FOURTH-ORDER TIME-STEPPING FOR STIFF PDES ON THE SPHERE

HADRIEN MONTANELLI* AND YUJI NAKATSUKASA†

Abstract. We present in this paper algorithms for solving stiff PDEs on the unit sphere with spectral accuracy in space and fourth-order accuracy in time. These are based on a variant of the double Fourier sphere method in coefficient space with multiplication matrices that differ from the usual ones, and implicit-explicit time-stepping schemes. Operating in coefficient space with these new matrices allows one to use a sparse direct solver, avoids the coordinate singularity and maintains smoothness at the poles, while implicit-explicit schemes circumvent severe restrictions on the time-steps due to stiffness. A comparison is made against exponential integrators and it is found that implicit-explicit schemes perform best. Implementations in MATLAB and Chebfun make it possible to compute the solution of many PDEs to high accuracy in a very convenient fashion.

Key words. Stiff PDEs, exponential integrators, implicit-explicit, PDEs on the sphere, double Fourier sphere method, Chebfun

AMS subject classifications. 65L04, 65L05, 65M20, 65M70, 65T40

1. Introduction. We are interested in computing smooth solutions of stiff PDEs on the unit sphere of the form

$$u_t = \mathcal{L}u + \mathcal{N}(u), \quad u(t = 0, x, y, z) = u_0(x, y, z), \tag{1.1}$$

where $u(t, x, y, z)$ is a function of time t and Cartesian coordinates (x, y, z) with $x^2 + y^2 + z^2 = 1$. The function u can be real or complex and (1.1) can be a single equation, as well as a system of equations. In this paper, we restrict our attention to $\mathcal{L}u = \alpha\Delta u$ and to a nonlinear non-differential operator $\mathcal{N}$ with constant coefficients, but the techniques we present can be applied to more general cases. A large number of PDEs of interest in science and engineering take this form. Examples on the sphere include the (diffusive) Allen–Cahn equation $u_t = \epsilon\Delta u + u - u^3$ with $\epsilon \ll 1$ [18], the (dispersive) focusing nonlinear Schrödinger equation $u_t = i\Delta u + iu|u|^2$ [49], the Gierer–Meinhardt [7], Ginzburg–Landau [44] and Brusselator [55] equations, and many others.

There are several methods to discretize the spatial part of (1.1) with spectral accuracy, including spherical harmonics [5], radial basis functions (RBFs) [21] and the double Fourier sphere (DFS) method [33, 37]. The DFS method is the only one that leads to a $\mathcal{O}(N \log N)$ complexity per time-step, where N is the total number of grid points in the spatial discretization of (1.1). For spherical harmonics, the cost per time-step is $\mathcal{O}(N^{3/2})$ since there are no effective "fast" spherical transforms,[1] and for global RBFs [21, Ch. 6], the cost per time-step is $\mathcal{O}(N^2)$ since these generate dense differentiation matrices.[2] We focus in this paper on the DFS method and present a novel formulation operating in coefficient space.

---

[1] Fast $\mathcal{O}(N \log N)$ spherical transforms have received significant attention but require so far a $\mathcal{O}(N^2)$ precomputational cost [43, 56, 57]. Note that in a recent manuscript [47] Slevinsky proposed a new fast spherical transform based on conversions between spherical harmonics and bivariate Fourier series with a lower $\mathcal{O}(N^{3/2})$ precomputational cost. For implicit-explicit schemes with the DFS method, the precomputation is $\mathcal{O}(N)$.

[2] RBF-FD [21, Ch. 7] generate sparse matrices but only achieve algebraic orders of accuracy. Moreover, the solution time for these sparse matrices is not necessarily $\mathcal{O}(N)$.
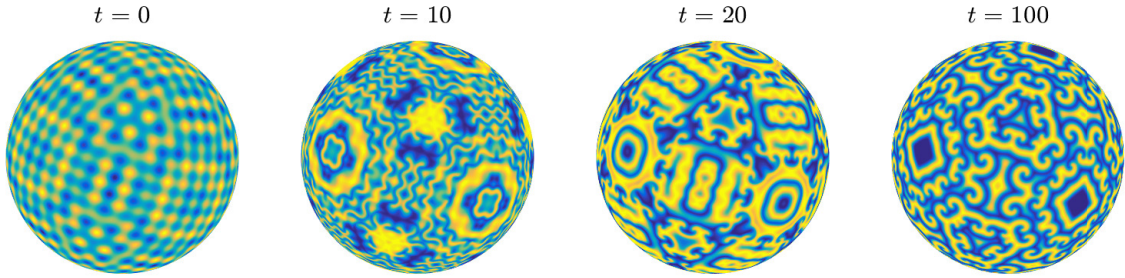
$t = 0$ $t = 10$ $t = 20$ $t = 100$

Fig. 1.1: *Initial condition and real part of the solution at times $t = 10, 20, 100$ of the Ginzburg–Landau equation computed by the* `spinsphere` *code. This solution is oriented in a direction at a $\pi/8$ angle from the north-south axis, so the symmetry maintained in the computations is a reflection of global accuracy.*

Once the spatial part of (1.1) has been discretized by the DFS method on an $n \times m$ uniform longitude-latitude grid, it becomes a system of nm ODEs,

$$\hat{u}' = \mathbf{L}\hat{u} + \mathbf{N}(\hat{u}), \quad \hat{u}(0) = \hat{u}_0, \tag{1.2}$$

where $\hat{u}(t)$ is a vector of nm Fourier coefficients and $\mathbf{L}$ (an $nm \times nm$ matrix) and $\mathbf{N}$ are the discretized versions of $\mathcal{L}$ and $\mathcal{N}$ in Fourier space. Solving the system (1.2) with generic explicit time-stepping schemes can be highly challenging because of *stiffness*: the large eigenvalues of $\mathbf{L}$—due to the second differentiation order in (1.1) and the clustering of points near the poles—force one to use very small time-steps. Exponential integrators and implicit-explicit (IMEX) schemes are two classes of numerical methods that are aimed at treating stiffness. For exponential integrators, the linear part $\mathbf{L}$ is integrated exactly using the matrix exponential while a numerical scheme is applied to the nonlinear part $\mathbf{N}$. For IMEX schemes, an explicit formula is used to advance $\mathbf{N}$ while an implicit scheme is used to advance $\mathbf{L}$. We show in this paper that the DFS method combined with IMEX schemes leads to $\mathcal{O}(nm \log nm)$ per time-step algorithms for both diffusive and dispersive PDEs, that exponential integrators achieve this complexity for diffusive PDEs only, and that IMEX schemes outperform exponential integrators in both cases. For numerical comparisons, we consider two versions of the fourth-order ETDRK4 exponential integrator of Cox and Matthews [13] and two fourth-order IMEX schemes, the IMEX-BDF4 [28] and LIRK4 [9] schemes.

