbf{r}\cdot\nabla\times\nabla\times(\boldsymbol{\omega}\times\mathbf{v})]_l^m\right], \tag{8}$$

$$\partial_t\Theta_l^m = \sigma^{-1}\mathcal{D}_l\Theta_l^m + \sigma^{-1}l(l+1)R\eta(1-\eta)^{-2}r^{-3}\Phi_l^m - [(\mathbf{v}\cdot\nabla)\Theta]_l^m, \tag{9}$$

with boundary conditions

$$\Psi_l^m = \Phi_l^m = \partial_r\Phi_l^m = \Theta_l^m = 0. \tag{10}$$

The spherical harmonic coefficients of the operator $\mathcal{Q} = \mathcal{Q}^u + \mathcal{Q}^l$ are

$$[\mathcal{Q}^u f]_l^m = -l(l+2)c_{l+1}^m D_{l+2}^+ f_{l+1}^m, \quad [\mathcal{Q}^l f]_l^m = -(l-1)(l+1)c_l^m D_{1-l}^+ f_{l-1}^m, \tag{11}$$

$$\text{with} \quad D_l^+ = \partial_r + \frac{l}{r}, \quad c_l^m = \left(\frac{l^2-m^2}{4l^2-1}\right)^{1/2}, \quad \text{and} \quad \mathcal{D}_l = \partial_{rr}^2 + \frac{2}{r}\partial_r - \frac{l(l+1)}{r^2}. \tag{12}$$

In the radial direction, a collocation method on a Gauss–Lobatto mesh of $N_r + 1$ points is employed ($N_r - 1$ being the inner number of points).

The spherical harmonic coefficients of the non-linear terms in Eqs. (7)–(9) are obtained following [12]. The velocity and vorticity fields are computed first on a collocation mesh in the three coordinates (r,θ,φ) with the help of dealiased Legendre and fast Fourier transforms [19]. The cross product is computed on the mesh, and, finally, transformed to the spectral space. The computation of the coefficients of the non-linear terms of Eq. (9) requires the evaluation of the inner product $(\mathbf{v}\cdot\nabla)\Theta$ on the collocation mesh, and then to transform back to the spectral space.

The mode $m = l = 0$ is only nonzero for Θ, and the amplitudes for $m = 0$ are real. With these considerations, a large system of ordinary differential equations of dimension $N = (3L^2 + 6L + 1)(N_r - 1)$ must be integrated in time.

3. Time integration methods

In order to simplify the notation, Eqs. (7)–(9) are written in the form

$$\mathcal{L}_0\dot{u} = \mathcal{L}u + \mathcal{N}(u), \tag{13}$$

where $u = (\Psi_l^m(r_i), \Phi_l^m(r_i), \Theta_l^m(r_i))$ is the vector containing the values of the spherical harmonic coefficients at the inner radial collocation points, and $\mathcal{L}_0$ and $\mathcal{L}$ are linear operators including the boundary conditions. The former is invertible. It is the identity for Ψ_l^m and Θ_l^m, and the operator $\mathcal{D}_l$ for Φ_l^m (see the time derivatives in Eqs. (7)–(9)). The operator $\mathcal{L}$ includes the terms $\mathcal{D}_l\Psi_l^m, \mathcal{D}_l^2\Phi_l^m - \Theta_l^m$, and $\sigma^{-1}\mathcal{D}_l\Theta_l^m + \sigma^{-1}l(l+1)R\eta(1-\eta)^{-2}r^{-3}\Phi_l^m$ of Eqs. (7)–(9), respectively, in any of the schemes used, and part of the Coriolis terms to be specified below. The operator $\mathcal{N}$, which will be treated explicitly in the IMEX-BDF formulae, will always contain the non-linear, and the rest of the Coriolis terms.

Time integration methods look for approximations $u^n \approx u(t_n)$ to solutions of (13) at time levels $t_n = t_{n-1} + \Delta t_{n-1}$, $n = 1,2,\ldots,$. The IMEX-BDF formulae mentioned before are related to the backward differentiation formulae (BDF) [20,16]. These are collocation multistep methods, which obtain u^{n+1} from the previous approximations u^{n-j}, $j = 0,1,\ldots,k-1$, k being the number of steps in the formula, in the way we now describe. Consider first the interpolating polynomial $q_{n,k}$ of degree at most k, such that $q_{n,k}(t_{n-j}) = u^{n-j}$, for $j = -1,0,\ldots,k-1$, where the unknown u^{n+1} is included. The value of the latter is determined by requiring that $q_{n,k}$ satisfies the differential equation at $t = t_{n+1}$, that is, $\mathcal{L}_0\dot{q}_{n,k}(t_{n+1}) = \mathcal{L}u^{n+1} + \mathcal{N}(u^{n+1})$. In order to avoid solving a full non-linear problem at every step, in the IMEX-BDF formula [21] the term $\mathcal{N}(u^{n+1})$ is replaced by the extrapolated value

$$\mathcal{L}_0\dot{q}_{n,k}(t_{n+1}) = \mathcal{L}u^{n+1} + p_{n,k-1}(t_{n+1}), \tag{14}$$

where $p_{n,k-1}$ is the interpolating polynomial of degree at most $k-1$, such that $p_{n,k-1}(t_{n-j}) = \mathcal{N}(u^{n-j})$, for $j = 0,1,\ldots,k-1$. It is useful to express $q_{n,k}(t)$ as the sum $q_{n,k}(t) = q^0_{n,k}(t) + u^{n+1} l_{n,k}(t)$, where $l_{n,k}$ is the polynomial of degree at most k taking value 1 at t_{n+1}, and 0 at t_{n-j}, for $j = 0,1,\ldots,k-1$. Consequently $q^0_{n,k}$ coincides with $q_{n,k}$ at t_{n-j}, for $j = 0,1,\ldots,k-1$, and takes value 0 at t_{n+1}. Then, the linear system to be solved in order to find u^{n+1} can be expressed as

$$\left(\mathcal{I} - \frac{\Delta t_n}{\gamma_0(n)}\mathcal{L}_0^{-1}\mathcal{L}\right)u^{n+1} = \frac{\Delta t_n}{\gamma_0(n)}\mathcal{L}_0^{-1}p_{n,k-1}(t_{n+1}) - \frac{\dot{q}^0_{n,k}(t_{n+1})}{\dot{l}_{n,k}(t_{n+1})}, \tag{15}$$

where $\mathcal{I}$ is the identity operator, and $\gamma_0(n) = \dot{l}_{n,k}(t_{n+1})\Delta t_n$. The practical computation of $\gamma_0(n)$, $\dot{q}^0_{n,k}(t_{n+1})/\dot{l}_{n,k}(t_{n+1})$, etc. can be done systematically and easily by following ideas in [22, S III.5].

When the time steps are constant ($\Delta t_n = \Delta t$) the expressions (14) and (15) become simpler. For example, (15) becomes

$$\left(\mathcal{I} - \frac{\Delta t}{\gamma_0}\mathcal{L}_0^{-1}\mathcal{L}\right)u^{n+1} = \sum_{i=0}^{k-1}\frac{\alpha_i}{\gamma_0}u^{n-i} + \sum_{i=0}^{k-1}\frac{\beta_i\Delta t}{\gamma_0}\mathcal{L}_0^{-1}\mathcal{N}(u^{n-i}), \tag{16}$$

where the coefficients α_i, β_j and γ_0 do not depend on n, and are listed, for instance, in [6]. It is well known that, if the step number k satisfies that $k \leqslant 6$, and the time steps are constant or the ratios $\Delta t_n/\Delta t_{n-1}$ are kept adequately bounded (see e.g. [22, S III.3]), the BDF formulae are 0-stable, i.e., they propagate the errors in a stable way. If $k \geqslant 7$ they are unstable and useless for time integration. It is also known that for $k \leqslant 6$ the k-step formula is convergent of order k, that is, the errors $u(t_n) - u^n$ are $O((\max_{0\leqslant j\leqslant n-1}\Delta t_j)^k)$. Therefore the k-step formula is also termed the kth order formula. Their regions of absolute stability are shown in Fig. 1. A point $z = \lambda\Delta t$ is in the absolute stability region of a numerical time integration method, if when it is applied to the equation $y' = \lambda y$, $\lambda \in \mathbb{C}$, with fixed stepsize Δt, the numerical solution y_n is bounded as $n \to \infty$. For the BDF formulas, the regions are the exterior of the closed curves shown in Fig. 1a. Fig. 1(b) shows a detail of them close to the origin. Their knowledge helps to understand the behaviors described in Section 5.

