Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 220 (2007) 824–838

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

An implementation of the exponential time differencing scheme to the magnetohydrodynamic equations in a spherical shell

Philip W. Livermore [*]

Department of Applied Mathematics, School of Mathematics, Leeds University, Leeds, West Yorkshire LS2 9JT, UK

Received 14 February 2006; received in revised form 24 May 2006; accepted 24 May 2006
Available online 12 July 2006

Abstract

Over the last decade there has been renewed interest in applying exponential time differencing (ETD) time stepping schemes to the solution of stiff systems. In this paper, we present an implementation of such a scheme to the fully spectral solution of the incompressible magnetohydrodynamic equations in a spherical shell. One problem associated with ETD schemes is the accurate calculation of the necessary matrices; we implement and discuss in detail a variety of different methods including direct computation, contour integration, spectral expansions and recurrence relations. We compare the accuracy of six different second-order methods in determining the evolution of a three-dimensional magnetic field under the action of a prescribed time-dependent flow of electrically conducting fluid, and find that for the timestep restriction imposed by the nonlinear terms, ETD methods are no more accurate than linearly implicit methods which have the significant advantage of being easier to implement. However, ETD methods are more readily extendable than those which are linearly implicit and will become much more advantageous at higher order.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Stiff system; Exponential time differencing; Spectral method; Magnetohydrodynamics

1. Introduction

Stiff systems of equations often arise when solving partial differential equations with spectral methods. These problems are characterised by having a large range of timescales: often the large-scale solution sought will vary much more slowly in time than small perturbations to it decay or disperse. Because of numerical stability constraints, explicit time stepping schemes require these (usually uninteresting) transients to be resolved, forcing the user to choose a small time step with a concomitant high computational cost. Larger time steps may be taken when using schemes designed for stiff problems [1] that are now only limited by accuracy, namely linearly implicit and a variety of nonlinear methods including exponential time differencing [2,3].

[*] Tel.: +44 113 343 5180.
E-mail address: livermor@maths.leeds.ac.uk.

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.05.029

Stiff problems arise in many areas where vastly different timescales all play a role, for example in reaction kinetics and in the modelling of electrical circuits; more importantly to the subject of this paper, they occur in the solutions of the Navier–Stokes and magnetic induction equations of magnetohydrodynamics (MHD). The stiffness in these latter problems is generated by the (second order) linear diffusive damping of flow velocity or magnetic field respectively, both of which occur on timescales scaling as the square of the lengthscales involved; small-scales therefore diffuse on timescales which are minuscule compared to the evolution of the large-scale solution. This discrepancy is exacerbated by the use of Chebyshev spectral methods which, although providing fast spatial convergence compared to other local methods (e.g. finite difference or finite element), generate spurious diffusive timescales of $O(N^{-4})$ where N is the spectral truncation parameter, compared to the true timescales of $O(N^{-2})$ [4, p. 237]. Methods suitable for stiff problems are therefore required for such diffusive terms, for explicit schemes would require prohibitively small time steps of size $O(N^{-4})$ for stability. The nonlinear terms in the equations are usually timestepped with an explicit scheme because any other choice would be computationally too expensive, in general involving iterative methods. The restriction on the time step from these terms, being determined by the CFL stability criterion, is not as severe as that on the stiff part and is typically $O(N^{-2})$ for non-periodic spectral methods.

One of the most sought after characteristics of a timestepping scheme is *A-stability*: the property that physically decaying solutions are numerically damped for any choice of time step. This is highly desirable for stiff problems since fast decaying perturbations would be damped even with time steps much longer than their lifetime. Linearly implicit schemes are restricted from having an order higher than two if A-stability is required (this is the second Dahlquist stability barrier, see e.g. [5]); despite their simplicity and frequent usage they are therefore not extendable to high order. A subset of these, the backwards differencing schemes are frequently used in stiff problems because although they may not be A-stable for orders greater than two, they do correctly damp non-oscillatory decaying perturbations (but not those which are oscillatory in general); they are however unstable for orders greater than six. Nonlinear schemes are not restricted by the Dahlquist Barrier and may be generalised to arbitrary order: the exponential time differencing (ETD) method treats the linear diffusion term exactly (and so is automatically A-stable); amongst others some further possibilities are the integrating factor (competitive in certain problems) and implicit Runge–Kutta methods (often costly to implement).

Over the last decade, ETD methods have made a resurgence in the literature [2,6–8] and in recent studies by Kassam and Trefethen [9] and Cox and Matthews [2] who solved a variety of 1D stiff equations, they were shown to be the most accurate of the range of methods implemented. Methods like ETD, based on the exact treatment of the linear terms, have not been popular to implement since they require the use of matrix exponentials, presenting some difficulties when the spatially discretised diffusion operator is not diagonal. However, even in cases where this discretised matrix is large and dense, various procedures exist to speed up the necessary computations, including Schur factorisation [10] or Krylov subspace methods [11]. The spectral discretisation adopted in this paper renders the discretised diffusion operator block banded and so each block can be treated individually; the computation of any such associated matrix exponentials are therefore relatively efficient. A further issue with ETD schemes is the accurate computation of other necessary matrices [9,12]. Various methods have been proposed including contour integration [9], recurrence relations [8] and spectral expansions [13]. These methods are implemented and compared in terms of their accuracy in Section 5.

In this paper, we present an ETD method suitable for solving the equations of magnetohydrodynamics. Since we focus ourselves on the details of the time stepping technique however, we only include a brief summary of the derivation of the two model equations that we are required to solve (which is now fairly standard, e.g. [14–16]) in Section 2. In Section 3 we introduce the ETD scheme along with some common alternatives and present an implementation for our model equations in Section 4. The problem of accurately computing the required matrices is addressed in Section 5 and in Section 6 we compare six time stepping techniques in the evolution of a 3D magnetic field solution.

2. The equations of magnetohydrodynamics

The equations of magnetohydrodynamics in an electrically conducting incompressible non-rotating medium may be written in the following non-dimensional form:

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{1}{R_\mathrm{e}}\nabla^2\mathbf{u} + \mathbf{u}\times(\boldsymbol{\nabla}\times\mathbf{u}) - \boldsymbol{\nabla}\left(p + \frac{1}{2}|\mathbf{u}|^2\right) + (\boldsymbol{\nabla}\times\mathbf{B})\times\mathbf{B} + \mathbf{F}, \tag{1a}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \frac{1}{R_\mathrm{m}}\nabla^2\mathbf{B} + \boldsymbol{\nabla}\times(\mathbf{u}\times\mathbf{B}), \tag{1b}$$

which are coupled with the conditions $\boldsymbol{\nabla}\cdot\mathbf{u} = \boldsymbol{\nabla}\cdot\mathbf{B} = 0$. In the above equations, the magnetic field energy is scaled with the kinetic energy, and the parameters $R_\mathrm{e} = UL/\nu$ and $R_\mathrm{m} = UL/\eta$ are the kinetic and magnetic Reynolds numbers, respectively, U and L being typical velocity and length scales and η and ν the magnetic and viscous diffusivities; the vector $\mathbf{F}$ represents any external forces. The domain of interest is a spherical shell: $r \in [r_i, r_o]$ where we will take $r_i = 1/2$, $r_o = 3/2$ and (r, θ, ϕ) are spherical polar coordinates.

Many codes have been developed to solve these equations, most of which represent the divergence-free flow and magnetic fields in toroidal–poloidal decomposition (e.g. [14–16]) and adopt a Crank–Nicolson timestepping scheme. A popular choice of spatial discretisation is in terms of spherical harmonics in solid angle and Chebyshev polynomials in radius. Such fully spectral methods converge to the unknown solution exponentially fast, deal well with (global) magnetic boundary conditions and avoid the stringent CFL criterion at the poles incurred by convergent grid points (see e.g. [4]). Other possibilities, particularly in the choice of radial discretisation have been employed by other authors; a comparison of such schemes to solve the magnetic induction equation is given in [17].