By contrast, Kassam and Trefethen demonstrated in [29] that exponential integrators (ETDRK4) outperform IMEX schemes (IMEX-BDF4); this can be explained by two factors. First, they focused on problems with diagonal matrices $\mathbf{L}$. (IMEX-BDF4 performed better than ETDRK4 for the only non-diagonal problem they considered.) For diagonal problems, exponential integrators are particularly efficient since the computation of the matrix exponential is trivial and the matrix exponential is diagonal too (hence, its action on vectors can trivially be computed in linear time). Second, IMEX-BDF4 is unstable for dispersive PDEs since it is based on the fourth-order backward differentiation formula, which is unstable for dispersive PDEs—this is why they could not make it work for the KdV equation. Our DFS method in coefficient space leads to matrices that are not diagonal but have a sparsity structure that makes IMEX schemes particularly efficient, and we consider not only IMEX-BDF4 but also LIRK4, which is stable for dispersive PDEs.

There are libraries for solving time-dependent PDEs on the sphere, including SPHEREPACK [1] and FEniCS [42]. However, none of these are aimed at solving stiff PDEs and easily allow for computing in an integrated environment. The algorithms we shall describe in this paper are aimed at solving stiff PDEs, and have been implemented in MATLAB and made available as part of Chebfun [16] in the `spinsphere` code. (Note that `spin` stands for **s**tiff **P**DE **in**tegrator.) The recent extension of Chebfun to the sphere [51], built on its extension to periodic problems [59], provides a very convenient framework for working with functions on the sphere. For example, the function

$$f(\lambda, \theta) = \cos(1 + \cos\lambda\sin(2\theta)) \tag{1.3}$$

can be approximated to machine precision by the following command:

```
f = spherefun(@(lam,th) cos(1 + cos(lam).*sin(2*th)));
```

Using `spherefun` objects as initial conditions, the `spinsphere` code allows one to solve stiff PDEs with a few lines of code, using IMEX-BDF4 for diffusive and LIRK4 for dispersive PDEs. For example, the following MATLAB code solves the Ginzburg–Landau equation $u_t = 10^{-4}\Delta u + u - (1 + 1.5i)u|u|^2$ with 1024 grid points in longitude and latitude and a time-step $h = 10^{-1}$:

```
n = 1024;                                  % number of grid pts
h = 1e-1;                                  % time-step
tspan = [0 100];                           % time interval
S = spinopsphere(tspan);                   % initialize operator
S.lin = @(u) 1e-4*lap(u);                  % linear part
S.nonlin = @(u) u-(1+1.5i)*u.*abs(u).^2;   % nonlinear part
u0 = @(x,y,z) 1/3*(cos(40*x)+cos(40*y)+cos(40*z)); % initial cond.
th = pi/8; c = cos(th); s = sin(th);       % pi/8 rotation
S.init = spherefun(@(x,y,z)u0(c*x-s*z,y,s*x+c*z)); % rotated initial cond.
u = spinsphere(S, n, h);                   % solve
```

The initial condition and the solution at times $t = 10, 20, 100$ are shown in Figure 1.1. (The Ginzburg–Landau equation in 2D goes back to the 1970s with the work of Stewartson and Stuart [48]. Rubinstein and Sternberg studied it on the sphere [44], with applications in the study of liquid crystals [40] and nonequilibrium patterns [41].) Figure A.1 lists a detailed MATLAB code to solve the same problem; a more sophisticated version of this code is used inside `spinsphere`. (Note that this code might be a little bit slow; for speed, the reader might want to adjust n and h, and change the initial condition. We also encourage the reader to type `spinsphere('gl')` or `spinsphere('nls')` to invoke an example computation.)

The paper is structured as follows. In the next section, we review the DFS method (Section 2.1) and present a new Fourier spectral method in coefficient space, which, using multiplication matrices that differ from the usual ones, avoids the coordinate singularity (Section 2.2), takes advantage of sparse direct solvers (Section 2.3), and maintains smoothness at the poles (Sections 2.4 and 2.5). The time-stepping schemes are presented in Section 3 while Section 4 is dedicated to numerical comparisons on simple PDEs.

2. A Fourier spectral method in coefficient space. We present in this section a Fourier spectral method for the spatial discretization of (1.1), based on the DFS method and novel Fourier multiplication matrices in coefficient space. The accuracy of the method is tested by solving the Poisson and heat equations.

2.1. The double Fourier sphere method. The DFS method uses the longitude-latitude coordinate transform,

$$x = \cos\lambda\sin\theta, \ y = \sin\lambda\sin\theta, \ z = \cos\theta, \tag{2.1}$$

with $(\lambda, \theta) \in [-\pi, \pi] \times [0, \pi]$. The azimuth angle λ corresponds to the longitude while the polar (or zenith) angle θ corresponds to the latitude.[3] A function $u(x, y, z)$ on the sphere is written as $u(\lambda, \theta)$ using (2.1), i.e.,

$$u(\lambda, \theta) = u(\cos\lambda\sin\theta, \sin\lambda\sin\theta, \cos\theta), \quad (\lambda, \theta) \in [-\pi, \pi] \times [0, \pi], \tag{2.2}$$

and (1.1) with $\mathcal{L} = \alpha\Delta$ becomes

$$u_t = \alpha\Delta u + \mathcal{N}(u), \quad u(t = 0, \lambda, \theta) = u_0(\lambda, \theta), \quad (\lambda, \theta) \in [-\pi, \pi] \times [0, \pi]. \tag{2.3}$$

Note that the function $u(\lambda, \theta)$ in (2.2) is 2π-periodic in λ but not periodic in θ. The key idea of the DFS method—developed by Merilees [33] and further studied by Orszag [37] in the 1970's, and recently revisited by Townsend et al. with the use of low-rank approximations [51]—is to associate a function $\tilde{u}(\lambda, \theta)$ with $u(\lambda, \theta)$, 2π-periodic in both λ and θ, defined on $[-\pi, \pi] \times [-\pi, \pi]$, and constant along the lines $\theta = 0$ and $\theta = \pm\pi$ corresponding to the poles. Mathematically, the function $\tilde{u}(\lambda, \theta)$ is defined as

$$\tilde{u}(\lambda, \theta) = \begin{cases} u(\lambda, \theta), & (\lambda, \theta) \in [-\pi, \pi] \times [0, \pi], \\ u(\lambda + \pi, -\theta), & (\lambda, \theta) \in [-\pi, 0] \times [-\pi, 0], \\ u(\lambda - \pi, -\theta), & (\lambda, \theta) \in [0, \pi] \times [-\pi, 0]. \end{cases} \tag{2.4}$$

The function u is "doubled-up" in the θ-direction and flipped; see, e.g., [51, Fig. 1]. Since the function $\tilde{u}$ is 2π-periodic in both λ and θ, it can be approximated by a 2D Fourier series,

$$\tilde{u}(\lambda, \theta) \approx \sum_{j=-m/2}^{m/2} {}' \sum_{k=-n/2}^{n/2} {}' \hat{u}_{jk} e^{ij\theta} e^{ik\lambda}. \tag{2.5}$$