Notice that contrary to the variable stepsize case (15), the matrix in the system to be solved at every step in (16) is constant for all n, being this property one of the main attractions of constant stepsize formulae. On the other hand, the main reason to use variable stepsizes is to maintain local (time integration) errors below a desired tolerance, with the aim of maintaining a given accuracy along all the time interval considered. Changing the stepsize can also be combined with using formulae of different orders (step numbers) k, while maintaining accuracy. In this way, the integration can be started with $k = 1$ (and small Δt_0), when the lack of previously computed values precludes the use of higher order formulae, and then increase the order (and the stepsize) as the integration advances and previous approximations u^{n-j} are available. The strategy of variable stepsize and variable order (VSVO) codes we have used is outlined in the following paragraph. Let us mention first that for the fixed stepsize codes, the starting values u^j, $j = 1,\ldots,k-1$ are obtained by time integration from t_{j-1} to t_j with a VSVO code and sufficiently small tolerances ε^a and ε^r (see (18) below for the meaning of these values).

As explained in [22, S III.7] the local error of the k-step BDF can be estimated as $\Delta t_n[\dot{q}_{n,k+1}(t_{n+1}) - \dot{q}_{n,k}(t_n)]$, so that, in a similar way, that of the k-step IMEX-BDF formula can be estimated as

$$\mathrm{EST}^n(k) = \Delta t_n\left[\mathcal{L}_0^{-1}(p_{n,k}(t_{n+1}) - p_{n,k-1}(t_{n+1})) - (\dot{q}_{n,k+1}(t_{n+1}) - \dot{q}_{n,k}(t_n))\right]. \tag{17}$$

The corresponding factor for the stepsize, i.e., the factor by which Δt_n is multiplied to obtain the next time stepsize Δt_{n+1}, is

$$f^n(k) = \min\left(f_{max}(k), \max_{i=1,\ldots,N}\left(\frac{\varepsilon^r|u_i^{n+1}| + \varepsilon^a}{\mathrm{EST}_i^n(k)}\right)^{1/(k+1)}\right), \tag{18}$$

where ε^a and ε^r are, respectively, the tolerances below which the absolute and relative values of the local (time discretization) errors are required, i indexes the components of the vectors, and $f_{max}(k)$ are upper bounds for the time step ratio $\Delta t_{n+1}/\Delta t_n$, which are used in practical implementations of the BDF methods to enhance their stability properties (see [22, S III.5]). They depend on the order k, and we have taken $f_{max}(k) = 5,4,3,2,1.6,1.4$ for $k = 1,\ldots,6$, after diverse numerical experiments.

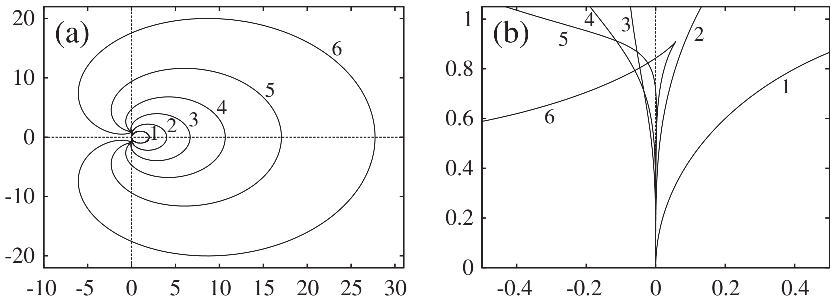

Fig. 1. (a) Absolute stability regions of the BDF up to order six with constant stepsize. The regions are the exterior of the closed curves. (b) Detail close to the origin.

If $f^n(k) < 1$, it is assumed that the local error is unacceptably large, and the computed value u^{n+1} is rejected and recomputed with a smaller stepsize, namely, $\Delta t'_n = 0.9 f^n(k)\Delta t_n$ (0.9 being a safety factor). If, on the contrary, $f^n(k) > 1$, u^{n+1} is considered sufficiently accurate, so that the integration can proceed to find a new approximation u_{n+2}. Before doing so, the estimations $\mathrm{EST}^n(k-1)$ and $\mathrm{EST}^n(k+1)$ of the local error of the $(k-1)$-step and $(k+1)$-step formulae are obtained, together with their corresponding factors $f^n(k-1)$ and $f^n(k+1)$. The value $k' = k-1, k$, or $k+1$ corresponding to the largest factor is chosen for the next step (always with k' not larger than a previously fixed maximum allowed order), and u_{n+2} is computed with the k'-step formula, and with stepsize equal to $\Delta t_{n+1} = 0.9 f^n(k')\Delta t_n$.

Whether the k-step IMEX-BDF formula is used with variable or constant stepsizes, the linear systems (15) or (16) have to be solved to compute u^{n+1}. In the sequel, we will represent these systems as

$$\mathcal{H}u^{n+1} = v^n, \tag{19}$$

$\mathcal{H}$ being the radial discretization of $\mathcal{I} - h\mathcal{L}_0^{-1}\mathcal{L}$ with $h = \Delta t_n/\gamma_0(n)$ or $h = \Delta t/\gamma_0$, and we comment now on their effective solution. We first notice that once v^n is evaluated, Eq. (19) decouples for each azimuthal wave number m, thus, at every time step, $L+1$ linear systems of the form

$$H^m U^m = V^m, \quad m = 0, \ldots, L \tag{20}$$

have to be solved. The vectors U^m and V^m contain, respectively, the unknowns and the right hand side of (19) with azimuthal wave number m. The structure of the matrices H^m depend on which terms of Eqs. (7)–(9) are included in the operator $\mathcal{L}$, that is, on which terms are treated implicitly (of course they depend also on the way the unknowns are ordered). We describe now the main features of these matrices for all the methods tested. They are summarized in Tables 1 and 2. The full details have been left to the Appendix A.

The inclusion in $\mathcal{L}$ of the diagonal parts of the Coriolis term (from now on referred to as $\mathcal{C}^d$) containing $im\Psi_l^m$ and $im\Phi_l^m$ (see Eqs. 7, 8), and of $\mathcal{Q}$ in $\mathcal{N}$, gives block-diagonal matrices H^m, whose memory requirements are $O(N_r^2(L+1)(L+2))$ due to the triangular truncation (6), namely, quadratic in the spherical harmonic truncation order. From now on, the time discretization with this treatment of the operators will be called the Q-explicit method.

Several authors (see [12,2], and [15] in the low rotation method section, among others) take the Coriolis term fully explicit in their codes, then the linear systems (19) decouple also for the spherical harmonic degree l, and all the H^m are equal. They have block-diagonal structure, each block depending on l. This means that matrices of dimension N_r^2 must be inverted using LU factorizations for each degree l. This approach is efficient in memory requirements since only a storage $O(N_r^2(L+1))$ is needed. However, at low Ekman and high Rayleigh numbers (those with realistic significance) very small time steps are needed to satisfy the CFL condition.

Taking into account that, in principle, the more implicit the method is, the larger the stepsize can be, and that, for the problem we are considering, the evaluation of the non-linear terms is expensive in CPU time, it could be computationally efficient to include other parts of the Coriolis term in the linear operator $\mathcal{L}$.

By adding $\mathcal{Q}^u$ or $\mathcal{Q}^l$ (see Eq. (11)) to $\mathcal{L}$, the matrices H^m become upper or lower block-Hessenberg matrices, respectively. They can be solved with the same memory requirements and number of operations as the Q-explicit method, because $\mathcal{Q}^u$ and $\mathcal{Q}^l$ must be evaluated in any case. The systems (20) are solved by using backward or forward block substitution, by inverting the same diagonal submatrices as in the Q-explicit method. In order to implement this possibility in a symmetric way, the two options are used alternately, that is, one step is performed with $\mathcal{Q}^u$ implicit and $\mathcal{Q}^l$ explicit, and vice versa in the following step. From now on, this time discretization will be called the Q-splitting method.