In the case of the velocity $\mathbf{u}$ (and analogously for the magnetic field $\mathbf{B}$), the discretisation of Eq. (1a) into toroidal–poloidal potential form and spatially into spherical harmonics in solid angle is given by

$$\mathbf{u} = \boldsymbol{\nabla}\times\left(\sum_{lm} T_l^m Y_l^m(\theta,\phi)\hat{\mathbf{r}}\right) + \boldsymbol{\nabla}\times\boldsymbol{\nabla}\times\left(\sum_{lm} S_l^m Y_l^m(\theta,\phi)\hat{\mathbf{r}}\right), \tag{2}$$

where the time dependent coefficients $S_l^m(r,t)$ and $T_l^m(r,t)$ are to be determined. The function Y_l^m is a spherical harmonic of degree l and order m, where $1 \leqslant l \leqslant L_\mathrm{MAX}$ and $l \geqslant m \geqslant 0$ where L_MAX is the angular truncation parameter, and $\hat{\mathbf{r}}$ is the unit position vector. By operating on Eq. (1b) with $\hat{\mathbf{r}}\cdot$ and $\hat{\mathbf{r}}\cdot\boldsymbol{\nabla}\times$ and on (1a) with $\hat{\mathbf{r}}\cdot\boldsymbol{\nabla}\times$ and $\hat{\mathbf{r}}\cdot\boldsymbol{\nabla}\times\boldsymbol{\nabla}\times$ (thus eliminating the pressure), the equations separate in spherical harmonics and can be reduced to the following model forms (see e.g. [16]):

$$\frac{\partial T_l^m}{\partial t} = \frac{1}{R_\mathrm{e}}\mathscr{D}_l T_l^m + F(\mathbf{u},\mathbf{B}), \tag{3a}$$

$$\mathscr{D}_l\frac{\partial S_l^m}{\partial t} = \frac{1}{R_\mathrm{e}}\mathscr{D}_l^2 S_l^m + G(\mathbf{u},\mathbf{B}), \tag{3b}$$

where $\mathscr{D}_l = \frac{\partial^2}{\partial r^2} - \frac{l(l+1)}{r^2}$ and F and G are nonlinear functions. In what follows we refer to (3a) as the *toroidal* equation and (3b) as the *poloidal* equation in the discretisation of (1a), although both the toroidal and poloidal components of (1b) are described by (3a) on replacing R_e by R_m. We note that the inductive term $\boldsymbol{\nabla}\times(\mathbf{u}\times\mathbf{B})$ although strictly linear in $\mathbf{B}$ couples together all the different spherical harmonic coefficients, and since its computation is the same as the formally nonlinear terms, we group it similarly into F.

To calculate the nonlinear functions F and G, we adopt a method similar to Glatzmaier [14], where we transform between spectral and physical space. We perform fast transforms in the longitudinal and radial directions, although dealiasing is only possible in longitude as in the radial direction, factors of $1/r$ appear that have no finite Chebyshev representation. In latitude we adopt a slow legendre transform; although alternative fast transforms are available (e.g. [18]) they are not competitive except at resolutions higher than we require. The boundary conditions are typically chosen to be non-slip or stress-free for the fluid and a matched condition for the magnetic field to an electrical insulator or perfect conductor [14,15].

3. Some schemes suitable for stiff problems

In this section, we consider some second order schemes suitable for solving Eq. (3a) which, on discretising with Chebyshev polynomials (as detailed in Section 4.1), may be written as

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \mathsf{M}\mathbf{v} + \mathbf{F}(\mathbf{v}, t), \tag{4}$$

where M is the discretisation of $\mathscr{D}_l/R_e$, $\mathbf{v}$ is the vector of unknown Chebyshev coefficients and $\mathbf{F}$ is the nonlinear term. The matrix M (for either fluid or magnetic boundary conditions) has eigenvalues which are real and negative, some of which are large in magnitude and represent decay on a timescale much shorter than that typical of the nonlinear term; this is therefore a cause of stiffness in the system.

3.1. Linearly implicit schemes

The simplest linearly implicit method of second order is the Crank–Nicolson (CN) scheme (also known as the Trapezium rule) which may be combined with an Adams–Bashforth scheme for the nonlinear terms and is defined as

$$\textbf{CNAB2}\quad \mathbf{v}_{n+1}-\mathbf{v}_n=\frac{h\mathsf{M}}{2}(\mathbf{v}_{n+1}+\mathbf{v}_n)+\frac{h}{2}(3\mathbf{F}_n-\mathbf{F}_{n-1}), \tag{5a}$$

where $h=t_{n+1}-t_n$ and $\mathbf{F}_n$ denotes the value of the nonlinear function vector $\mathbf{F}$ at time t_n. When $\mathbf{F}=\mathbf{0}$ this scheme is A-stable: any solution will always decay (with our assumptions on M), although it may not decay at the correct rate. In particular, the fastest decaying modes favour oscillatory behaviour [5,16], this being a problem especially for the spuriously high decay rates unavoidable with Chebyshev discretisations.

3.2. Integrating factor and ETD methods

Both integrating factor and ETD methods treat the linear part of Eq. (4) exactly (and so are necessarily A-stable), but differ in the assumptions used when handling the nonlinear part. We begin by multiplying Eq. (4) by the factor $\mathrm{e}^{-\mathsf{M}t}$ and integrating over one timestep:

$$\mathbf{v}_{n+1}=\mathrm{e}^{\mathsf{M}h}\mathbf{v}_n+\mathrm{e}^{\mathsf{M}h}\int_0^h \mathrm{e}^{-\mathsf{M}\tau}\mathbf{F}(\mathbf{v}(t_n+\tau),t_n+\tau)\,\mathrm{d}\tau, \tag{6}$$

which as it stands is *exact*. The matrix exponential is defined in the usual Taylor-series fashion and may be calculated in a myriad of different ways [19]. The integrating factor method now approximates the vector $\mathbf{F}\mathrm{e}^{-\mathsf{M}\tau}$ by an extrapolation from previous values in time; for example, the second order Adams–Bashforth approximation which assumes a linear fit using the values at t_n and t_{n-1} leads to

$$\textbf{IFAB2}\quad \mathbf{v}_{n+1}=\mathrm{e}^{\mathsf{M}h}\mathbf{v}_n+\frac{3h}{2}\mathrm{e}^{\mathsf{M}h}\mathbf{F}_n-\frac{h}{2}\mathrm{e}^{2\mathsf{M}h}\mathbf{F}_{n-1}. \tag{7}$$

This method generalises to arbitrary order and is equivalent to timestepping the transformed variable $\mathbf{w}=\mathrm{e}^{-\mathsf{M}t}\mathbf{v}$ which is clearly a good idea if the sought solution decays as $\mathrm{e}^{\mathsf{M}t}$, for then $\mathbf{w}=O(1)$. However, if the solution does not decay in this fashion which is generally true if $\mathbf{F}\neq\mathbf{0}$, then the introduction of the fast decay timescale into the scheme when it does not exist in the solution introduces large errors into the system [4, p. 268]. Additionally such schemes do not preserve fixed points of the equations that they discretise.

Exponential time differencing schemes approximate $\mathbf{F}$, instead of $\mathrm{e}^{-\mathsf{M}\tau}\mathbf{F}$, by polynomial extrapolation from previous values; this assumption does not introduce an unwanted fast timescale into the solution and similarly the scheme may be generalised to arbitrary order. There are many variants, but for the purposes of this paper we will use the second order Adams–Bashforth approximation of $\mathbf{F}$ which yields the ETD2 scheme of Cox and Matthews [2]:

$$\textbf{ETD2}\quad \mathbf{v}_{n+1}=\mathrm{e}^{\mathsf{M}h}\mathbf{v}_n+hE_1(h\mathsf{M})\mathbf{F}_n+hE_2(h\mathsf{M})\mathbf{F}_{n-1}, \tag{8}$$

where the functions E_1 and E_2 are defined as

$$E_1(z)=\frac{(1+z)\mathrm{e}^z-1-2z}{z^2},\quad E_2(z)=\frac{1+z-\mathrm{e}^z}{z^2}. \tag{9}$$

A further alternative, as used by Pisarenko et al. [20] and Tilgner [16] is to replace $\mathbf{F}$ in Eq. (6) by a constant. One optimal choice is that derived from the second order Adams–Bashforth approximation, being the linear interpolant at time $t_n+h/2$ of the nonlinear terms at times t_n and t_{n-1}:

$$\textbf{ETDC2}\quad \mathbf{v}_{n+1} = e^{\mathsf{M}h}\mathbf{v}_n + \frac{1}{2}(3\mathbf{F}_n - \mathbf{F}_{n-1})e^{\mathsf{M}h}\int_0^h e^{-\mathsf{M}\tau}\,d\tau = e^{\mathsf{M}h}\mathbf{v}_n + \frac{h}{2}(3\mathbf{F}_n - \mathbf{F}_{n-1})E_3(\mathsf{M}h), \tag{10}$$

where $E_3(z) = (e^z - 1)/z = E_1(z) + E_2(z)$. This scheme is second order and preserves fixed points.