The numbers n and m are assumed to be even (this will be the case throughout this paper) and the primes on the summation signs mean that the boundary terms $j = \pm m/2$ or $k = \pm n/2$ are halved. The Fourier coefficients are defined by

$$\hat{u}_{jk} = \frac{1}{nm} \sum_{p=1}^{m} \sum_{q=1}^{n} \tilde{u}(\lambda_q, \theta_p) e^{-ij\theta_p} e^{-ik\lambda_q}, \quad -\frac{m}{2} \le j \le \frac{m}{2} - 1, \quad -\frac{n}{2} \le k \le \frac{n}{2} - 1, \tag{2.6}$$

with $\hat{u}_{j,n/2} = \hat{u}_{j,-n/2}$ for all j and $\hat{u}_{m/2,k} = \hat{u}_{-m/2,k}$ for all k, and correspond to a 2D uniform grid with n points in longitude and m points in latitude,

$$\lambda_q = -\pi + (q-1)\frac{2\pi}{n}, \quad 1 \le q \le n, \quad \theta_p = -\pi + (p-1)\frac{2\pi}{m}, \quad 1 \le p \le m. \tag{2.7}$$

The nm Fourier coefficients $\hat{u}_{jk}$ can be computed by sampling $\tilde{u}$ on the grid and using the 2D FFT, costing $\mathcal{O}(nm\log nm)$ operations. In practice we take $m = n$ since it leads to the same resolution in each direction around the equator where the spacing is the coarsest.

---

[3]To be precise, θ is the colatitude, defined as "$\pi/2$ minus latitude" with latitude in $[-\pi/2, \pi/2]$. For simplicity, we will refer to it as latitude. Note that $\theta = 0$ corresponds to the north pole and $\theta = \pi$ to the south pole.

As mentioned in [51], every smooth function $u(\lambda, \theta)$ on the sphere is associated with a smooth bi-periodic function $\tilde{u}(\lambda, \theta)$ on $[-\pi, \pi]^2$ via (2.4), but the converse is not true since smooth bi-periodic functions might not be constant along the lines $\theta = 0$ and $\theta = \pm\pi$ corresponding to the poles. To be smooth on the sphere, functions of the form (2.4) have to satisfy the *pole conditions*, which ensures that $\tilde{u}(\lambda, \theta)$ is single-valued at the poles despite the fact that latitude circles degenerate into a single point there. For approximations of the form (2.5)–(2.6), this is given by

$$\sum_{j=-m/2}^{m/2}{}' \hat{u}_{jk} = \sum_{j=-m/2}^{m/2}{}' (-1)^j \hat{u}_{jk} = 0, \quad |k| \geq 1. \tag{2.8}$$

As we will see in the numerical experiments of Sections 2.4 and 2.5, when solving a PDE involving the Laplacian operator, if the right-hand side (for Poisson's equation) or the initial condition (for the heat equation) is a smooth function on the sphere, then the solutions obtained with our DFS method are also smooth functions on the sphere. Therefore, we do not have to impose the conditions (2.8). Similarly, we do not impose the "doubled-up" symmetry in (2.4), as it was preserved throughout our experiments (we leave as an open problem to prove this). For brevity we only illustrate the condition (2.8) in our experiments.

Let us finish this section with some comments about Fourier series for solving PDEs on the sphere. One way of using them is to use standard double Fourier series on a "doubled-up" version of u, i.e., the DFS method (2.4)–(2.6). This is what Merilees did and, combined with a Fourier spectral method in value space, he solved the shallow water equations [33]. Another way is to use half-range cosine or sine series [8, 11, 37, 46, 60]—after all, spherical harmonics are represented as proper combinations of half-ranged cosine or sine series. For example, Orszag [37] suggested the use of approximations of the form

$$u(\lambda, \theta) \approx \sum_{k=-n/2}^{n/2} \sum_{j=0}^{n/2} \hat{u}_{jk} \sin^s \theta \cos j\theta e^{ik\lambda}, \tag{2.9}$$

where $s = 0$ if k is even, $s = 1$ if k is odd. (Note that the Fourier series in (2.9) approximates u directly, as opposed to $\tilde{u}$.) When using half-range cosine or sine terms as basis functions, one must be careful that the pole conditions are satisfied. One can either select basis functions that satisfy the pole conditions or impose a constraint on the Fourier coefficients to enforce it. For solving PDEs involving the Laplacian operator (in both value and coefficient spaces), Orszag imposed certain constraints on the coefficients $\hat{u}_{jk}$ in (2.9), analogous to those in (2.8),

$$\sum_{j=0}^{n/2} \hat{u}_{jk} = \sum_{j=0}^{n/2} (-1)^j \hat{u}_{jk} = 0, \quad |k| \geq 2. \tag{2.10}$$

Boyd [8] studied Orszag's method and showed that the constraints (2.10) are actually not necessary for solving *time-independent* PDEs in value space but mentioned that the "absence of pole constraints is still risky" when working with coefficients.

The spectral method we present in this paper is based on Merilees' approach and is similar to the method of Townsend et al. [51], but the Fourier multiplication matrices we use are different. It is simpler to implement than the half-range cosine/sine methods since there are no constraints to impose, and gives comparable accuracy.

2.2. Fourier multiplication matrices in coefficient space. In this section, we are interested in finding a matrix for multiplication by $\sin^2\theta$ that is nonsingular. This is crucial because some of the time-stepping methods we shall describe in Section 3 need to compute the inverse of such a matrix. For this discussion we can restrict our attention to the 1D case.

Consider an even number m of equispaced points $\{\theta_p\}_{p=1}^m$ on $[-\pi,\pi]$,

$$\theta_p = -\pi + (p-1)\frac{2\pi}{m}, \quad 1 \le p \le m. \tag{2.11}$$

(Note that these points include $-\pi$ but not π.) Let u be a complex-valued function on $[-\pi,\pi]$ with values $\{u_p\}_{p=1}^m$ at these points. It is well known [25, Ch. 13] that there exists a unique degree $m/2$ trigonometric polynomial $p(\theta)$ that interpolates $u(\theta)$ at these m points, i.e., such that $p(\theta_p) = u_p$ for each p, of the symmetric form

$$p(\theta) = \sum_{j=-m/2}^{m/2}{}' \hat{u}_j e^{ij\theta}, \tag{2.12}$$

with Fourier coefficients

$$\hat{u}_j = \frac{1}{m}\sum_{p=1}^m u_p e^{-ij\theta_p}, \quad -\frac{m}{2} \le j \le \frac{m}{2} - 1, \tag{2.13}$$

and $\hat{u}_{m/2} = \hat{u}_{-m/2}$. The prime on the summation sign indicates that the terms $j = \pm m/2$ are halved. Hence, we shall define the vector of $m+1$ Fourier coefficients as

$$\hat{u} = \left(\frac{\hat{u}_{-m/2}}{2}, \hat{u}_{-m/2+1}, \ldots, \hat{u}_{m/2-1}, \frac{\hat{u}_{m/2}}{2} = \frac{\hat{u}_{-m/2}}{2}\right)^T. \tag{2.14}$$