By setting $\mathcal{Q}$ totally implicit the operator $\mathcal{N}$ only includes the non-linear terms. Using this scheme the matrices H^m become block-tridiagonal. A direct block method for solving these linear systems involves a big amount of memory storage. A LU factorization requires the storage of block matrices with a memory size $O(N_r^2(L-m+1)^2)$ for each $m = 0, \ldots, L$, due to the change of the inner block structure during the Gaussian elimination. Therefore, the total amount of memory needed is cubic in the spherical harmonic truncation parameter L. The storage of these matrices becomes prohibitive for high resolutions when dealing with direct solvers [14,15]. However it will be seen that *matrix-free* methods based on Krylov

Table 1
Summary of methods tested for the equation $\mathcal{L}_0\dot{u} = \mathcal{L}u + \mathcal{N}(u)$. The second and third columns indicate which part of the Coriolis terms are treated implicitly (included in $\mathcal{L}$) and explicitly (included in $\mathcal{N}$), respectively. The operators $\mathcal{C}^d$, $\mathcal{Q}^l$ and $\mathcal{Q}^u$ represent, respectively, the diagonal part of the Coriolis terms, and the lower and upper parts of the operator $\mathcal{Q}$. The last column contains the symbols and lines used in Figs. 2–4 to label each method.

Method	Terms in $\mathcal{L}$	Terms in $\mathcal{N}$	Symbol, line
Q-explicit	$\mathcal{C}^d$	$\mathcal{Q}^l$and $\mathcal{Q}^u$	+, solid
Q-splitting	$\mathcal{C}^d$, $\mathcal{Q}^l$and $\mathcal{Q}^u$ alternately	$\mathcal{Q}^u$and $\mathcal{Q}^l$ alternately	$\times$, dotted
Q-implicit	$\mathcal{C}^d$, $\mathcal{Q}^l$ and $\mathcal{Q}^u$		*, dashed
Q-explicit VSVO	$\mathcal{C}^d$	$\mathcal{Q}^l$ and $\mathcal{Q}^u$	+, solid
Q-implicit VSVO	$\mathcal{C}^d$, $\mathcal{Q}^l$ and $\mathcal{Q}^u$		*, dashed
DLSODPK	fully implicit		$\circ$, dash-dotted

Table 2
Summary of solvers used for the linear systems with matrix $\mathcal{H} = \mathcal{I} - h\mathcal{L}_0^{-1}\mathcal{L}$ and $h = \Delta t_n/\gamma_0(n)$ or $h = \Delta t/\gamma_0$ (see Eqs. 15, 16). The second column indicates if the time step and the order are fixed or variable, the third indicates the linear solver, and the fourth the preconditioner used if the solver is iterative.

Method	Step and order	Linear solver	Preconditioner
Q-explicit	fixed	block-diag. LU	
Q-splitting	fixed	back and forward block substitution	
Q-implicit	fixed	GMRES	block-diag. LU
Q-explicit VSVO	variable	GMRES	block-diag. LU
Q-implicit VSVO	variable	GMRES	block-diag. LU
DLSODPK	variable	GMRES	block-diag. LU

techniques [23], GMRES [24] in our case, can be used efficiently, to solve the block-tridiagonal linear systems with the same memory requirements as in the *Q*-explicit case. The increase of the cost in solving the linear systems (20) may be offset by the increase of the time stepsize. The initial approximation for the solution of the linear system is obtained by extrapolation from the previous steps, as is usually done in the implementations of the BDF. From now on this method will be called the *Q*-implicit method.

The key point for using Krylov methods efficiently is to have a good preconditioner. It has to resemble as much as possible the original matrix, and the linear system with the preconditioning matrix must be easy to solve. We have used the block-diagonal, or the upper or lower block-triangular parts of H^m as preconditioners. All of them can be solved by an adapted direct LU method. We have concluded that adding the non-diagonal blocks does not improve significantly the convergence, and then only the results for the block-diagonal preconditioner will be reported.

As it is well known, time integration with a constant stepsize can be unnecessarily expensive because the stepsize has to be taken small enough to cope with possible fast transients. VSVO methods [22], though, do not suffer from this shortcoming. The price to pay is that the matrices of the linear systems to be solved at every step depend on the current time step, which changes during the integration. In order to avoid doing a massive number of factorizations, Krylov methods can be used with a preconditioning matrix depending on a fixed time stepsize Δt^*. At each step, the number of iterations needed by the iterative method will (partly) depend on how close the current Δt is to Δt^*. When the convergence of the iterative linear solver degrades the preconditioning matrix can be updated with the current time step. This is the strategy we have followed for all the VSVO algorithms employed (see the Appendix A for more details). In the case of the IMEX-BDF methods, the preconditioner is updated if the number of iterations of the linear solver exceeds a certain number (10 in all the calculations described).

All the semi-implicit methods described before have been implemented with constant stepsize, and with variable time stepsize and order using our own codes (except the *Q*-splitting VSVO method, which has been used only with constant stepsizes). From now on, the VSVO implementations of the *Q*-explicit and *Q*-implicit methods will be called *Q*-explicit VSVO and *Q*-implicit VSVO, respectively.

The last option considered is to take also the non-linear terms implicitly with a VSVO formulation of the BDF. This leads to the solution of a non-linear system of equations at each step. Due to the stiffness of the problem, it cannot be efficiently solved by fixed point iteration, and it is then solved by an inexact Newton's method. At each Newton's iteration, linear systems of similar structure to (19), but now including the Jacobian of the non-linear operator $\mathcal{N}$, have to be solved. The matrix products by the Jacobian are computed by finite differencing, and the linear systems are solved by a Krylov method. See [17] for further details. From now on this method will be called fully implicit method. We used the DLSODPK code of the ODEPACK package [17]. In this case the matrices depend not only on the current time step, but also on the current solution. As before, they can be preconditioned by the block-diagonal, or by the upper or lower block-triangular matrices computed with a fixed time step Δt^*. Again, a criterion for recomputing the LU factorizations must be established. In our implementation, the preconditioner is updated only if the ratio between the current Δt and Δt^* exceeds 10 or is smaller than 1/10.

Observe that the fully implicit method requires one evaluation of the non-linear terms per iteration of the linear solver, and a further one per iteration in Newton's method. Since, in this particular and other computational fluid dynamics problems the evaluation of the non-linear terms is expensive, semi-implicit are likely to outperform fully implicit methods despite the smaller stepsizes Δt they may be forced to take due to their poorer stability properties.

Finally it must be stressed that besides preconditioning techniques, the efficiency of Newton–Krylov methods can be improved by using weighted norms in all error-like vectors, i.e., using vectorial tolerances to control the convergence. This has proved useful in our calculations only for the fully implicit method (see [17] for details of the implementation). The weights we have used are given in Section 5.

Tables 1 and 2 summarize, for future reference, the main features of the methods tested, and the linear solvers and preconditioner used for each one.

4. Comparison with the benchmark

To check our new non-linear code we have compared our results with the non-magnetic data of [13], where several research groups present the results of a benchmark study for a convection-driven dynamo problem in a rotating spherical shell.

Although the benchmark deals with the convergence and accuracy of several spatial discretizations of the problem, and not with the time integration, we have used the same parameters because they are well known to the community of people working on convection in spherical geometry, and because the kind of computation we are interested in (computation of periodic orbits and their stability, for instance) are always close to regimes which are not yet chaotic. Therefore, a transient to a periodic regime such as that of [13] is a good test for our purposes.

The values of the parameters in [13] are $\eta = 0.35$, $\sigma = 1$, $E = 10^{-3}$ and $R_B = 100$, R_B being the Rayleigh number defined in [13]. Taking into account the different non-dimensional units of both formulations, it turns out that $R = (1-\eta)\sigma Ta^{1/2}R_B$, so $R = 65000$. In addition, the relation between the temperatures is $T = R\sigma^{-1}T_B$, T_B being the temperature scaled as in [13]. The initial condition recommended is velocity $\mathbf{v} = \mathbf{0}$, and temperature

$$T_B(r,\theta,\varphi) = \frac{r_i r_o}{r} - r_i + \frac{2A}{\sqrt{2\pi}}(1-x^2)^3 P_m^m(\theta)\cos m\varphi, \tag{21}$$

with $A = 0.1$, $x = 2r - r_i - r_o$, $m = 4$, and $P_m^m(\theta) = \sqrt{(2m+1)!!/2(2m)!!}\,\sin^m\theta$ the normalized associated Legendre function of order and degree m. By starting from it the solution tends, after an abrupt transient, to an azimuthal traveling wave of wave number $m = 4$.