4. An implementation of the ETD2 scheme

The ETD2 method may be directly implemented for the toroidal equation (3a) as it is already of the correct form. The only issue is how to discretise the operator $\mathscr{D}_l$ in a well defined way; as with most timestepping methods, the biggest problem is how to impose the two boundary conditions. In all cases we choose a spectral method: expanding in Chebyshev polynomials and formulating a scheme to determine their unknown coefficients. We could have chosen to solve the equations using a collocation method with the appropriate differentiation matrices [9,21] which would have produced a similar analysis. However, we find it easier to implement arbitrary linear homogeneous boundary conditions in the spectral case. An alternative Galerkin scheme, in which we expand in a basis that explicitly satisfies all boundary conditions is discussed in Section 4.2 and Appendix A.

4.1. Solving the toroidal equation with a Chebyshev-tau method

In order to solve Eq. (3a) we adopt an expansion in terms of Chebyshev polynomials:

$$T_l^m = \sum_{n=0}^{N} a_n T_n(f(r)), \tag{11}$$

where $f(r) = \frac{2r-(r_o+r_i)}{r_o-r_i}$ is the linear mapping of the physical domain $[r_i, r_o]$ to $[-1, 1]$ and T_n is a Chebyshev polynomial of degree n. One simple approach might be to form D_l^N, the discretisation of $\mathscr{D}_l$ within the basis of Chebyshev polynomials (up to a maximum degree N), each matrix element $[\mathsf{D}_l^N]_{ij}$ being the projection of $\mathscr{D}_l T_j(f(r))$ onto $T_i(f(r))$, efficiently computed using a quasi-dealiased transform. Substituting this matrix into the ETD method however turns out to be a poor idea as the eigenvalues of D_l^N are scattered over $\mathbb{C}$; in particular, eigenvalues with a positive real part arise that make the scheme unstable. The failure is due to the exclusion of the boundary conditions: they must be somehow incorporated into the matrices to keep the eigenvalues on the negative real axis.

A more sensible alternative is similar to those commonly used in differentiation matrix schemes in which boundary conditions can be implicitly incorporated into the discretisation [21,22]. We can formulate a spectral equivalent of this idea by forming the matrix product $\mathsf{D}_\tau = \mathsf{D}_l^{N\prime}\mathsf{E}$ where $\mathsf{D}_l^{N\prime}$ is the upper $N-1$ rows of (the $(N+1)^2$ matrix) D_l^N and E is an "extension matrix" of dimensions $(N+1)\times(N-1)$ which maps a vector of Chebyshev coefficients of length $N-1$ (i.e. of degree $N-2$) to one of length $N+1$ by determining the extra two coefficients through the boundary conditions. Multiplication by $\mathsf{D}_l^{N\prime}\mathsf{E}$, a square matrix of dimension $(N-1)^2$, is equivalent to taking a Chebyshev polynomial of degree $N-2$, extending it to degree N by enforcing the boundary conditions, taking the projection of the result of the operator $\mathscr{D}_l$ onto the space of Chebyshev polynomials but truncating the result to degree $N-2$. This method is easily generalised to any number of arbitrary homogeneous boundary conditions which are all essentially imposed by the matrix E. The ETD2 scheme formulated using $\mathsf{M} = \mathsf{D}_\tau/R_e$ is stable since D_τ has the correct eigenvalue spectrum.

Note that after each timestep we will have only calculated the first $N-1$ Chebyshev coefficients by the ETD2 scheme (as restricted by the dimensions of our matrix D_τ) of our solution. The boundary conditions are then used for a second time to determine the two remaining coefficients.

4.2. Solving the poloidal equation

Eq. (3b) is of a distinctly different type to Eq. (3a) because the time and space derivatives are mixed. In particular, we cannot simply invert the $\mathscr{D}_l$ operator, reducing its form to (3a) because in general we do not know the boundary conditions for $\mathscr{D}_l S_l^m$. We must also be careful not to lose the fourth order character of the equation by subsequent matrix manipulations, as we have four boundary conditions to impose.

The problem of mixed derivatives is dealt with in many different schemes for timestepping the Navier–Stokes equations; if the scheme retains all four spatial derivatives in each stage then we can simply substitute matrix rows which implement the equation to those implementing the boundary conditions (e.g. [15]). We avoid these problems however if we use a Galerkin scheme, expanding in terms of a basis comprising recombined Chebyshev polynomials which implicitly satisfies all boundary conditions; a differentiation matrix analogue has been formulated successfully [23]. Because the boundary conditions have been taken care of, we can merely multiply through by the inverse discretisation of $\mathscr{D}_l$ and (3b) becomes of the same form as (3a). However, we find that although our implementation is stable for solving (3a), it is unstable for (3b) unless the timestep used is very small, making it practically useless; the details are given in Appendix A. A similarly unexpected and unfortunate instability has been previously found by Hollerbach [15] when applying a predictor–corrector method to the same equations. Instead, we follow an alternative but more involved two step approach called the influence matrix method [16,24].

4.3. An influence matrix method for the poloidal equation

To solve Eq. (3b), we first remove the troublesome $\mathscr{D}_l$ operator acting on the time derivative by solving the second order spatial problem for ϕ:

$$\mathscr{D}_l\phi = G, \tag{12a}$$

$$\phi = \frac{\partial S_l^m}{\partial t} - \frac{1}{R_e}\mathscr{D}_l S_l^m. \tag{12b}$$

The four relevant boundary conditions are that S_l^m must vanish and satisfy a further mixed homogeneous condition at either end of the radial domain. Nothing is known about the behaviour of ϕ on either boundary. In continuous variables, the solution ϕ of (12a) will be the sum of a complementary function and a particular integral, the undetermined coefficients hidden in the complementary functions being fixed by the boundary conditions. We can proceed despite the unknown boundary conditions on ϕ by noting that any fictitious boundary conditions determine one particular integral. We can then write the solution of (12a) as

$$\phi = A\phi_1 + B\phi_2 + \psi, \tag{13}$$

where ψ is the particular integral with $\psi(r_i) = \psi(r_o) = 0$, and $A\phi_1$ and $B\phi_2$ are the two complementary functions of (12a), being $\phi_1 = r^{-l}$ and $\phi_2 = r^{l+1}$ (or any linear combination thereof). Determining ψ is numerically achieved by multiplying the discretisation of G by the matrix D_τ^{-1}, where D_τ is formulated using zero boundary conditions. In fact, to make the method numerically better conditioned we use $\phi_1 = (\alpha_1 r^{-l} + \alpha_2 r^{l+1})$, $\phi_2 = (\beta_1 r^{-l} + \beta_2 r^{l+1})$ where $\alpha_1,\alpha_2,\beta_1$, β_2 are chosen to make $\phi_1(r_i) = 0, \phi_1(r_o) = 1$; $\phi_2(r_i) = 1, \phi_2(r_o) = 0$. The boundary conditions (which are yet to be implemented) determine the values of A and B.

The second stage is to solve

$$\frac{\partial S_l^m}{\partial t} - \frac{1}{R_e}\mathscr{D}_l S_l^m = \psi + A\phi_1 + B\phi_2, \tag{14}$$

by applying Eq. (8) directly with zero boundary conditions. Since $A\phi_1$ and $B\phi$ are constant in time, their contribution to $S_l^m(t_{n+1})$ is given exactly by the ETD2 method as

$$hE_3(h\mathsf{D}_\tau/R_e)(A\mathbf{q}_1 + B\mathbf{q}_2), \tag{15}$$

where $\mathbf{q}_i$ are the quasi-dealiased (as r^{-l} has no finite representation) Chebyshev representations for ϕ_i. At the end of each timestep (noting that the zero conditions are already imposed by stage two) we determine A and B by imposing the remaining boundary conditions.