Let us emphasize that if the trigonometric interpolant were defined as

$$p(\theta) = \sum_{j=-m/2}^{m/2-1} \hat{u}_j e^{ij\theta}, \tag{2.15}$$

the derivative of (2.15) would have a mode $(-im/2)e^{-im\theta/2}$ leading to complex values for real data.[4] However, FFT codes only store m coefficients, i.e., they assume that $p(\theta)$ is of the form (2.15) with

$$\hat{u} = \left(\frac{\hat{u}_{-m/2}}{2} + \frac{\hat{u}_{m/2}}{2} = \hat{u}_{-m/2}, \hat{u}_{-m/2+1}, \ldots, \hat{u}_{m/2-1}\right)^T. \tag{2.16}$$

As a consequence, the first entry of the $m \times m$ first-order Fourier differentiation matrix $\mathbf{D}_m$, which acts on (2.16), is zero,

$$\mathbf{D}_m = \mathrm{diag}\Big(i(0, -m/2+1, -m/2+2, \ldots, m/2-1)\Big), \tag{2.17}$$

---

[4]Consider for example $u(\theta) = \cos(\theta)$ with $m = 2$. The representation (2.15) gives $p(\theta) = e^{-i\theta}$ with correct values -1 and 1 at grid points $\theta_1 = -\pi$ and $\theta_2 = 0$ but its derivative $p'(\theta) = -ie^{-i\theta}$ is complex-valued on the grid. The representation (2.12) gives $p(\theta) = 1/2(e^{-i\theta} + e^{i\theta})$, which is indeed the correct answer.

to cancel the mode $(-im/2)e^{-im\theta/2}$. Another way of seeing this is to adopt the following point of view: to compute derivatives, we map the vector of m Fourier coefficients (2.16) to the representation (2.14) with $m+1$ coefficients, differentiate, and then map back to m coefficients. Thus, $\mathbf{D}_m$ can be written as the product of three matrices,[5]

$$\mathbf{D}_m = \mathbf{Q}\mathbf{D}_{m+1}\mathbf{P} \tag{2.18}$$

where the $(m+1) \times m$ matrix $\mathbf{P}$ maps (2.16) to (2.14),

$$\mathbf{P} = \begin{pmatrix} \frac{1}{2} & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ \frac{1}{2} & & & 0 \end{pmatrix}, \tag{2.19}$$

$\mathbf{D}_{m+1}$ is the $(m+1) \times (m+1)$ first-order Fourier differentiation matrix,

$$\mathbf{D}_{m+1} = \mathrm{diag}\Big(i(-m/2, -m/2+1, \ldots, m/2)\Big), \tag{2.20}$$

and $\mathbf{Q}$ is the $m \times (m+1)$ matrix that maps back to m coefficients,

$$\mathbf{Q} = \begin{pmatrix} 1 & & & & 1 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{pmatrix}. \tag{2.21}$$

(Note that the first entry of the differentiation matrix (2.20) is nonzero.)

The same point of view can be adopted for multiplication matrices, with the difference that multiplying by $\sin^2\theta$ or $\cos\theta\sin\theta$ will increase the length of the representation by four since

$$\sin^2\theta = -\frac{1}{4}e^{-2i\theta} + \frac{1}{2} - \frac{1}{4}e^{2i\theta}, \quad \cos\theta\sin\theta = -\frac{1}{4}e^{-2i\theta} + \frac{1}{4}e^{2i\theta}. \tag{2.22}$$

Therefore, to multiply by, e.g., $\sin^2\theta$, we map (2.16) to (2.14), multiply by $\sin^2\theta$ with an $(m+1+4) \times (m+1)$ matrix, and then truncate and map back to m coefficients. The resulting matrix for multiplication by $\sin^2\theta$, which we denote by $\mathbf{T}_{\sin^2}$, is given by

$$\mathbf{T}_{\sin^2} = \mathbf{Q}\mathbf{M}_{\sin^2}(:, 3 : m+3)\mathbf{P}, \tag{2.23}$$

---

[5]Readers might find details such as (2.18)–(2.21) unexciting, and we would not disagree. But what trouble it causes in computations if you do not get these details right!

where $\mathbf{M}_{\sin^2}$ is the $(m+1+4) \times (m+1+4)$ matrix defined by

$$
\mathbf{M}_{\sin^2} = \begin{pmatrix}
\frac{1}{2} & 0 & -\frac{1}{4} & & & & \\
0 & \frac{1}{2} & 0 & -\frac{1}{4} & & & \\
-\frac{1}{4} & 0 & \ddots & \ddots & \ddots & & \\
& -\frac{1}{4} & \ddots & \ddots & \ddots & \ddots & \\
& & \ddots & \ddots & \ddots & \ddots & -\frac{1}{4} \\
& & & \ddots & \ddots & \ddots & 0 \\
& & & & -\frac{1}{4} & 0 & \frac{1}{2}
\end{pmatrix},
\tag{2.24}
$$

$\mathbf{P}$ is defined as before and $\mathbf{Q}$ is the following $m \times (m+1+4)$ matrix,

$$
\mathbf{Q} = \begin{pmatrix}
0 & 0 & 1 & & & & 1 & 0 & 0 \\
& & & 1 & & & & & \\
& & & & \ddots & & & & \\
& & & & & 1 & 0 & 0 & 0
\end{pmatrix}.
\tag{2.25}
$$

We have used MATLAB notation in (2.23): $\mathbf{M}_{\sin^2}(:, 3 : m+3)$ is obtained from (2.24) by removing the first and last two columns—these columns would hit zero coefficients in the padded-with-zeros version of $\hat{u}$. This leads to

$$
\mathbf{T}_{\sin^2} = \begin{pmatrix}
\frac{1}{2} & 0 & -\frac{1}{4} & & & & -\frac{1}{4} & 0 \\
0 & \frac{1}{2} & 0 & -\frac{1}{4} & & & & 0 \\
-\frac{1}{8} & 0 & \ddots & \ddots & \ddots & & & \\
& -\frac{1}{4} & \ddots & \ddots & \ddots & \ddots & & \\
& & -\frac{1}{4} & \ddots & \ddots & \ddots & -\frac{1}{4} & \\
& & & \ddots & \ddots & \ddots & \ddots & -\frac{1}{4} \\
-\frac{1}{8} & & & & -\frac{1}{4} & \ddots & \frac{1}{2} & 0 \\
0 & 0 & & & & -\frac{1}{4} & 0 & \frac{1}{2}
\end{pmatrix}.
\tag{2.26}
$$

Using the Gershgorin circle theorem [58] we see that the $m \times m$ matrix (2.26) is nonsingular since it is row diagonally dominant, with strict diagonal dominance in the second row, and irreducible.