In table 4 of [13] the authors provide the values and the estimated errors of some data corresponding to the reference solution, which can be considered that obtained with the pseudo-spectral code of Christensen, Wicht and Glatzmaier (CWG). They are the precession frequency ω, the mean kinetic energy density

$$E_{kin} = \frac{1}{2V_s}\int_{V_s} v^2\,dV, \tag{22}$$

V_s being the volume of the fluid shell, and the temperature T_B and the azimuthal velocity v_φ at the point $r = (r_i + r_o)/2$, $\theta = \pi/2$, whose φ-coordinate is given by $v_r(r,\theta,\varphi) = 0$ and $\partial_\varphi\, v_r(r,\theta,\varphi) > 0$. Their values are

$$\omega = 0.1824 \pm 0.0050, \quad E_{kin} = 58.348 \pm 0.050, \quad T_B = 0.42812 \pm 0.00012, \quad \text{and} \quad v_\varphi = -10.1571 \pm 0.0020. \tag{23}$$

We have integrated with the spatial resolution which corresponds to the code by CWG, i.e., with $N_r = 24$, and a triangular truncation of the spherical harmonics (6) with $L = 32$. The total number of equations is $N = 75095$. From now on, the test case with this resolution, and the parameters of the benchmark will be called C_1.

Most of the groups in [13] used a second-order Crank–Nicholson scheme for the implicit terms, and a second-order Adams–Bashforth method for the explicit ones. Therefore, to compare our solutions, it is enough to integrate using order $k = 2$. However we have checked that all the solutions described in Section 5, for the C_1 case, give values inside the intervals (23), although in the case of VSVO methods only when local error tolerances were taken smaller than 10^{-2}. These results are not surprising because our codes are based on the same spatial discretization than those of the CWG group, who obtained the benchmark reference data, and we use at least second-order time integrators.

5. Results

To compare the time integration schemes presented in Section 3 we integrate the C_1 case of the previous section, and a second test also with $\eta = 0.35$ and $\sigma = 1$, but with $E = 10^{-4}$, $R = 800000$, $m = 7$, $N_r = 48$ and $L = 63$ (from now on it will be called the C_2 case). The total number of equations is now $N = 577442$. The higher resolution is needed to properly solve the problem with a lower Ekman number. The solution tends now to a traveling wave with $m = 7$. In both tests $R \simeq 1.78R_c$ has been selected to be at the same relative distance to the critical Rayleigh number, R_c, at which the convection sets in.

Since at the initial condition (21) the norm $\|\partial_t u\|_2$ is extremely large, there is a strong initial transient. This produces large irregular errors when the equations are integrated with a constant stepsize method. Therefore the very initial transient has been discarded to make our comparisons, and all the test runs of the time-stepping codes are started with the same initial condition obtained after the solution has smoothed by a short time integration t_0. To obtain it, the Q-implicit VSVO method with very low tolerances has been used. Then the system (7)–(10) is evolved from the new initial condition at time t_0 to a final time t_f, at which the solution, $u(t)$, is close to the periodic regime. As the limit solution is a wave, $\|u\|_2$ tends to a constant. In the C_1 case, $t_0 = 0.053$ and $t_f = 0.253$, and, in the C_2 case, $t_0 = 0.0112$ and $t_f = 0.12$. The time t_0 has been selected as the first time at which $\|u(t_0)\|_2 \simeq \|u(t_f)\|_2$. After t_0, the norm of u increases until it reaches a maximum, and then finally decreases to a constant value close to $\|u(t_f)\|_2$.

To check the efficiency of the different schemes the relation between the relative error, and the run time in seconds, rt, is studied. The former is defined as

$$\varepsilon(u) = \frac{\|u - u_r\|_2}{\|u_r\|_2}, \tag{24}$$

where $u = u(t_f)$ is the solution we want to check, and $u_r = u_r(t_f)$ is an accurate reference solution obtained with the Q-implicit VSVO method, with tolerances for the relative and absolute errors set to $\varepsilon^r = \varepsilon^a = 10^{-14}$ and 10^{-13} in the C_1 and C_2 tests, respectively.

The decrease of the relative error (24) is achieved by decreasing the stepsize in the case of fixed stepsize methods, or by decreasing the tolerances for the local errors in the case of the VSVO methods. In our implementation of the semi-implicit

VSVO methods we demand $f^n(k)$ in (18) to satisfy $f^n(k) < 1$. In the test we show below we have taken $\varepsilon^a = \varepsilon^r$, since we checked that other choices of ε^a do not significantly alter the efficiency of the methods.

On the other hand, the fully implicit DLSODPK can use component-dependent tolerances ε_i^a for each component e_i of the estimated local error. In fact, if N is the size of the vector $u = (u_1, \ldots, u_N)^T$, DLSODPK requires

$$\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{|e_i|}{\varepsilon_i^r|u_i| + \varepsilon_i^a}\right)^2\right)^{1/2} < 1, \tag{25}$$

which is in general less demanding than having $f^n(k) < 1$ as in our semi-implicit methods. For the tests we present below, we have taken the ε_i^a proportional to ε^r in the following way: $\varepsilon_i^a = f_\psi \varepsilon^r$, $\varepsilon_i^a = f_\phi \varepsilon^r$ and $\varepsilon_i^a = f_\Theta \varepsilon^r$, for the blocks corresponding to the variables Ψ, Φ and Θ, respectively. The factors used are $f_\psi = 10$, $f_\phi = 1$, and $f_\Theta = 10^4$, which correspond to the orders of magnitude of the variables. We checked that the efficiency of the method was improved only marginally if other values of ε_i^a were used.

All the calculations have been performed on a cluster of Pentium IV personal computers using a single processor at 3.4 GHz, with 2 Mb of cache memory. The FFT routines used to evaluate the non-linear terms are those of the FFTW3 library [25]. The action of the linear operators required by the linear solvers, the computation of the non-linear terms, and the Legendre transforms have been implemented as matrix–matrix products using the GotoBLAS library [26] to increase the efficiency of the codes. The block structure of some of the matrices is used to minimize the number of operations. For instance, when matrix products by H^m in any of its variants are required, they are computed by using matrix–matrix subroutines for their sub-blocks (see the Appendix A). It is easy to see that each sub-matrix of dimension $N_r - 1$ of H^m always multiplies the real and imaginary parts of the same spherical harmonic coefficient of one of the functions Ψ, Φ or Θ. These products and any other for which it is possible, are grouped to optimize the memory transfers.

Fig. 2(a) and (b) show the dependence of $\log\varepsilon(u)$ against the logarithm of the constant time step Δt for the cases C_1 and C_2, respectively. The order of the methods can be seen in the slopes of the curves, and it is indicated beside. For every order k in Fig. 2(a) and (b), the largest values of Δt in each curve are those in the limit of (absolute) stability of the corresponding kth order formula, that is, the largest value in the increasing sequences of Δt, which we tried, that did not produced an overflow due to instability. The negative slopes at the left of the $k = 5$ curves correspond to the accumulation of rounding errors, which grows as Δt decreases. We have checked that, for very low Δt, the slope corresponds to a power -1, as is expected when the accumulated rounding error is proportional to the number of time steps, i.e., to Δt^{-1} (See e.g., [27, S 1.4–2]). The combination of the Crank–Nicholson and Adams–Bashforth schemes used by some authors would be comparable to the $k = 2$ case. Although the coefficient of the leading term of the local truncation error is lower for the Crank–Nicholson method (1/12)