We note that the matrices required in the ETD2 scheme to timestep the nonlinear terms in Eq. (3b) (i.e. $\mathsf{D}_\tau^{-1}E_1(h\mathsf{D}_\tau/Re)$ and $\mathsf{D}_\tau^{-1}E_2(h\mathsf{D}_\tau/Re)$) are efficiently computed from those required to timestep equation (3a).

5. Calculating the ETD matrices accurately

As pointed out by Kassam and Trefethen [9] in their differentiation matrix implementation of ETD schemes, a computational problem arises when evaluating the necessary matrices, such as $E_1(h\mathsf{M})$, when the argument $h\mathsf{M}$ has small eigenvalues. This may either happen when following a transient in detail (i.e. $h \ll 1$), or when M contains eigenvalues close to zero, and may be elucidated by considering the scalar equation

$$E_1(z) = \frac{(1+z)e^z - 1 - 2z}{z^2}, \tag{16}$$

when z is small. As $z \to 0$ both the numerator and denominator are $O(z^2)$ but their ratio is $O(1)$. Consequently, large relative errors are introduced because any roundoff due to finite precision becomes magnified. The problem is demonstrated in Fig. 1(a) which shows the error in the calculation of $hE_1(h\mathsf{D}_\tau/R_m)$ as a function of h for $R_m = 1$, performed in *Matlab* double (8-byte) precision as short-dashed, compared to a higher precision (16-byte) Fortran calculation as the solid line. Both errors are computed relative to a high accuracy calculation described below which we take to be exact. As h decreases, it is clear that hE_1 does not behave as $O(h)$ as it should in either case, although the absolute error is fairly small being of $O(10^{-9})$ or smaller for the 8-byte calculation and $O(10^{-16})$ for the 16-byte. The accuracy problem however is exacerbated by using a higher value of R_m which shifts all the eigenvalues towards zero. In the same plot, the long-dashed line shows the same error as a function of R_m^{-1} for a typical time step of $h = 10^{-3}$, plotted on the same axes. For physically interesting values of $R_m = O(10^5)$, the absolute error is significant, being $O(10^{-4})$ which corresponds to a relative error of 10%.

In the above example and the rest of Section 5, we analyse different ways of computing $hE_1(h\mathsf{D}_\tau)$ where D_τ is the matrix discretisation of $\mathscr{D}_l$ (defined in Section 4.1) of size 19×19 associated with spherical harmonic degree $l = 1$ and poloidal electrically insulating boundary conditions. We determine the absolute error in our calculations as the matrix norm of the matrix difference relative to a high-precision calculation performed in the package *Maple* using 35 digits of accuracy, which we take to be exact for our purposes. An analysis of the computational cost of a similar range of numerical schemes is given in [25]. It is clearly unsatisfactory that the required matrices become unusably inaccurate for large R_m (or R_e) and small h. Furthermore, the problem is exacerbated in higher order schemes, since both the numerator and denominator in the high order equivalent of E_1 behave as $O(z^n)$ where n is the order of the method. The ETD methods are therefore not readily extendable to arbitrary order when computing the necessary matrices directly.

When we are required to calculate matrix exponentials, there are a variety of schemes available (see [19]). One of the best is a Padé approximation (order 6 is sufficient) coupled with scaling and squaring which is implemented in the *Expokit* package [26]; this is in fact identical to that used in the package *Matlab*.

5.1. Taylor series and recurrence

One simple solution to the problem of computing $E_1(z)$ might be to use a Taylor series for "small" values of z and the usual formula for "large values". While this might work well at both extremes, there is an ill-defined

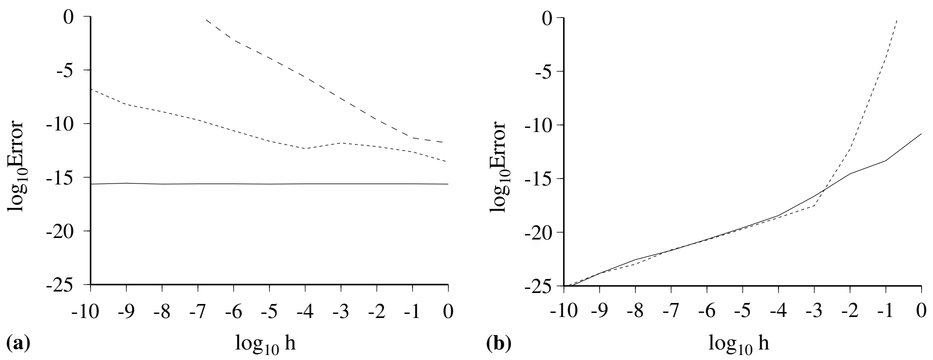

Fig. 1. The absolute error in the computation of $hE_1(h\mathsf{D}_\tau/R_m)$ as a function of h for $R_m = 1$; (a) shows the accuracy of Matlab double precision (8-byte) as short-dashed, Fortran quadrapolar precision (16-byte) as solid and the error as a function of R_m^{-1} with $h = 10^{-3}$ plotted on the same axes as long-dashed. (b) shows the accuracy of the recurrence relations of Beylkin et al. as solid and the Beylkin–Hochbruck relations as dashed.

grey area in the middle where neither method works well [9]. An alternative, following the suggestion of Beylkin et al. [8] is to rescale z so that it is suitably small – a Taylor series with (say) seven terms therefore being an excellent approximation, and then using recurrence relations to rescale the result. If we define the quantities

$$Q_0(z) = \mathrm{e}^z, \quad Q_1(z) = \frac{\mathrm{e}^z - 1}{z}, \quad Q_2(z) = \frac{\mathrm{e}^z - 1 - z}{z^2}, \tag{17}$$

then $E_1(z) = Q_1(z) + Q_2(z)$ and we may use the following recurrence:

$$Q_2(2z) = \frac{(Q_1(z)^2 + 2Q_2(z))}{4}, \tag{18a}$$

$$Q_1(2z) = \frac{(1 + e^z)}{2} Q_1(z). \tag{18b}$$

In the matrix case, we choose the initial rescaling to be $h\mathsf{D}_\tau/2^k$ where k is an integer chosen such that the matrix has a maximum singular value of at most 1. An alternative suggested by Hochbruck et al. [6] uses instead the formula

$$Q_1(2z) = \frac{Q_1(z)}{2}(zQ_1(z) + 2), \tag{19}$$

in place of (18b). This scheme has the property that we need never compute a matrix exponential; indeed, the Beylkin et al. relations may be similarly formulated if we make use of the identity $Q_0(2z) = Q_0(z)^2$ in Eq. (18b). Fig. 1(b) shows the error as a function of h for these two schemes: that based on (18a) and (18b) as solid and that based on (18a) and (19) as dashed. Both exhibit the proper $\mathrm{O}(h)$ behaviour as $h \to 0$ but the dashed curve becomes unstable at $h = \mathrm{O}(1)$ leading to large errors. The relations (18a) and (18b) work extremely well for every h considered.

5.2. Contour integration

An alternative and elegant method was proposed in Kassam and Trefethen [9] which is based on contour integration in the complex plane using the identity

$$E_1(z) = \frac{1}{2\pi \mathrm{i}} \oint_C \frac{E_1(t)}{t - z}\, \mathrm{d}t, \tag{20}$$

where the contour C must contain z in its interior and $\mathrm{i}^2 = -1$. For matrices, a similar form exists:

$$E_1(\mathsf{D}_\tau h) = \frac{1}{2\pi \mathrm{i}} \oint_C \frac{E_1(t)}{t\mathsf{I} - \mathsf{D}_\tau h}\, \mathrm{d}t, \tag{21}$$

where C must contain all the eigenvalues of $\mathsf{D}_\tau h$ and I is the appropriately sized identity matrix. The beauty of such a method lies in the fact that we may choose the contour to be sufficiently far from the origin that the numerical instability is eradicated: the troublesome part of the integrand $E_1(t)$ is never evaluated for small $|t|$. Such integration can be efficiently evaluated by use of the trapezium rule; additionally, symmetry may be exploited if we choose the quadrature points to be symmetric with respect to the real axis: we need only integrate in the upper (or lower) half plane and take the real part, since we know that the result must be real. Different shaped contours are investigated by Kassam [12] and we try some of the ideas below; two examples are shown in Fig. 2(b).