    Let us add some comments about (2.26). If we operated in value space, we would obtain a singular matrix, since the multiplication matrix in value space, $\mathbf{M}_{\sin^2}^v$, a diagonal matrix with entries $\{\sin^2 \theta_p\}_{p=1}^m$, has two zeros corresponding to $\theta_p = -\pi$ and $\theta_p = 0$. (The standard remedy in that case is to shift the θ-grid so that it does not contain the poles [33].) From this matrix, we can

obtain a multiplication in coefficient space by multiplying by the DFT matrix $\mathbf{F}$ and its inverse,

$$
\mathbf{F}\mathbf{M}^v_{\sin^2}\mathbf{F}^{-1} = \begin{pmatrix}
\frac{1}{2} & 0 & -\frac{1}{4} & & & & -\frac{1}{4} & 0 \\
0 & \frac{1}{2} & 0 & -\frac{1}{4} & & & & -\frac{1}{4} \\
-\frac{1}{4} & 0 & \ddots & \ddots & \ddots & & & \\
& -\frac{1}{4} & \ddots & \ddots & \ddots & \ddots & & \\
& & \ddots & \ddots & \ddots & \ddots & -\frac{1}{4} & \\
& & & \ddots & \ddots & \ddots & \ddots & -\frac{1}{4} \\
-\frac{1}{4} & & & & -\frac{1}{4} & \ddots & \frac{1}{2} & 0 \\
0 & -\frac{1}{4} & & & & -\frac{1}{4} & 0 & \frac{1}{2}
\end{pmatrix}.
\tag{2.27}
$$

This matrix is indeed singular since $\mathbf{F}$ defines a unitary transformation, and its null space contains the vectors $(1,1,\ldots)^T$ and $(1,-1,1,-1,\ldots)^T$, which correspond to the Fourier coefficients of the delta functions at $\theta = 0$ and $\theta = -\pi$.

In [51], the authors use the $m \times m$ version of (2.24) (which is also nonsingular) as opposed to (2.26); this leads to incorrect results for trigonometric polynomials of degree $m/2 - 2$. To illustrate this, let us consider $m = 6$ and multiply $\sin^2(\theta)$ by a trigonometric polynomial of degree $m/2 - 2 = 1$, e.g., $\cos(\theta)$. In the representation (2.16), the function $\cos(\theta) = 1/2(e^{-i\theta} + e^{i\theta})$ has coefficients

$$
c = \left(0, 0, \frac{1}{2}, 0, \frac{1}{2}, 0\right)^T,
\tag{2.28}
$$

while the product $\cos(\theta)\sin(\theta) = -1/8(e^{-3i\theta} + e^{3i\theta}) + 1/8(e^{-i\theta} + e^{i\theta})$ has coefficients

$$
d = \left(-\frac{1}{4}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0\right)^T.
\tag{2.29}
$$

For $m = 6$, we have

$$
\mathbf{T}_{\sin^2} = \begin{pmatrix}
\frac{1}{2} & 0 & -\frac{1}{4} & 0 & -\frac{1}{4} & 0 \\
0 & \frac{1}{2} & 0 & -\frac{1}{4} & 0 & 0 \\
-\frac{1}{8} & 0 & \frac{1}{2} & 0 & -\frac{1}{4} & 0 \\
0 & -\frac{1}{4} & 0 & \frac{1}{2} & 0 & -\frac{1}{4} \\
-\frac{1}{8} & 0 & -\frac{1}{4} & 0 & \frac{1}{2} & 0 \\
0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{2}
\end{pmatrix}, \quad
\mathbf{T}_{\sin^2} c = \left(-\frac{1}{4}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0\right)^T = d,
\tag{2.30}
$$

while

$$
\mathbf{M}_{\sin^2} = \begin{pmatrix}
\frac{1}{2} & 0 & -\frac{1}{4} & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & -\frac{1}{4} & 0 & 0 \\
-\frac{1}{4} & 0 & \frac{1}{2} & 0 & -\frac{1}{4} & 0 \\
0 & -\frac{1}{4} & 0 & \frac{1}{2} & 0 & -\frac{1}{4} \\
0 & 0 & -\frac{1}{4} & 0 & \frac{1}{2} & 0 \\
0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{2}
\end{pmatrix}, \quad
\mathbf{M}_{\sin^2} c = \left(-\frac{1}{8}, 0, \frac{1}{8}, 0, \frac{1}{8}, 0\right)^T \neq d.
\tag{2.31}
$$

Similarly, the matrix for multiplication by $\cos\theta\sin\theta$ is the product of three matrices,

$$
\mathbf{T}_{\cos\sin} = \mathbf{Q}\mathbf{M}_{\cos\sin}(:, 3 : m + 3)\mathbf{P},
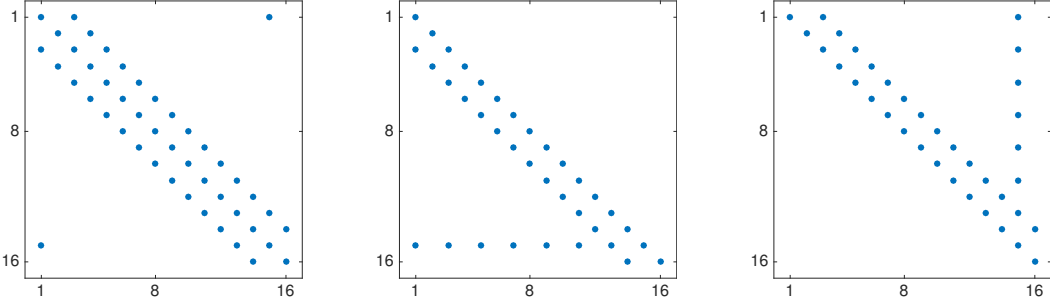\tag{2.32}
$$

with

$$\mathbf{M}_{\cos\sin} = \begin{pmatrix} 0 & 0 & \frac{i}{4} & & & & \\ 0 & 0 & 0 & \frac{i}{4} & & & \\ -\frac{i}{4} & 0 & \ddots & \ddots & \ddots & & \\ & -\frac{i}{4} & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & \frac{i}{4} \\ & & & \ddots & \ddots & \ddots & 0 \\ & & & & -\frac{i}{4} & 0 & 0 \end{pmatrix}, \tag{2.33}$$

and is given by

$$\mathbf{T}_{\cos\sin} = \begin{pmatrix} 0 & 0 & \frac{i}{4} & & & & -\frac{i}{4} & 0 \\ 0 & 0 & 0 & \frac{i}{4} & & & & 0 \\ -\frac{i}{8} & 0 & \ddots & \ddots & \ddots & & & \\ & -\frac{i}{4} & \ddots & \ddots & \ddots & \ddots & & \\ & & -\frac{i}{4} & \ddots & \ddots & \ddots & \frac{i}{4} & \\ & & & \ddots & \ddots & \ddots & \ddots & \frac{i}{4} \\ \frac{i}{8} & & & & -\frac{i}{4} & \ddots & 0 & 0 \\ 0 & 0 & & & & -\frac{i}{4} & 0 & 0 \end{pmatrix}. \tag{2.34}$$