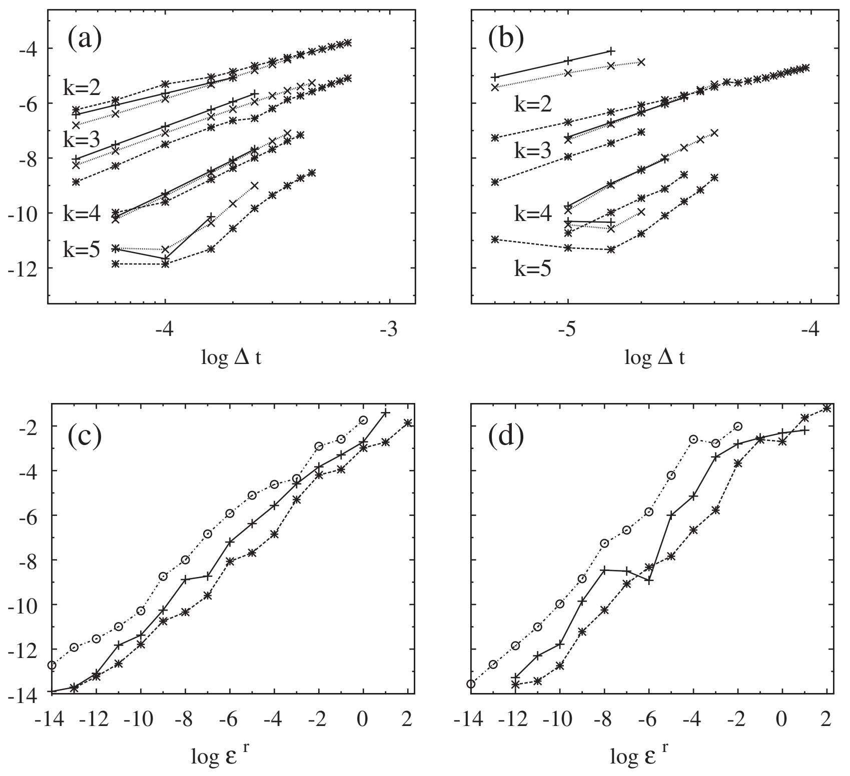

Fig. 2. (a) $\log\varepsilon(u)$ versus $\log\Delta t$ for constant stepsize integration, k from 2 to 5, in the C_1 case. (c) $\log\varepsilon(u)$ versus $\log\varepsilon^r$ for the VSVO methods in the C_1 case. (b), (d) Same as (a) and (c), respectively, for the C_2 case. The symbols mean: Q-explicit (+, solid line), Q-splitting (×, dotted line), Q-implicit (*, dashed line), Q-explicit VSVO (+, solid line), Q-implicit VSVO (*, dashed line) and DLSODPK (○, dash-dotted line). The vertical axis is the same in each row of plots.

than for the second-order BDF formula (2/9), the former requires an additional evaluation of the linear terms of the equations. Moreover if the Crank–Nicholson method is used with fixed time step, oscillations due to eigenvalues of large and negative real parts inherent to the scheme are not damped [28, S 3.6]. To solve this, usually, either an average between consecutive steps is done from time to time, or the first steps of the integration are performed with a smaller time-step size.

With respect to the size of the error for a given Δt, we notice both in Fig. 2(a) and (b) that the Q-implicit method is in general more accurate than the other two, more markedly so in Fig. 2(b). With respect to the largest stepsize the methods can use for a given order k, Fig. 2(a) shows that, in the C_1 case, the Q-implicit method can take larger stepsizes than the other two methods for all the orders. In the C_2 case, as the Ekman number is smaller, the $\mathcal{Q}$-term starts to be dominant, and, as expected, the relative differences of the upper limit of Δt among the methods are larger. Surprisingly, and as opposed to the C_1 case, for the orders $k = 3$ and $k = 4$ in the C_2 case, the Q-splitting method can use larger stepsizes than the Q-implicit. This may be partially explained as follows. For the smaller Ekman number, the eigenvalues λ of the $\mathcal{Q}$-term are close to the imaginary axis and their imaginary parts grow as E decreases (see [29,30,9]). Then, if Δt is not very small, the product $\lambda\Delta t$ is outside the absolute stability region of orders $k = 3$ and $k = 4$ (see Fig. 1(b)). This may render the Q-implicit unstable and, hence, limit the range of available stepsizes for these orders. Except for this two cases, the upper limit of Δt is lower for the Q-splitting method, and even lower for the Q-explicit. Notice also that in Fig. 2(b) the slope of the longest curve corresponds to the Q-implicit method with order $k = 2$. For this order, the stability region of the second-order BDF contains a larger portion of the imaginary axis, and this may allow the method to take significantly larger stepsizes.

Fig. 2(c) and (d) show the dependence of $\varepsilon(u)$ against ε^r for the VSVO methods (recall $\varepsilon^r = \varepsilon^a$). These two plots show that, for a given ε^r, the ratio of the error $\varepsilon(u)$ of DLSODPK over that of the Q-implicit VSVO method is, in average, $O(\sqrt{N})$. This is because our error control is much more restrictive than that of DLSODPK as was explained before. The figures also indicate the high accuracy of the reference solution, since in this plot we compare it with that obtained by a standard integrator, and $\varepsilon(u)$ can be made smaller than 10^{-12}. On the other hand, the fact that the errors of the Q-explicit are slightly larger than those of the Q-implicit method simply indicates that the latter is a more accurate method.

Fig. 3(a–d) contains the efficiency curves for all the methods considered. The logarithm of the relative error $\varepsilon(u)$ is plotted against the logarithm of the run time rt, with rt expressed in seconds. Fig. 3(a) and (b) show the results for the constant time stepsize methods of orders 2 to 5, and Fig. 3(c) and (d) those for the VSVO codes together with the constant time stepsize Q-splitting method for comparison purposes. As before, plots (a) and (c) correspond to the C_1 case, and (b) and (d) to the C_2 case. Each point in Fig. 3 has its corresponding one in Fig. 2, but notice that the smaller the values of Δt and ε^r in Fig. 2 the larger the corresponding run times in Fig. 3.

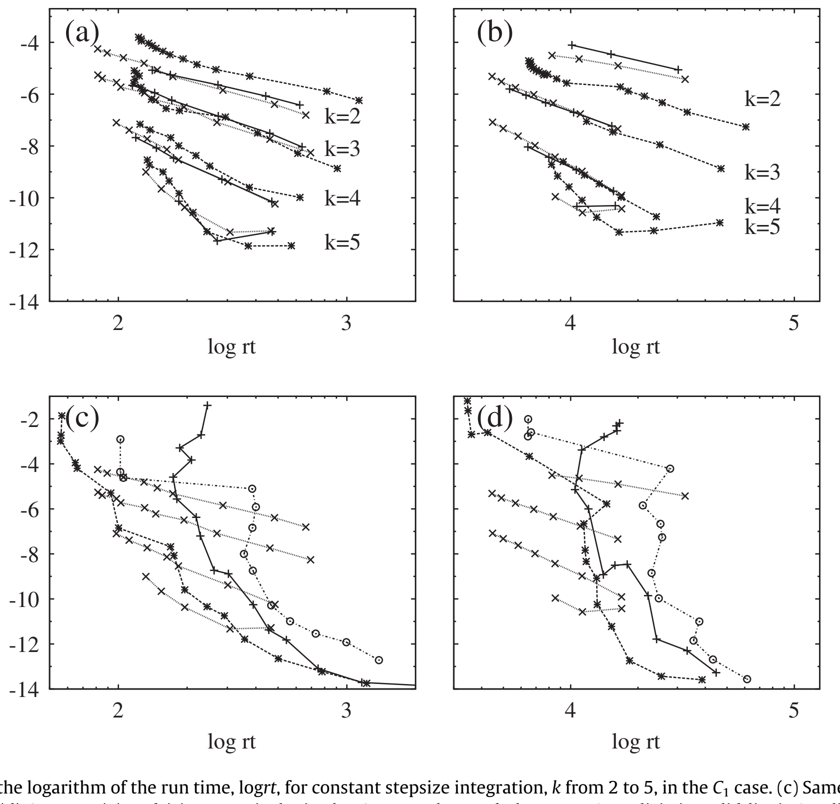

Fig. 3. (a) $\log\varepsilon(u)$ versus the logarithm of the run time, $\log rt$, for constant stepsize integration, k from 2 to 5, in the C_1 case. (c) Same as (a) for the Q-splitting and VSVO methods. (b), (d) Same as (a) and (c), respectively, in the C_2 case. The symbols mean: Q-explicit (+, solid line), Q-splitting (×, dotted line), Q-implicit (*, dashed line), Q-explicit VSVO (+, solid line), Q-implicit VSVO (*, dashed line), and DLSODPK (∘, dash-dotted line). The vertical axis is the same in each row of plots.

In order to understand the results, the computational cost of the most expensive processes involved in the time integration with constant stepsize has been considered in the formulae defined below. The initial integration with a VSVO method has not been included, but it is negligible because the integration time, $t_f - t_0$, was chosen to be large enough. The computational cost of a single step of the Q-explicit and the Q-splitting methods are essentially the same, so only the former will be described.