We will consider elliptical contours symmetric with respect to the real axis (of which a circle is a particular case) that can be parameterised as

$$t = x_0 + A\cos\theta + \mathrm{i}B\sin\theta, \tag{22}$$

where x_0 is its centre. If m quadrature points in the upper half plane are chosen to be $t_j = x_0 + A\cos\theta_j + \mathrm{i}B\sin\theta_j$, $j = 1, \ldots, m$, where $\theta_j = (2j - 1)\pi/2m$, then $E_1(\mathsf{D}_\tau h)$ may be approximated by $E_1^m(\mathsf{D}_\tau h)$ where

$$E_1^m(\mathsf{D}_\tau h) = \frac{1}{m}\Re\sum_{j=1}^{m}(t_j\mathsf{I} - \mathsf{D}_\tau h)^{-1}E_1(t_j)(B\cos\theta_j + \mathrm{i}A\sin\theta_j). \tag{23}$$

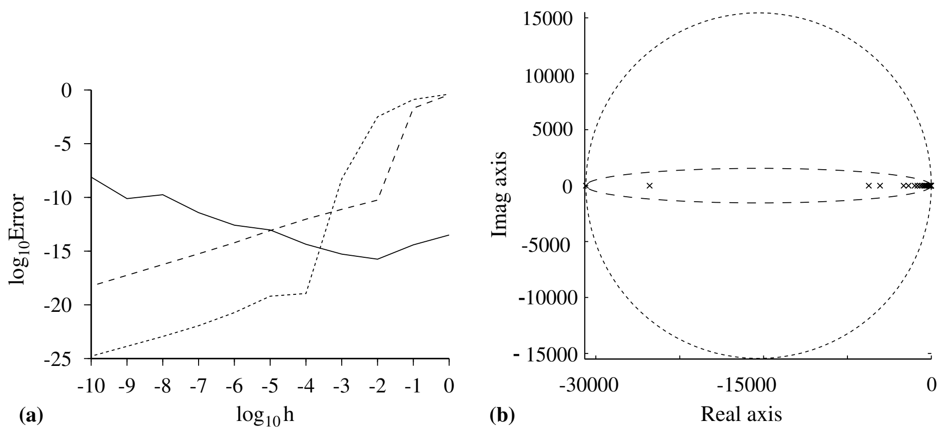

Fig. 2. Plots showing the accuracy of $hE_1(h\mathsf{D}_\tau)$ as a function of h where D_τ is a 19×19 discretised diffusion matrix as described in the text. (a) shows a variety of contour methods: solid represents circular contours around each eigenvalue in the diagonalised system, short-dashed is the large circular contour and long-dashed is the large ellipse; the latter two contours which both enclose all the eigenvalues are shown in (b) for $h = 1$ with eigenvalues of D_τ marked on as crosses. The most negative eigenvalue of D_τ is approximately -3×10^4.

One simple application of the method is to diagonalise $\mathsf{D}_\tau h$ and use the scalar version of Eq. (23) on each eigenvalue separately. We can write $E_1(\mathsf{D}_\tau h) = VE_1(\Lambda)V^{-1}$ where V is the matrix containing the eigenvectors of $\mathsf{D}_\tau h$ as columns and Λ is the diagonal matrix of the eigenvalues. A suitable suite of contours centred on each (real) eigenvalue $q < 0$ are circles of radius $\min(-q/2, 1)$ (which avoids the origin). Fig. 2(a) shows the error of $hE_1(\mathsf{D}_\tau h)$ as a function of h (solid curve) for this method which shows a similar lack of accuracy as $h \to 0$ as the 8-byte direct calculations. The error will get worse with increasing size of D_τ as the added complication of computing the eigenvalues accurately becomes progressively more difficult.

Instead, we consider two ellipses centred on $x_0 = \frac{1}{2}\lambda$ where λ is the most negative eigenvalue of $\mathsf{D}_\tau h$, $A = -x_0 + 1$ and we choose $B = A$ and $B = A/10$. These contours are shown in Fig. 2(b) with the eigenvalues marked as crosses. This choice of A and x_0 ensures that the contours enclose all the eigenvalues and do not pass through the origin: in fact they cross the real axis at $t = 1$ and $t = \lambda - 1$. Fig. 2(a) shows their accuracy in determining $hE_1(h\mathsf{D}_\tau)$ for various values of h all evaluated with $m = 256$; short dashed shows the circular contour, long dashed the ellipse. This value of m is vastly more than is required ([9] use $m = 32$) but with such a high value we can be sure that the error is due to the choice of method and not to the truncation in m. As will be noted immediately, these contour methods behave very well as $h \to 0$, the circular contour being more accurate for most values of h shown, but have large errors as h approaches $O(1)$. The reason why these fail seems to be linked to the increased size of the contour which is required to enclose all the eigenvalues. The flattened ellipse offsets this effect a little because the magnitude of $|t|$ on the contour is lower; however, it too suffers catastrophically at large h. This problem will only get worse as the truncation is increased, since the most negative eigenvalue scales as $O(N^4)$ and the contours become concomitantly larger.

5.3. An extension: keeping the contour small

To avoid the numerical imprecision when using large contours, an alternative approach as suggested by Kassam [12] is to use an ellipse which is only defined (for example) on the interval $[-41, 1]$, that is, centred at -20 and having $A = 21$, $B = 2$ (say). However, such a method which misses out many of the eigenvalues is only accurate when their contribution is negligible; while this is true for the exponential function, it is not the case for $E_1(z)$. This is illustrated in Fig. 3 by the solid line, where we show the error of $hE_1(\mathsf{D}_\tau h)$ as a function of h using this "small" ellipse. For values of $h \ll 1$ all the eigenvalues of $h\mathsf{D}_\tau$ lie inside $[-41, 1]$ and so the method is accurate; however, as h increases, some of the eigenvalues begin to be missed out and the error increases dramatically. The number of missed eigenvalues will grow as the truncation becomes larger and the method becomes increasingly inaccurate.

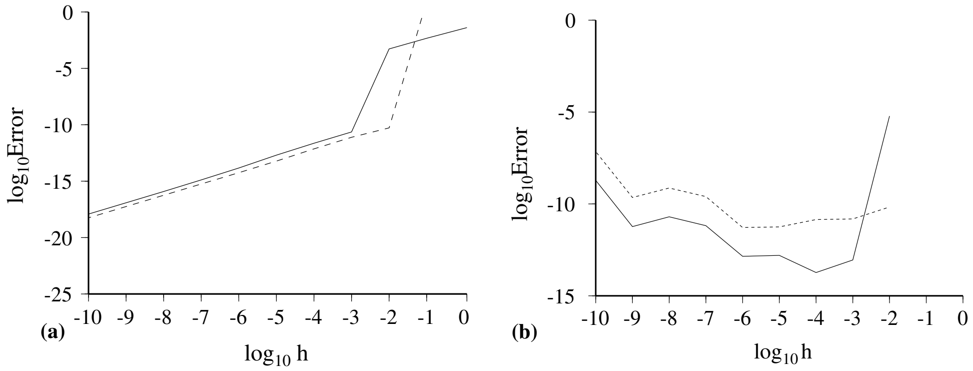

Fig. 3. The accuracy of the computation of $hE_1(\mathsf{D}_\tau h)$ using (a) the contour of an ellipse defined over $[-41, 1]$ shown as the solid line, and the ellipse defined over (approximately) $[-1,0]$ with respect to the rescaled variable w as dashed; (b) the accuracy of the Chebyshev expansion method with maximal degrees of 50 (solid) and 500 (dashed).

A further alternative to keep the contour small is to consider the linear rescaling $w = \epsilon t$ where $\epsilon = -1/\lambda > 0$ under which we can rewrite (21) as

$$E_1(\mathsf{D}_\tau h) = \frac{1}{2\pi i}\oint_C (w\mathsf{I} - \epsilon \mathsf{D}_\tau h)^{-1} E_1(w/\epsilon)\, \mathrm{d}w, \tag{24}$$

where C is chosen to enclose the eigenvalues of $\epsilon\mathsf{D}_\tau h$ that lie only in the interval $[-1,0)$. As usual, we must be careful to avoid the origin by a sufficient margin, although to pass too far into the right half plane would cause other problems since then $w/\epsilon \gg 1$ and severe roundoff error would result due to the dominating exponentials when computing $E_1(w/\epsilon)$. A contour on which the real part of w/ϵ is suitably bounded is the ellipse centred on $-1/2$ with $A = 10\epsilon + 1/2$, $B = A/10$; a plot of the accuracy is shown in Fig. 3(a) as a dashed line. This method again fails as h becomes large, and in any event the notion of being "close" but not "too close" to the origin is rather vague, probably case specific and is best avoided.