2.3. Laplacian matrix and linear systems. The Laplacian operator on the sphere is

$$\Delta u = \frac{1}{\sin\theta}\big(\sin\theta\, u_\theta\big)_\theta + \frac{1}{\sin^2\theta} u_{\lambda\lambda}, \tag{2.35}$$

which we write as

$$\Delta u = u_{\theta\theta} + \frac{\cos\theta\sin\theta}{\sin^2\theta} u_\theta + \frac{1}{\sin^2\theta} u_{\lambda\lambda}. \tag{2.36}$$

We want to discretize Δ with a matrix $\mathbf{L}$ using a Fourier spectral method in coefficient space on an $n \times m$ uniform longitude-latitude grid (2.7), and we look for a solution of the form (2.5)–(2.6). Using Kronecker products, we can write $\mathbf{L}$ as

$$\mathbf{L} = \mathbf{I}_n \otimes (\mathbf{D}_m^{(2)} + \mathbf{T}_{\sin^2}^{-1}\mathbf{T}_{\cos\sin}\mathbf{D}_m) + \mathbf{D}_n^{(2)} \otimes (\mathbf{T}_{\sin^2}^{-1}), \tag{2.37}$$

where $\mathbf{T}_{\sin^2}$, $\mathbf{T}_{\cos\sin}$ and $\mathbf{D}_m$ have been defined in the previous section, $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\mathbf{D}_m^{(2)}$ is the second-order Fourier differentiation matrix,

$$\mathbf{D}_m^{(2)} = \mathrm{diag}\Big(-(m/2)^2, -(m/2-1)^2, \ldots, -1, 0, -1, \ldots, -(m/2-1)^2\Big). \tag{2.38}$$

Fig. 2.1: *Sparsity pattern of the matrix $z\mathbf{T}_{\sin^2} + w\mathbf{T}_{\sin^2}\mathbf{L}_i$ (left), and its* L *(middle) and* U *(right) factors for $m = 16$. Triangular systems involving* L *and* U *are solvable in $\mathcal{O}(m)$ operations.*

Note that the matrix $\mathbf{L}$ is block diagonal with n dense blocks of size $m \times m$. Let us emphasise that the n blocks correspond to the n longitudinal wavenumbers $-n/2 \le k \le n/2 - 1$ and that the size m of each block corresponds to the m latitudinal wavenumbers $-m/2 \le j \le m/2 - 1$.

Some of the time-stepping schemes we shall describe in Section 3 involve solving linear systems of the form $(z\mathbf{I}_{nm} + w\mathbf{L})x = b$, where $\mathbf{I}_{nm}$ denotes the $nm \times nm$ identity matrix. Fortunately, the matrix structure allows for a linear-cost direct solver. The key observation is that $\mathbf{L}$ is block diagonal, and each $m \times m$ block $\mathbf{L}_i$ of $\mathbf{L}$,

$$\mathbf{L}_i = \mathbf{D}_m^{(2)} + \mathbf{T}_{\sin^2}^{-1}\mathbf{T}_{\cos\sin}\mathbf{D}_m + \mathbf{D}_n^{(2)}(i,i)\mathbf{T}_{\sin^2}^{-1}, \qquad (2.39)$$

is dense but $(z\mathbf{I}_m + w\mathbf{L}_i)x = b$ can be solved in $\mathcal{O}(m)$ operations since it is equivalent to solving

$$(z\mathbf{T}_{\sin^2} + w\mathbf{T}_{\sin^2}\mathbf{L}_i)x = \mathbf{T}_{\sin^2}b, \qquad (2.40)$$

and $(z\mathbf{T}_{\sin^2} + w\mathbf{T}_{\sin^2}\mathbf{L}_i)$ is pentadiagonal with two (near-)corner elements. Therefore, using a sparse direct solver based on the standard LU factorization without pivots [14], the L and U factors have the sparsity patterns indicated in Figure 2.1 (due to the diagonal dominance in most blocks, the LU factorization without pivoting completes without breaking down[6]). Since L and U have at most three nonzero elements per column and row respectively, each triangular linear system is solvable in $\mathcal{O}(m)$ operations; thus once an LU factorization is computed, the linear system (2.40) can be solved in $\mathcal{O}(m)$ operations. Therefore, linear systems of the form

$$(z\mathbf{I}_{nm} + w\mathbf{L})x = b \qquad (2.41)$$

can be solved blockwise in $\mathcal{O}(nm)$ operations.[7] Because of the structure of the LU factors (see Figure 2.1), the coefficients one gets when solving systems of the form (2.41) have the property

---

[6]For the IMEX schemes of Section 3.2, we can prove that all the blocks but one are diagonally dominant; for ETDRK4-CF (see Section 3.1.1), the diagonal dominance is violated much more frequently. Nonetheless, diagonal dominance is merely a sufficient condition for the LU factorization to not require pivoting, and in practice, all the methods result in linear systems for which the LU factorization causes no stability issues.

[7]In practice we do not solve linear systems of the form (2.41) blockwise, i.e., by solving n linear systems (2.40). Instead, a more efficient approach is to store the left-hand sides of (2.40) altogether as a sparse matrix and use MATLAB's sparse linear solver.

that the even modes in λ correspond to even functions in θ and the odd modes to odd functions. To see this, note that every other term in the LU factors is zero, so the even and odd modes are decoupled.

2.4. Poisson's equation. To test the accuracy of (2.37), we first solve a *time-independent* PDE, *Poisson's equation*, with a zero-mean condition for uniqueness,

$$\Delta u = f(\lambda, \theta), \quad (\lambda, \theta) \in [-\pi, \pi] \times [0, \pi],$$

$$\int_0^\pi \int_{-\pi}^\pi u(\lambda, \theta) \sin \theta \, d\lambda \, d\theta = 0, \tag{2.42}$$

where f also has zero mean on $[-\pi, \pi] \times [0, \pi]$. Using the DFS method, we seek a solution $\tilde{u}$ of the "doubled-up" version of (2.42),

$$\Delta \tilde{u} = \tilde{f}(\lambda, \theta), \quad (\lambda, \theta) \in [-\pi, \pi]^2,$$

$$\int_0^\pi \int_{-\pi}^\pi \tilde{u}(\lambda, \theta) \sin \theta \, d\lambda \, d\theta = 0, \tag{2.43}$$

of the form (2.5)–(2.6). The true solution u can be recovered by restricting $\tilde{u}$ to $(\lambda, \theta) \in [-\pi, \pi] \times [0, \pi]$. (Note that the zero-mean condition in (2.43) is on the original domain $[-\pi, \pi] \times [0, \pi]$ since $\tilde{u}$ must coincide with u on this domain.) Townsend et al. [51] showed that the zero-mean condition can be discretized as

$$\int_0^\pi \int_{-\pi}^\pi \tilde{u}(\lambda, \theta) \sin \theta \, d\lambda \, d\theta \quad \approx \sum_{j=-m/2}^{m/2} {}' \sum_{k=-n/2}^{n/2} {}' \hat{u}_{jk} \int_0^\pi \sin \theta e^{ij\theta} d\theta \int_{-\pi}^\pi e^{ik\lambda} d\lambda$$

$$= 2\pi \sum_{j=-m/2}^{m/2} {}' \hat{u}_{j0} \frac{1 + e^{ij\pi}}{1 - j^2} = 0. \tag{2.44}$$