The run time of the integration with the Q-explicit and Q-implicit methods, during the interval $[t_0, t_f]$, and using time steps $\Delta t_{Q\ exp}$ and $\Delta t_{Q\ imp}$, respectively, can be estimated by

$$\begin{aligned} C_{Q\,exp} &= N_{S\,exp}(C_{NL} + C_Q + C_{LS}), \\ C_{Q\,imp} &= N_{S\,imp}(C_{NL} + (N_{GMR} + 1)(C_{MP} + C_{LS})), \end{aligned}$$

where C_{NL}, C_Q, C_{LS}, and C_{MP} are, respectively, the costs of the evaluation of the non-linear terms, the evaluation of the operator $\mathcal{Q}$, the solution of the block-diagonal linear systems (20) in the case of taking $\mathcal{Q}$ explicitly, and the matrix products by the block-tridiagonal matrices H^m when $\mathcal{Q}$ is taken implicitly. The integer N_{GMR} is the average number of iterations performed by the linear iterative solver (GMRES in our case). Each iteration involves a matrix product and an application of the preconditioner, that is, a solution with the block-diagonal matrix. An additional unit is added to N_{GMR} to take into account the initial evaluation of the residual, and the final preconditioner application. The numbers of steps are $N_{S\ exp} = (t_f - t_0)/\Delta t_{Q\ exp}$, and $N_{S\ imp} = (t_f - t_0)/\Delta t_{Q\ imp}$. As C_{NL} is the highest cost, the rest of them are written as a fraction of it as $C_Q = f_Q C_{NL}$, $C_{LS} = f_{LS} C_{NL}$, and $C_{MP} = f_{MP} C_{NL}$. In this way

$$C_{Q\,exp} = N_{S\,exp} C_{NL}(1 + f_Q + f_{LS}), \tag{26}$$

$$C_{Q\,imp} = N_{S\,imp} C_{NL}(1 + (N_{GMR} + 1)(f_{MP} + f_{LS})). \tag{27}$$

The cost C_{NL}, in seconds, and the factors f_Q, f_{LS}, and f_{MP} only depend on the spatial resolution employed, and not on the parameters of the problem. Their ranges of variation are $0.1 \leqslant C_{NL} \leqslant 3.42$, $0.043 \leqslant f_Q \leqslant 0.053$, $0.22 \leqslant f_{LS} \leqslant 0.33$, and $0.19 \leqslant f_{MP} \leqslant 0.33$ when the truncation parameters vary in the intervals $24 \leqslant N_r \leqslant 70$, and $32 \leqslant L \leqslant 90$. We have checked that the differences between the real run time, and the estimation obtained by the above formulae are so small that they model accurately the run time of the integrations.

Fig. 3(a) and (b) show that for approximately $\varepsilon(u) < 10^{-9}$, the $k = 5$ method is the most efficient. The corresponding curves cannot be extended to the left (higher Δt) due to stability reasons. If $\varepsilon(u) > 10^{-9}$ the most efficient methods vary from order 2 to 4 depending on the error required.

In the ranges of run time for which all the methods of a given order are available, and for a given run time, all the methods of the same order have similar efficiency, except for the case C_2 with $k = 2$ (see Fig. 3(b)).

For a given constant stepsize, the Q-explicit and the Q-splitting methods of all the orders have almost the same computational cost, and therefore the higher order methods should be preferred.

At first sight it could seem from (26)–(27), and the values of the factors given above, that the computational cost of the Q-implicit method is much higher than for the others, and that it does not depend on the order. However, as the order increases, the predictions of the solution at the end of each step, based on extrapolation using the order of the integrator, are better, and then the number of iterations N_{GMR} to solve the linear system during the corrections is lower. Moreover, the stepsize required for a given $\varepsilon(u)$ is larger, and then $N_{S\ imp}$ is lower than for the other methods. All this makes its computational cost comparable to the rest of constant stepsize methods.

In the low-cost regions (left of the plots), not all the methods are available due to the constraints of stability. If very small errors are not required, the Q-splitting method, with orders 2 and 3 in the C_1 case, and 3 and 4 in the C_2 case, is the most efficient of the constant stepsize methods. Determining which are the optimal order and stepsize requires some experiments. If many computations are going to be performed with a fixed spatial resolution, and in a relatively small region of parameters, it is worth making these initial experiments. As stated before the cost of the Q-explicit and the Q-splitting methods is essentially the same, but the latter has shown to be more stable, allowing for larger stepsizes, and hence, better efficiency.

In Fig. 3(c) and (d) the VSVO methods are compared with the Q-splitting of orders from 2 to 5, which is the best of the constant stepsize methods, except in some regions of very low error. The fully implicit method using DLSODPK is always more expensive than the Q-implicit VSVO methods because each iteration of the linear solver, and of the Newton's method requires an expensive evaluation of the non-linear terms. The Q-explicit VSVO method is also less expensive than DLSODPK, except for the higher $\varepsilon(u)$, for which the cost of the former increases due to the increase of the number of rejections, and of iterations of the linear solver. The factor f_{LS} indicates that an evaluation of the non-linear terms is between 3 and 4 times more expensive than a block-diagonal linear solving. Therefore DLSODPK will never be more efficient than the Q-implicit VSVO method, unless the former could take much larger time steps. Our numerical experiments suggest that this is not the case, especially for very low values of the Ekman number, where very rapid time scales must be resolved to maintain the accuracy.

Fig. 3(c) shows that the Q-implicit VSVO method is the best option for the C_1 case. Some constant stepsize methods can be more efficient, but at the price of doing some previous experiments to find out the optimal stepsize. In the C_2 case (Fig. 3(d)), there are more eigenvalues of larger imaginary part, and small and negative real part, as explained before. Therefore, in order to keep the local error below a given tolerance, the Q-implicit VSVO method is forced to use the BDF of order 2 in the left part of the curve shown in Fig. 3(d), and 5 in the right part, depending on the tolerance, as can be seen in Fig. 4(b), where the average order, k_{av}, is plotted versus the relative tolerance, ε^r, for the semi-implicit VSVO methods. The only curve with a jump is that corresponding to the Q-implicit VSVO method in the C_2 case. At the jump the order increases, the stepsize

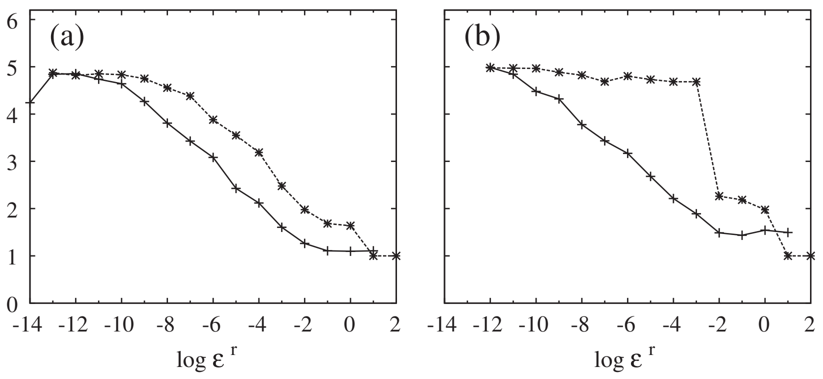

Fig. 4. (a), (b) The average order, k_{av}, versus the logarithm of the relative tolerance, $\log\varepsilon^r$, for the semi-implicit VSVO methods, and the C_1 and C_2 cases, respectively. The symbols mean: Q-explicit VSVO (+, solid line), and Q-implicit VSVO (*, dashed line). The vertical axis is the same for the two plots.

becomes larger, and the cost for a given error decreases. The jump reflects the shape of the stability regions of the BDF for constant stepsizes (shown in Fig. 1) which, as commented above, for orders 3 and 4 contain a smaller portion of the imaginary axis around to the origin. This abrupt transition between the two parts prevents the Q-implicit VSVO method from being as efficient as the Q-splitting method in the region of intermediate errors.

6. Conclusions

We have compared three semi-implicit time integration methods, both with constant stepsize and VSVO implementations (this last option only for two of them), and a fully implicit VSVO method. The semi-implicit methods differ among them in the treatment of the Coriolis term of the equation, which, at very low Ekman numbers (E), dominate the dynamics.