5.4. A Chebyshev expansion method

A further alternative method, following [13], is based on the Chebyshev expansion of the exponential function. If the eigenvalues of $h\mathsf{D}_\tau$ lie in the interval $[-2R, 0]$, then writing $z = (w-1)R$ where $-1 \leqslant w \leqslant 1$ we may write

$$e^z = \sum_{k=0}^{\infty} b_k T_k(w(z)), \quad b_k = \mathrm{e}^{-R} c_k I_k(R), \tag{25}$$

where I_k is a modified Bessel function of order k [27] and $c_0 = 1$, $c_k = 2, k \geqslant 1$. This relation may then be applied directly to the computation of $\mathrm{e}^{h\mathsf{D}_\tau}$, forming the matrices $T_k(h\mathsf{D}_\tau)$ by the standard three-term recurrence relation in degree satisfied by Chebyshev polynomials. The coefficients for the expansion of $E_1(z(w)) = \sum_k d_k T_k(w(z))$ can be immediately found by solving the pentadiagonal system for d_k, at some maximal truncation of degree K:

$$(1 + R(w-1))\sum_k b_k T_k(w) - 1 - 2R(w-1) = R^2(w-1)^2 \sum_k d_k T_k(w), \tag{26}$$

where we make use of the identity $(w-1)^2 T_k(w) = 1/4(T_{k-2}(w) - 4\,T_{k-1}(w) + 6T_k(w) - 4T_{k+1}(w) + T_{k+2}(w))$. Fig. 3(b) shows the error as a function of h for two different truncations $K = 50$ (solid) and $K = 500$ (dashed). The main difficulty with this method is the computation of $I_k(R)$ when R is large; the problem is hinted at in the behaviour of the solid curve at $h = 10^{-2}$ when the modified Bessel function series does not converge sufficiently. Indeed, for $h \geqslant 10^{-1}$, standard double precision routines cannot compute $I_k(R)$ as it is too large: they return an "Inf" error code (hence the absence of such points from the figure). For most of the values of h

shown, interestingly the computation with a smaller truncation is more accurate, although neither line shows the O(h) behaviour required for small h. This method is therefore not competitive as it neither shows sufficient accuracy at small h nor can it be used with large time steps.

5.5. Summary of methods

The most accurate methods of those we have implemented in determining $hE_1(\mathsf{D}_\tau h)$ are the recurrence relations of Beylkin et al. [8] and the Fortran 16-byte direct calculation. The latter method has the problem that it is computationally slow at least on 32-bit computers and is not supported by all current compilers. The best method therefore is that of the recurrence relations, which may be coded in the same 8-byte precision as the rest of the code. We note though that of the two recurrence relations implemented, one was unstable for large timesteps. We therefore caution against using such relations to calculate general ETD-required quantities without first testing their accuracy and stability; indeed, the accuracy exhibited here could well be case specific. Despite their elegance, neither the integration nor the spectral expansion methods are sufficiently accurate when h becomes large and are therefore not of practical use.

6. A comparison of time stepping methods

In this section, we compare the accuracy of a variety of different second order time stepping methods in computing the evolution of the magnetic field in a fully 3D calculation as determined by Eq. (1b). Our goal is to determine whether or not the effort involved in implementing the ETD scheme makes it a worthwhile choice of timestepping method, especially in view of the accuracy problems encountered in Section 5. We measure the accuracy as the absolute difference of magnetic energy ($\int_V \mathbf{B}^2\, \mathrm{d}V$ where V is the spherical shell of inner and outer radii $r_i = 1/2$, $r_o = 3/2$), between the results of each timestepping scheme and an "exact" solution which we define below. In each test, we prescribe the flow $\mathbf{u}$ to be that of Hollerbach et al. [28], which is axisymmetric and time periodic (with period 8). We additionally fix $R_\mathrm{m} = 300$, which represents a marginally unstable state, so that the fastest exponentially growing eigenmode of (1b) has a growth rate whose real part is positive but small. This means that the magnetic energy does not grow or decay too quickly so that we can perform a comparison over a meaningful length of time. We chose two initial fields both rescaled to have unit initial energy, one which is physically "smooth", and another which is "rough", the latter being approximately the fastest growing eigenmode (which is taken to be the magnetic field at $t = 40$ when using the smooth initial conditions). The physically smooth initial state was chosen to be of the form

$$S_l^m(r) = (r - r_o)^2 (r - r_i)^2, \quad T_l^m(r) = \sin\left(\frac{r - r_i}{r_o - r_i}\pi\right), \tag{27}$$

which satisfies electrically insulating boundary conditions; the energy in each harmonic was normalised to be exponentially decaying both in l and m. Fig. 4(a) shows, for both initial fields, the magnetic energy spectrum in spherical harmonic degree l and a 'quasi' energy spectrum in Chebyshev degree n, defined as the sum of the squares of all spectral coefficients, both poloidal and toroidal, for any particular Chebyshev degree. We use a truncation of 47 in spherical harmonic degree and 32 in Chebyshev degree which was sufficient to resolve the calculations; the Fortran 16-byte precision method was used to compute the ETD matrices.

We now perform a series of runs until $t = 100$ i.e. 12.5 flow periods, with various time stepping schemes as summarised in Table 1. The first four methods are all of the form $\mathbf{v}_{n+1} = \mathsf{A}\mathbf{v}_n + \mathsf{B}\mathbf{F}_n + \mathsf{C}\mathbf{F}_{n-1}$ and all have an identical computational cost per time step, each involving the same three matrix-vector multiplications. The last two methods are multistep of the form $\mathbf{v}_{n+1} = \mathbf{a} + \mathsf{X}\,(\mathbf{F}(\mathbf{a}, t_n + h) - \mathbf{F}_n)$ and themselves have identical cost.

Tests showed that the largest timestep possible, as dictated by the stability of the explicit treatment of the nonlinear terms, was $h_0 = 4 \times 10^{-3}$ for those methods requiring one transform per timestep (ETD2, IFAB2, ETDC2, CNAB2) and $2h_0$ for those requiring two transforms per timestep (ETD2RK, PC2). We use these choices of timestep to ensure that each run has the same computational cost. The "exact" solution (shown in Fig. 4(b)) is computed using the ETD2 method with a timestep of $h_0/20$ and has been verified to sufficiently high accuracy with the PC2 scheme (of Table 1) using a similarly small timestep. We note that for the first time step, the schemes ETD2, IFAB2, ETDC2 and CNAB2 all take a first order step, although a second-order

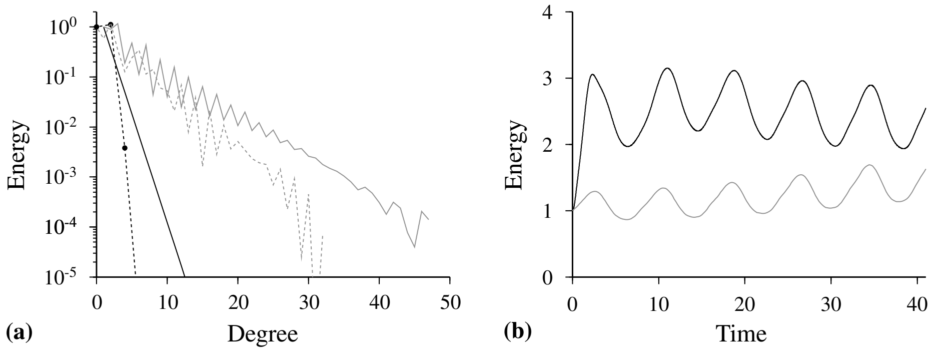

Fig. 4. (a) The spectra in spherical harmonic degree l (solid) and Chebyshev degree n (dashed) of the physically smooth initial condition (black) and that of the "rough" eigenmode initial state (grey). Each line is rescaled so that the lowest degree component is 1. The physically smooth state has energy only in even Chebyshev degrees by symmetry and those which are appreciable are shown as black circles; the dashed line merely connects them and does not indicate energy at any intermediate degree. (b) The evolution (which we take to be exact) of the physically smooth initial state (black) and the eigenmode initial state (grey); both have unit initial energy.