Poisson's equation (2.43) is then discretized by

$$\mathbf{L}\hat{u} = \hat{f}, \tag{2.45}$$

where $\hat{u}$ and $\hat{f}$ are the vectors of nm Fourier coefficients (2.6) of $\tilde{u}$ and $\tilde{f}$, and $\mathbf{L}$ is the Laplacian matrix (2.37). We impose the zero-mean condition by replacing the $(m/2+1)$st row of the $(n/2+1)$st block of $\mathbf{L}$ by (2.44).[8] Note that, given the Fourier coefficients of $\tilde{u}$ and $\tilde{f}$, (2.45) can be solved in $\mathcal{O}(nm)$ operations since it is of the form (2.41) with $z = 0$.

Since the Laplacian operator on the sphere has real eigenvalues $-l(l+1)$, $l \geq 0$, with eigenfunctions the spherical harmonics $Y_l^m(\lambda, \theta)$ [5], a simple test is to take the right-hand side f to be a spherical harmonic Y_l^m with exact solution $u = -1/(l(l+1))Y_l^m$. A slightly more complicated test is given in [11] and this is what we shall present here. We solve Poisson's equation for a family of right-hand sides f_l defined by

$$f_l(\lambda, \theta) = l(l+1) \sin^l \theta \cos(l\lambda) + (l+1)(l+2) \cos \theta \sin^l \theta \cos(l\lambda), \quad l \geq 1. \tag{2.46}$$

---

[8]The $(n/2 + 1)$st block of $\mathbf{L}$ corresponds to longitudinal wavenumber $k = 0$ while the $(m/2 + 1)$st row of this block corresponds to latitudinal wavenumber $j = 0$.
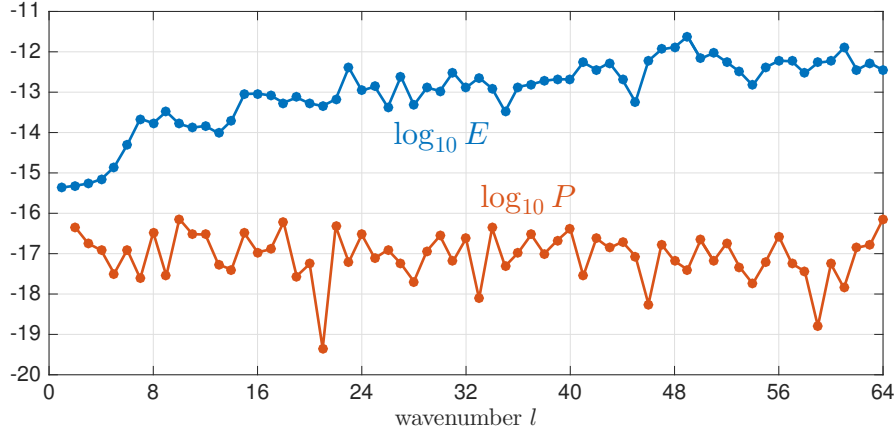
Fig. 2.2: *Variation of the relative error* $\log_{10} E$ *and of the error in the pole condition* $\log_{10} P$ *with wavenumber l for $m = n = 128$. The accuracy is excellent for every wavenumber $1 \leq l \leq 64$.*

The exact solution $u_l^{ex}(\lambda, \theta)$ is given by

$$u_l^{ex}(\lambda, \theta) = -\sin^l \theta \cos(l\lambda) - \cos \theta \sin^l \theta \cos(l\lambda), \quad l \geq 1. \tag{2.47}$$

We compute the solutions u_l for $m = n = 128$ grid points in each direction for $1 \leq l \leq 64$ and, following [11, Fig. 1], we plot the logarithm (in base 10) of the relative L^2-error E,

$$E = \frac{||u_l(\lambda, \theta) - u_l^{ex}(\lambda, \theta)||_2}{||u_l^{ex}(\lambda, \theta)||_2}, \tag{2.48}$$

with (continuous) L^2-norm on the sphere $|| \cdot ||_2$, against l. (The L^2-norm of a `spherefun` can be computed in Chebfun with the `norm` command.) We also plot the logarithm of the error P in satisfying the pole condition (2.8), i.e.,

$$P = \max \left(\max_{k \neq 0} \left| \sum_{j=-m/2}^{m/2}{}' \hat{u}_{jk}^l \right|, \max_{k \neq 0} \left| \sum_{j=-m/2}^{m/2}{}' (-1)^j \hat{u}_{jk}^l \right| \right), \tag{2.49}$$

where the $\hat{u}_{jk}^l$ are the computed coefficients of $\tilde{u}_l$; see Figure 2.2. The accuracy is excellent; the results are similar to those shown in [11, Fig. 1] and to results we have obtained with the Poisson solver of [51] (although the discretization matrices are different as noted in Section 2.2, the effect on the solution is negligible when m and n are taken large enough).

2.5. Heat equation. As we mentioned in the introduction, Orszag [37] and Boyd [8] showed that when solving a *time-dependent* PDE involving the Laplacian with half-range cosine/sine series à la (2.9), it is crucial to impose the pole conditions (2.10); otherwise, they observed that "the numerical solution still converged as the time step was shortened—to a wildly wrong answer." Therefore, let us now test the accuracy of (2.37) with a time-dependent PDE and illustrate that the pole conditions (2.8) are also satisfied in this case.
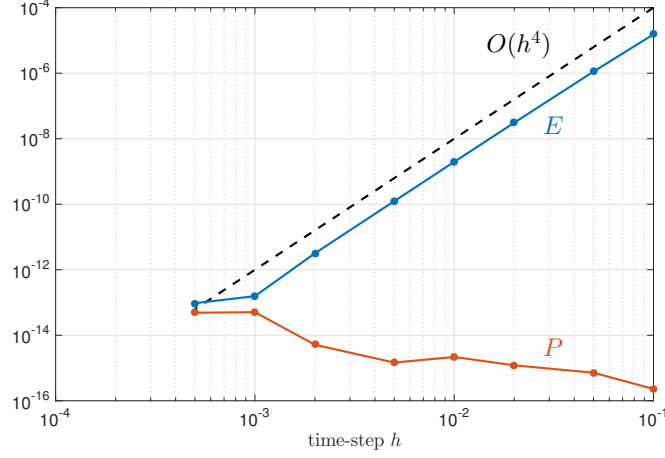
Fig. 2.3: *Variation of the relative error E at $t = 1$ and of the error in the pole condition P with time-step h for $m = n = 128$. The error E scales as $\mathcal{O}(h^4)$. Note that the initial condition $Y_{64}^{64}(\lambda, \theta)$ corresponds to the highest wavenumbers that can be resolved on a 128×128 grid.*