In view of the difficulties of treating the Coriolis term implicitly, it could be thought that writing the equations in an inertial frame of reference could be competitive. In this case there are two possibilities. The first consists in writing the solution of (13) as $u = u_h + u_{bc}$, u_h and u_{bc} satisfying, respectively, homogeneous and non-homogeneous boundary conditions, the latter coming from the solid-body rotation of the two spheres. The function u_{bc} can be expressed in terms only of the spherical harmonic Y_1^0 for the potential Ψ and would be proportional to E^{-1}. After substituting u into (13) some new linear terms would appear coming from the non-linear operators with a structure at least as complicated as that of the Coriolis terms. These new terms would be dominant for low E, and one would be again interested in treating them implicitly to cope with Courant restrictions of the time step. The second option consists in using non-homogeneous boundary conditions for u. This affects only the boundary conditions for the coefficient Ψ_1^0 of the spherical harmonic Y_1^0 in the expansion of Ψ. In this case E enters not only in the boundary conditions, but also hidden in the non-linear terms preventing the use of semi-implicit methods. There is another reason to avoid the equations in the inertial frame. The solutions after the first bifurcation from the conductive state are azimuthal waves of high azimuthal wave number, and drifting frequency ω usually positive and below that of the spheres Ω. Then, integrating in the inertial frame implies coping with the typical limitation of the time step of high oscillatory problems, no matter the order of the method employed (see [28, S 3.7]). In our case, if one is interested in computations close to this waves, it could be beneficial to write the equations for the deviations from the waves.

We have shown that when integrating the thermal convection equations of fast rotating fluid spherical shells, it is possible to handle the Coriolis term, and even the non-linear term, implicitly, if iterative Krylov methods are used to solve the corresponding linear systems. This is due to the low memory requirements of these iterative linear solvers. We have also shown that taking the Coriolis term implicitly results in a more efficient time integration for low values of the Ekman number. On the other hand, for the high resolutions needed to resolve the Ekman boundary layers, using adapted LU decompositions to solve the linear systems in semi-implicit or fully implicit methods is not viable, due to the prohibitive amounts of memory this would require.

We have checked our new time integration codes with a known benchmark [13]. We have found that to get relative errors of size 10^{-3}, of the order of those of the benchmark results, the most efficient methods are the fully implicit DLSODPK and the Q-implicit VSVO, with $\epsilon^a = 10^{-2}$.

The results show that high-order methods can be used not only to compute high-accurate solutions, but to obtain the same accuracy as with lower order ones but more efficiently. In practice, the most efficient method depends on the value of the Ekman number E, on the precision wanted, and even on the type of solution. For instance, if one is just interested in obtaining smooth solutions by DNS, the best choice is to implement a plain Q-explicit or, better, a third or fourth-order Q-splitting method. However, if the time integration is part of a continuation process, and/or one is interested in calculating the stability of the solutions, small time integration errors must be obtained. In this case, the Q-implicit VSVO method will probably be the most efficient option. Moreover, since the lowest run times correspond to the Q-implicit VSVO method with large tolerances, this method may also be useful for passing long uninteresting transients, where having a control of the time stepsize might be important.

It must be pointed out that, since the results obtained depend on the solution integrated, and in this case it was smooth, the constant stepsize methods have made accurate computations with relatively large stepsizes. However, had the selected test solution been irregular, including for instance repeated transients [31] or bursts [32], it is reasonable to assume that in order to obtain similar accuracy the constant stepsize methods would have had to use a very small step, and then the Q-implicit VSVO method would have been the most efficient method under any circumstances.

With regard to possible parallel implementations, the main difference between the methods is in the relative number of evaluations of the linear and non-linear terms. Evaluating or solving the linear terms is highly parallelizable and is not difficult to implement. The evaluation of the non-linear terms is completely different. Their parallelization is complex, mainly due to the triangular expansion in spherical harmonics. In the fully implicit method, actions of the full Jacobian are required, which imply evaluations of the non-linear terms or their linearizations. Therefore it would be more sensitive to any inefficiency in the parallelization process. The rest of the methods would experience very similar speedups if implemented in parallel because they require a similar number of evaluation of the non-linear terms.

Finally, the results we have obtained are also relevant in other situations, when the partial differential equations to be solved have linear diffusion terms, which can be treated implicitly because the resulting linear systems are relatively easy to solve, and other terms expensive to compute, and whose inclusion in a fully implicit scheme is impossible or very difficult.

Acknowledgments

The research of F. Garcia, M. Net and J. Sánchez has been supported by MEC-DGI project FIS2007-64993 and AGAUR-GENCAT project 2009SGR-67, and that of B. Garcia-Archilla by MEC-DGI project MTM2006-00847.

Appendix A

The structure of the matrices H^m appearing in Eq. (20) is detailed here. Recall that $\mathcal{H}u^{n+1} = v^n$, with $\mathcal{H} = \mathcal{I} - h\mathcal{L}_0^{-1}\mathcal{L}$ and $h = \Delta t_n/\gamma_0(n)$ or $h = \Delta t/\gamma_0$, is the system to be solved to obtain the solution at time t_{n+1}, u^{n+1}, once the right hand side of Eq. (15) or (16), v^n, has been computed. When this equation is separated into its azimuthal components the systems $H^m U^m = V^m$, $m = 0,\dots,L$ are obtained.

If the functions Ψ, Φ, and Θ are expanded in spherical harmonics as in Eq. (6), we define the vectors $U_0^0 = (\mathcal{R}\Theta_0^0)^\top$, $U_l^0 = (\mathcal{R}\Psi_l^0, \mathcal{R}\Phi_l^0, \mathcal{R}\Theta_l^0)^\top$, and $U_l^m = (\mathcal{R}\Psi_l^m, \mathcal{I}\Psi_l^m, \mathcal{R}\Phi_l^m, \mathcal{I}\Phi_l^m, \mathcal{R}\Theta_l^m, \mathcal{I}\Theta_l^m)^\top$ if $m \neq 0$, where $\mathcal{R}$ and $\mathcal{I}$ indicate real and imaginary parts, respectively. Then the vector of all amplitudes for a given order m, U^m, and the vector of all unknowns, u, are

$$U^m = (U_m^m, U_{m+1}^m, \dots, U_{L-1}^m, U_L^m)^\top, \quad \text{and} \quad u = (U^0, U^1, \dots, U^{L-1}, U^L)^\top.$$

Suppose now that all the linear terms, included $\mathcal{C}^d$, $\mathcal{Q}^l$ and $\mathcal{Q}^u$, are treated implicitly in the operator $\mathcal{L}$. With the ordering given to U^m, the matrix H_m has dimension $6(L - m + 1)(N_r - 1)$ and the block-tridiagonal structure

$$H^m = \begin{pmatrix} A_m^m & B_m^m & 0 & \dots & 0 \\ C_{m+1}^m & A_{m+1}^m & B_{m+1}^m & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ & & & & \\ 0 & 0 & \dots & 0 & B_{L-1}^m \\ 0 & \dots & 0 & C_L^m & A_L^m \end{pmatrix}, \tag{28}$$

where the blocks of dimension $6(N_r - 1)$, which also have sub-block structure, are

$$A_l^m = \begin{pmatrix} (A_l)^\psi & \alpha_l^m I & 0 & 0 & 0 & 0 \\ -\alpha_l^m I & (A_l)^\psi & 0 & 0 & 0 & 0 \\ 0 & 0 & (A_l)^\phi & \alpha_l^m I & (E_l)^\phi & 0 \\ 0 & 0 & -\alpha_l^m I & (A_l)^\phi & 0 & (E_l)^\phi \\ 0 & 0 & F_l & 0 & (A_l)^\Theta & 0 \\ 0 & 0 & 0 & F_l & 0 & (A_l)^\Theta \end{pmatrix}, \tag{29}$$

$$B_l^m = \begin{pmatrix} 0 & 0 & (B_l^m)^\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & (B_l^m)^\phi & 0 & 0 \\ (B_l^m)^\psi & 0 & 0 & 0 & 0 & 0 \\ 0 & (B_l^m)^\psi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{30}$$

and C_l^m has the same structure as B_l^m with $(B_l^m)^\phi$ substituted by $(C_l^m)^\phi$, etc. The sub-blocks of dimension $N_r - 1$ are

$$(A_l)^\psi = I - h(D_l)^\psi, (A_l)^\phi = I - h(D_l^{-1})^\phi(D_l^2)^\phi, (A_l)^\Theta = I - (h/\sigma)(D_l)^\Theta,$$
$$(B_l^m)^\psi = -\frac{(l+2)}{(l+1)}2hE^{-1}c_{l+1}^m(D_l^{-1})^\phi(D_{l+2}^+)^\psi, (B_l^m)^\phi = -\frac{(l+2)}{(l+1)}2hE^{-1}c_{l+1}^m(D_{l+2}^+)^\phi,$$
$$(C_l^m)^\psi = -\frac{(l-1)}{l}2hE^{-1}c_l^m(D_l^{-1})^\phi(D_{1-l}^+)^\psi, (C_l^m)^\phi = -\frac{(l-1)}{l}2hE^{-1}c_l^m(D_{1-l}^+)^\phi.$$

$(E_l)^\phi = h(D_l^\phi)^{-1}$, $F_l = -(h/\sigma)l(l+1)R\eta(1-\eta)^{-2}Diag(r_i^{-3})$, $\alpha_l^m = 2hE^{-1}m/l(l+1)$, and I is the identity matrix of dimension $N_r - 1$. The superscripts ψ, ϕ or Θ over a radial operator denote to which function they are applied. The corresponding boundary conditions are different and are included in the matrices.