Table 1
Summary of time stepping methods given by their temporal discretisation of $\mathrm{d}\mathbf{v}/\mathrm{d}t = \mathsf{M}\mathbf{v} + \mathbf{F}(\mathbf{v}, t)$. ETD (exponential time differencing), ETDC2 (ETD scheme of Tilgner), CNAB2 (Crank–Nicolson with Adams–Bashforth), IFAB2 (integrating factor with Adams–Bashforth), ETD2RK (ETD Runge–Kutta) and PC2 (predictor–corrector based on Crank–Nicolson)

Name	Definition	Reference
ETD2	$\mathsf{A} = e^{\mathsf{M}h}$, $\mathsf{B} = hE_1(\mathsf{M}h)$ $\mathsf{C} = hE_2(\mathsf{M}h)$	Section 3.2
ETDC2	$\mathsf{A} = e^{\mathsf{M}h}$, $\mathsf{B} = 3hE_3(\mathsf{M}h)/2$ $\mathsf{C} = -hE_3(\mathsf{M}h)/2$	Section 3.2
CNAB2	$\mathsf{A} = (\mathsf{I} - \mathsf{M}h/2)^{-1}(\mathsf{I} + \mathsf{M}h/2)$, $\mathsf{B} = 3h(2\mathsf{I} - \mathsf{M}h)^{-1}$ $\mathsf{C} = -h(2\mathsf{I} - \mathsf{M}h)^{-1}$	Section 3.1
IFAB2	$\mathsf{A} = e^{\mathsf{M}h}$, $\mathsf{B} = 3he^{\mathsf{M}h}/2$, $\mathsf{C} = -he^{2\mathsf{M}h}/2$	Section 3.2
ETD2RK	$\mathbf{a} = e^{\mathsf{M}h}\mathbf{v}_n + \mathsf{M}^{-1}(e^{\mathsf{M}h} - 1)\mathbf{F}_n$ $\mathsf{X} = \mathsf{M}^{-2}h^{-1}(e^{\mathsf{M}h} - 1 - h\mathsf{M})$	See Ref. [2]
PC2	$\mathbf{a} = (\mathsf{I} - \mathsf{M}h/2)^{-1}(\mathsf{I} + \mathsf{M}h/2)\mathbf{v}_n + h(\mathsf{I} - \mathsf{M}h/2)^{-1}\mathbf{F}_n$ $\mathsf{X} = h(2\mathsf{I} - \mathsf{M}h)^{-1}$	See Ref. [15]

In the first four methods, $\mathbf{v}_{n+1} = \mathsf{A}\mathbf{v}_n + \mathsf{B}\mathbf{F}_n + \mathsf{C}\mathbf{F}_{n-1}$ and in the last two $\mathbf{v}_{n+1} = \mathbf{a} + \mathsf{X}\,(\mathbf{F}(\mathbf{a}, t_n + h) - \mathbf{F}_n)$.

multi-step method could have been used instead. Additionally, the boundary conditions are handled identically in ETD2, ETDC2, IFAB2 and ETD2RK, whereas in CNAB2 and PC2 they replace matrix rows of the discretised equations before inversion (see [15]).

Figs. 5(a) and (b) show the error in magnetic energy for each time stepping scheme for the two different initial conditions. We first note that all the methods give an accuracy of O(0.1%) over the time period shown and would be graphically indistinguishable if overplotted. In fact, in both figures the errors for ETD2, ETDC2 and CNAB2 are almost identical, as are those for PC2 and ETD2RK. Both initial conditions gave qualitatively the same picture, namely that for a fixed computational cost, the most accurate methods are ETD2, ETDC2 and CNAB2, with ETD2 being marginally better. The methods requiring two nonlinear transforms per timestep behave similarly and have greater error, while the least accurate method is the integrating factor scheme.

In this particular problem, the timestep is restricted by the nonlinear terms to be sufficiently small that the linearly implicit scheme CNAB2 can compete with the ETD methods. Since the CNAB2 scheme is significantly easier to implement, this is the best second order timestepping scheme to use in this problem. However, were we to use a higher order scheme, or were the nonlinear terms able to permit a larger timestep, it is likely that the ETD methods would become significantly more accurate as suggested by recent 1D computations [2,9]. In

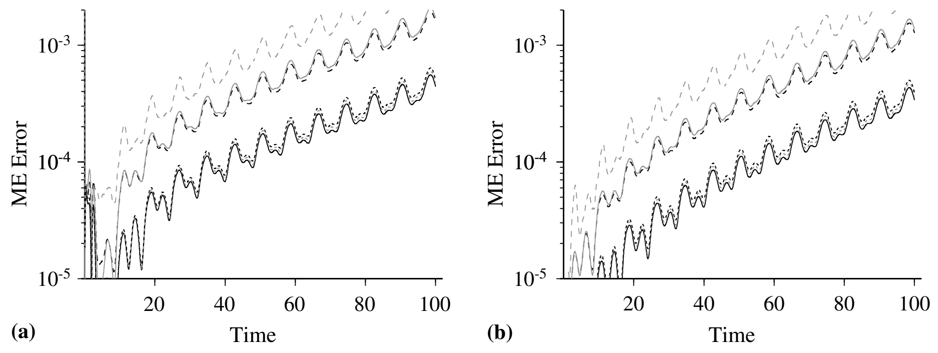

Fig. 5. The absolute error in magnetic energy for a fixed computational cost using (a) the physically smooth initial condition and (b) the eigenmode initial state shown as PC2 (solid grey), CNAB2 (short dashed grey), IFAB2 (long dashed grey), ETD2RK (long dashed black), ETDC2 (short dashed black) and ETD2 (solid black). In both cases, the CNAB2, ETD2 and ETDC2 lines almost overplot, as do PC2 and ETD2RK.

particular, it is trivial to extend the ETD2 scheme to arbitrary order, the only difference being that the code is required to store the nonlinear terms at many previous times. Since it is the evaluation of the nonlinear terms in such problems that is the most costly, such an extension would not increase the computational cost by any significant amount.

7. Conclusions

In this paper we have detailed an implementation of an exponential time differencing (ETD) time stepping scheme in a fully spectral code designed to solve the magnetohydrodynamics equations. One major problem with ETD schemes is the accurate computation of the necessary matrices; to this effect we compared a variety of different methods including recurrence relations, contour integration and spectral expansions. We found that the best, judged both on accuracy and computational expense, was that of the recurrence relations of Beylkin et al. [8] although the accuracy and stability of such a method could well be case specific. We lastly presented a comparison of six different second order time stepping schemes in the evolution of a 3D magnetic field in a spherical shell under the action of a time-dependent prescribed flow. We found that with a timestep constrained by the stability of the nonlinear terms, the linearly implicit and ETD schemes all exhibited similarly high accuracy, the ETD2 scheme of [2] being marginally the best at a fixed number of nonlinear transforms per simulation time. To timestep this problem at second order therefore, the best choice is a Crank–Nicolson scheme, since of the three most accurate this is the most straightforward to implement. This result vindicates the choice of method employed in most MHD timestepping codes to date. At higher order however, ETD methods being much more readily extendable are likely to exhibit higher accuracy than those which are linearly implicit and will become more advantageous.

Acknowledgements

P.L. thanks Paul Matthews, Steve Cox, Steve Tobias and David Hughes for many fruitful suggestions throughout this study, and Gary Glatzmaier for useful discussions on spectral methods. This work was supported by PPARC Grant PPA/G/S/2002/00545.

Appendix A. A Galerkin formulation for ETD2

A favourable alternative to the methods detailed in Section 4 is a Galerkin scheme in which we adopt an expansion in terms of a basis, every member of which explicitly satisfies the boundary conditions; such a basis may be expediently built from recombined Chebyshev polynomials. On discretising the system with respect to this basis, the boundary conditions may be essentially forgotten (e.g. [4,17]). Solutions of Eq. (3b) would then be particularly straightforward as we may immediately invert the $\mathscr{D}_l$ operator and timestep what remains, being of the form (3a). Although no fast transform exists between a general recombined basis of Chebyshev polynomials and physical space, we may incorporate both the forwards and backwards slow transforms into the timestepping matrices, which means that no loss of speed is suffered when timestepping. In some cases where the boundary conditions allow, the formulation of a Galerkin basis is particularly straightforward, for example, if the scalar functions S_l^m and $\mathrm{d}S_l^m/\mathrm{d}r$ vanish at both boundaries $r = r_i, r_o$ (nonslip), we may consider using

$$b_n(r) = (r_i - r)^2 (r_o - r)^2 T_n(f(r)). \tag{A.1}$$

However, this procedure does not extend to every equation in our situation: in particular, the magnetic boundary conditions are of mixed type and no such separable formulation is possible.

We now detail our implementation of a Galerkin scheme in the case of two boundary conditions, the extension to any greater number is immediate. A Chebyshev expansion of maximal degree N which satisfies these boundary conditions has $N-1$ degrees of freedom (since the remaining 2 are uniquely determined through the boundary conditions). We define the same number of independent basis functions b_n, $n = 0, 1, 2, \ldots, N-2$, by the relation

$$b_n(r) = T_{n+2}(f(r)) + \alpha_0 T_0(f(r)) + \alpha_1 T_1(f(r)), \tag{A.2}$$

where $f(r) = \frac{2r-(r_o+r_i)}{r_o-r_i} : [r_i, r_o] \to [-1, 1]$ and α_0, α_1 are determined by the boundary conditions. We may incorporate all these coefficients into a $(N+1) \times (N-1)$ "augmented" matrix G_{aug} which holds in each column n the Chebyshev representation of basis function n. Multiplication of G_{aug} with a vector defined with respect to the Galerkin basis simply gives the corresponding vector of Chebyshev coefficients. The matrix G defined by deleting the last 2 rows of G_{aug} actually contains the same information as G_{aug} since we may extend any column to its two remaining Chebyshev coefficients by use of the boundary conditions (hence the term "augmented" indicating superfluous information). The matrix G is a minimal representation of the Galerkin basis and is invertible; it therefore gives the transformation between the recombined Chebyshev basis and the Chebyshev coefficients themselves.

The ETD schemes require a discretisation of $\mathscr{D}_l$ with respect to the Galerkin basis, denoted D_{G}, which we form by projecting $\mathscr{D}_l b_j(r)$, for each $b_j(r)$, back onto the basis functions in a least squares methodology with a weighting function $q(r)$. This is given by $D_{\mathrm{G}} = B^{-1}L$ where B and L are

$$L_{ij} = \int_{r_i}^{r_o} b_i(r) \mathscr{D}_l b_j(r) q(r)\, \mathrm{d}r, \quad B_{ij} = \int_{r_i}^{r_o} b_i(r) b_j(r) q(r)\, \mathrm{d}r. \tag{A.3}$$

Note that D_{G} contains a factor of B^{-1} which takes account of the fact that the basis is not orthornormal [17]. We typically choose the (arbitrary) function $q(r)$ to be 1 or r^2, the latter representing a quasi energy. The matrix elements of B and L may be computed using Gauss-quadrature on a sufficiently large number of grid points.

We may implement this in the ETD2 scheme by absorbing the slow transforms between Chebyshev and Galerkin basis function space, for example by using $h\mathsf{G}E_1(D_{\mathrm{G}}h)\mathsf{G}^{-1}$ in place of $hE_1(D_\tau h)$.

Although we find this scheme to be stable when applied to Eq. (3a), it is unstable when extending the above methodology with four boundary conditions to Eq. (3b) except for very small time steps, when it agrees with the influence matrix method we have implemented. This instability seems to stem from a more stringent CFL restriction on the nonlinear terms and not from the linear part, which has the (correct) negative real eigenvalues.

It is worth pointing out that this is an analogous method to that suggested by Huang and Sloan [23] who solve the eigenvalue problem related to the linear part of Eq. (3b) by differentiation matrices. In their scheme, the polynomial interpolation of the data at the grid points incorporates a multiplicative function which ensures that the (four) boundary conditions are implicitly satisfied.

References

[1] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, second ed., Springer, Berlin, 2002.
[2] S.M. Cox, P.C. Matthews, Exponential time differencing for stiff systems, J. Comput. Phys. 176 (2002) 430–455.

[3] B.V. Minchev, W.M. Wright, A review of exponential integrators for first order semi-linear problems, Norwegian University of Science and Tech., Numerics (Rep. 2).
[4] J.P. Boyd, Chebyshev and Fourier Spectral Methods, Dover, New York, 2001.
[5] J.D. Lambert, Numerical Methods for Ordinary Differential Systems, Wiley, New York, 1991.
[6] M. Hochbruck, C. Lubich, H. Selhofer, Exponential integrators for large systems of differential equations, SIAM J. Sci. Comp. 19 (5) (1998) 1552–1574.
[7] S. Krogstad, Generalized integrating factor methods for stiff PDEs, J. Comput. Phys. 203 (2005) 72–88.
[8] G. Beylkin, J.M. Keiser, L. Vozovoi, A new class of time discretization schemes for the solution of nonlinear PDEs, J. Comput. Phys. 147 (1997) 362–387.
[9] A.-K. Kassam, L.N. Trefethen, Fourth-order time stepping for stiff PDEs, SIAM J. Sci. Comp. 26 (4) (2005) 1214–1233.
[10] P. Davies, N.J. Higham, A Schur–Parlett algorithm for computing matrix functions, SIAM J. Matrix Anal. Appl. 25 (2) (2003) 464–485.
[11] M. Hochbruck, C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 34 (5) (1997) 1911–1925.
[12] A.-K. Kassam, High order timestepping for stiff semilinear partial differential equations, Ph.D. Thesis, University of Oxford, 2004.
[13] H. Tal-Ezer, Spectral methods in time for parabolic problems, SIAM J. Numer. Anal. 26 (1) (1989) 1–11.
[14] G.A. Glatzmaier, Numerical simulations of stellar convective dynamos. I. The model and method, J. Comput. Phys. 55 (1984) 461–484.
[15] R. Hollerbach, A spectral solution of the magneto-convection equations in spherical geometry, Int. J. Numer. Meth. Fluids 32 (2000) 773–797.
[16] A. Tilgner, Spectral methods for the simulations of incompressible flows in spherical shells, Int. J. Numer. Meth. Fluids 30 (1999) 713–724.
[17] P.W. Livermore, A. Jackson, A comparison of numerical schemes to solve the magnetic induction eigenvalue problem in a spherical geometry, Geophys. Astrophys. Fluid Dyn. 99 (6) (2005) 467–480.
[18] M. Mohlenkamp, A fast transform for spherical harmonics, J. Fourier Anal. Appl. 5 (2/3) (1999) 159–184.
[19] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, Twenty five years later, SIAM Rev. 45 (1) (2003) 3–49.
[20] D. Pisarenko, L. Biferale, D. Courvoisier, U. Frisch, M. Vergassola, Further results on multifractality in shell models, Phys. Fluids A 5 (10) (1993) 2533–2538.
[21] L.N. Trefethen, Spectral Methods in MATLAB, SIAM, Philadelphia, PA, 2000.
[22] C.J. Talbot, A. Crampton, Application of the pseudo-spectral method for a 2D eigenvalue problems in elasticity, Numer. Alg. 38 (2005) 95–110.
[23] W. Huang, D.M. Sloan, The psuedospectral method for solving differential eigenvalue problems, J. Comput. Phys. 111 (1994) 399–409.
[24] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, Spectral Methods for Fluid Dynamics, Springer, Berlin, 1988.
[25] B.V. Minchev, Computing analytic matrix functions for a class of exponential integrators, Department of Informatics, University of Bergen, Norway: Reports in Informatics (Rep. 278).
[26] R.B. Sidje, Expokit: A software package for computing matrix exponentials, ACM Trans. Math. Soft. 24 (1) (1998) 130–156.
[27] M. Abramowitz, I.A. Stegun, Pocketbook of Mathematical Functions, Verlah Harri Deutsch, 1984.
[28] R. Hollerbach, D.J. Galloway, M.R.E. Proctor, Numerical evidence of fast dynamo action in a spherical shell, Phys. Rev. Lett. 74 (16) (1995) 3145–3148.