The diagonals of H^m with blocks B_l^m and C_l^m are the discretization of the operators $\mathcal{Q}^u$ and $\mathcal{Q}^d$, respectively, and the matrices A_l^m include that of the rest of the linear terms. Therefore, the matrices H^m of Eq. (28) correspond to the Q-implicit or the Q-implicit VSVO methods, depending on the value of h. If all the non-diagonal blocks B_l^m and C_l^m are set to zero, the resulting block-diagonal matrices correspond to the Q-explicit or Q-explicit VSVO algorithms. If only B_l^m or C_l^m are set to zero, H^m are the lower or upper block-Hessenberg matrices of the Q-splitting method.

The LU decomposition is performed only for the diagonal blocks A_l^m, either to solve the block-diagonal or block-Hessenberg systems in the Q-explicit or Q-splitting methods, or to be used as preconditioners for the iterative solvers in the rest of algorithms (see Table 2). As can be seen in Eq. (29), the A_l^m can be separated in two sub-blocks of dimensions $2(N_r - 1)$ and $4(N_r - 1)$, for which a tailored block-LU decomposition is performed.

When the block-diagonal is used as preconditioner, a frozen value h^* is set instead of the exact h. In the case of the Q-implicit method with fixed time stepsize, Δt is constant and known in advance and then $h^* = h$. For the VSVO methods, the value of h^* is selected at the first step, and the corresponding block-LU decomposition is computed. The updating of h^* and the preconditioner are performed, as explained in Section 3, when the linear solver requires too many iterations to converge or h^* is very different from h, depending on the method employed. The number of updates during the integrations used to be very small. In most of our calculations it was below 10. The higher values corresponded to the higher tolerances for the local errors. In these cases the range of Δt used during the time evolution was wider and this forced more updates. Therefore, except for the calculations for which high-order methods are not required, the time spent in preparing the preconditioner was negligible.

References

[1] H. Kitauchi, K. Araki, S. Kida, Flow structure of thermal convection in a rotating spherical shell, Nonlinearity 10 (1997) 885–904.

[2] A. Tilgner, Spectral methods for the simulation of incompressible flow in spherical shells, Int. J. Num. Meth. Fluids 30 (1999) 713–724.

[3] M.S. Miesch, J.R. Elliot, J. Toomre, T.L. Clune, G.A. Glatzmaier, P.A. Gilman, Three-dimensional spherical simulation of solar convection. I. Diferential rotation and pattern evolution achieved with laminar and turbulent states, Astrophys. J. 532 (2000) 593–615.

[4] U. Christensen, Zonal flow driven by strongly supercritical convection in rotating spherical shells, J. Fluid Mech. 470 (2002) 115–133.

[5] H.B. Keller, Numerical solution of bifurcation and non-linear eigenvalue problems, in: P.H. Rabinowitz (Ed.), Applications of Bifurcation Theory, Academic Press, 1977, pp. 359–384.

[6] J. Sánchez, M. Net, B. García-Archilla, C. Simó, Newton–Krylov continuation of periodic orbits for Navier–Stokes flows, J. Comput. Phys. 201 (1) (2004) 13–33.

[7] J. Sánchez, M. Net, C. Simó, Computation of invariant tori by Newton–Krylov methods in large-scale dissipative systems, Physica D 239 (2010) 123–133.

[8] J. Sánchez, M. Net, J. Vega, Amplitude equations close to a triple-(+1) bifurcation point of D_4-symmetric periodic orbits in $O(2)$-equivariant systems, Discrete Contin. Dyn. Syst. -Ser. B 6 (6) (2006) 1357–1380.

[9] F. Garcia, J. Sánchez, M. Net, Antisymmetric polar modes of thermal convection in rotating spherical fluid shells at high Taylor numbers, Phys. Rev. Lett. 101 (19) (2008) 194501-1–194501-4.

[10] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, Spectral Methods in Fluid Dynamics, Springer, 1988.

[11] S.A. Orszag, Transform method for calculation of vector-coupled sums: Application to the spectral form of the vorticity equation, J. Atmos. Sci. 27 (1970) 890–895.

[12] G. Glatzmaier, Numerical simulations of stellar convective dynamos. I. The model and method, J. Comput. Phys. 55 (1984) 461–484.

[13] U. Christensen, J. Aubert, P. Cardin, E. Dormy, S. Gibbons, G. Glatzmaier, E. Grote, Y. Honkura, C. Jones, M. Kono, M. Matsushima, A. Sakuraba, F. Takahashi, A. Tilgner, J. Wicht, K. Zhang, A numerical dynamo benchmark, Phys. Earth Planet. Int. 128 (2001) 25–34.

[14] G. Glatzmaier, P. Roberts, A three-dimensional convective dynamo solution with rotating and finitely conducting inner core and mantle, Phys. Earth Planet. Int. 91 (1995) 63–75.

[15] R. Hollerbach, A spectral solution of the magneto-convection equations in spherical geometry, Int. J. Num. Meth. Fluids 32 (2000) 773–797.

[16] E. Hairer, G. Wanner, Solving Ordinary Differential Equations. II. Stiff and Differential–Algebraic Problems, Springer, 1991.

[17] A.C. Hindmarsh, ODEPACK, a systematized collection of ODE solvers, in: R.S.S. et al. (Ed.), Scientific Computing, North-Holland, Amsterdam, 1983, pp. 55–364.

[18] R. Simitev, F.H. Busse, Patterns of convection in rotating spherical shells, New J. Phys. 5 (2003) 97.1–97.2, doi:10.1088/1367-2630/5/1/397.

[19] J.P. Boyd, Chebyshev and Fourier Spectral Methods, Dover, New York, 1999.

[20] C.F. Curtiss, J.O. Hirschfelder, Integration of stiff equations, PNASUSA 38 (1952) 235–243.

[21] G. Karniadakis, M. Israeli, S.A. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 97 (1991) 414–443.

[22] E. Hairer, H.P. Norsett, G. Wanner, Solving Ordinary Differential Equations. I Nonstiff Problems, Second. Revised ed., Springer-Verlag, 1993.

[23] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS pub. company, New York, 1996.

[24] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 865–869.

[25] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, in: Proceedings of the IEEE 93 (2) (2005) 216–231, special issue on Program Generation, Optimization, and Platform Adaptation.

[26] K. Goto, R.A. van de Geijn, Anatomy of high-performance matrix multiplication, ACM Trans. Math. Softw. 34 (3) (2008) 1–25.
[27] P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, John Willey and Sons, 1962.
[28] U.M. Ascher, L. Petzold, Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations, SIAM, 1998.
[29] K. Zhang, On equatorially trapped boundary inertial waves, J. Fluid Mech. 248 (1993) 203–217.
[30] M. Net, F. Garcia, J. Sánchez, On the onset of low-prandtl-number convection in rotating spherical shells: non-slip boundary conditions, J. Fluid Mech. 601 (2008) 317–337.
[31] P. Kolodner, Repeated transients of weakly non-linear traveling-wave convection, Phys. Rev. E 47 (2) (1993) 1038–1048.
[32] G. Sullivan, T.S. Ahlers, Nonperiodic time dependence at the onset of convection in a binary liquid mixture, Phys. Rev. A 38 (6) (1988) 3143–3146.