We consider the *heat equation* with a thermal diffusivity and an initial condition that lead to a particularly simple exact solution,

$$u_t = \frac{1}{l(l+1)}\Delta u, \quad u(t = 0, \lambda, \theta) = Y_l^m(\lambda, \theta), \quad (\lambda, \theta) \in [-\pi, \pi] \times [0, \pi]. \tag{2.50}$$

The exact solution is $u^{ex}(t, \lambda, \theta) = e^{-t}Y_l^m(\lambda, \theta)$. Using the DFS method, we seek a solution $\tilde{u}$ of the "doubled-up" version of (2.50),

$$\tilde{u}_t = \frac{1}{l(l+1)}\Delta\tilde{u}, \quad \tilde{u}(t = 0, \lambda, \theta) = Y_l^m(\lambda, \theta), \quad (\lambda, \theta) \in [-\pi, \pi]^2, \tag{2.51}$$

of the form

$$\tilde{u}(t, \lambda, \theta) \approx \sum_{j=-m/2}^{m/2}{}' \sum_{k=-n/2}^{n/2}{}' \hat{u}_{jk}(t)e^{ij\theta}e^{ik\lambda}. \tag{2.52}$$

We discretize the Laplacian operator with (2.37) and use the fourth-order backward differentiation formula to march in time. We take $l = 64$, $\tilde{u}(t = 0, \lambda, \theta) = Y_{64}^{64}(\lambda, \theta)$, $m = n = 128$ grid points and solve (2.51) up to $t = 1$ for various time-steps h. We plot the relative L^2-error E at $t = 1$,

$$E = \frac{||u(t = 1, \lambda, \theta) - u^{ex}(t = 1, \lambda, \theta)||_2}{||u^{ex}(t = 1, \lambda, \theta)||_2}, \tag{2.53}$$

and the error in the pole conditions (as defined in (2.49)) against h; the error E scales as $\mathcal{O}(h^4)$, see Figure 2.3. (With $m = n = 128$ grid points, the error due to the spatial discretization is small compared to the error due to the time discretization, so we are really measuring the latter.)

2.6. Bound for the eigenvalues of the Laplacian matrix. As mentioned in Section 2.4, the eigenvalues of the Laplacian operator are $-l(l+1)$, for integers $l \geq 0$. What about the eigenvalues of the Laplacian matrix (2.37)? We show in Appendix A that they are all real and nonpositive, and have observed numerically that some of them are spectrally accurate approximations to the eigenvalues $-l(l+1)$, but some others, the so-called *outliers* [52, Ch. 10], are of order $\mathcal{O}(n^2 m^2)$ as $n = m \to \infty$. We shall prove the latter fact below. These large eigenvalues are meaningless physically but of crucial importance in practice since for time-stepping algorithms applied to (1.2) to be stable, we need the eigenvalues of (2.37), scaled by the time-step, to lie in their stability region.

We examine now the largest (in magnitude) eigenvalues of (2.37). It suffices to examine each block

$$\mathbf{L}_i = \big(\mathbf{D}_m^{(2)} + \mathbf{T}_{\sin^2}^{-1}\mathbf{T}_{\cos\sin}\mathbf{D}_m + \mathbf{D}_n^{(2)}(i,i)\mathbf{T}_{\sin^2}^{-1}\big), \tag{2.54}$$

whose largest eigenvalue can be bounded as

$$|\lambda_{\max}(\mathbf{L}_i)| \leq \|\mathbf{L}_i\| \leq \|\mathbf{D}_m^{(2)}\| + \|\mathbf{T}_{\sin^2}^{-1}\|\|\mathbf{T}_{\cos\sin}\mathbf{D}_m + \mathbf{D}_n^{(2)}(i,i)\mathbf{I}_m\|. \tag{2.55}$$

We trivially have $\|\mathbf{D}_m^{(2)}\| = \mathcal{O}(m^2)$ and

$$\|\mathbf{T}_{\cos\sin}\mathbf{D}_m + \mathbf{D}_n^{(2)}(i,i)\mathbf{I}_m\| \leq \|\mathbf{T}_{\cos\sin}\mathbf{D}_m\| + |\mathbf{D}_n^{(2)}(i,i)| = \mathcal{O}(m + i^2). \tag{2.56}$$

It remains to bound $\|\mathbf{T}_{\sin^2}^{-1}\|$; we claim that this is $\mathcal{O}(m^2)$. To verify this, we come back to the definition of $\mathbf{T}_{\sin^2} = \mathbf{Q}\mathbf{M}_{\sin^2}(:, 3 : m + 3)\mathbf{P}$ from (2.23), and note that

$$\mathbf{Q}^T = \begin{pmatrix} \mathbf{0}_{2\times m} \\ \mathbf{P} \\ \mathbf{0}_{2\times m} \end{pmatrix} \mathrm{diag}(2, 1, \ldots, 1). \tag{2.57}$$

We can then write

$$\mathbf{T}_{\sin^2} = \mathbf{Q}\mathbf{M}_{\sin^2}\mathbf{Q}^T\mathrm{diag}(\tfrac{1}{2}, 1, \ldots, 1) = \mathrm{diag}(\sqrt{2}, 1, \ldots, 1)\tilde{\mathbf{Q}}\mathbf{M}_{\sin^2}\tilde{\mathbf{Q}}^T\mathrm{diag}(\tfrac{1}{\sqrt{2}}, 1, \ldots, 1), \tag{2.58}$$

where $\tilde{\mathbf{Q}} := \mathrm{diag}(\tfrac{1}{\sqrt{2}}, 1, \ldots, 1)\mathbf{Q}$ has orthonormal rows. Hence, using the fact that $\sigma_{\min}(\mathbf{ABC}) \geq \sigma_{\min}(\mathbf{A})\sigma_{\min}(\mathbf{B})\sigma_{\min}(\mathbf{C})$ (which holds when $\mathbf{A}$ and $\mathbf{C}$ are square), we obtain

$$\sigma_{\min}(\mathbf{T}_{\sin^2}) \geq \frac{1}{\sqrt{2}}\sigma_{\min}(\tilde{\mathbf{Q}}\mathbf{M}_{\sin^2}\tilde{\mathbf{Q}}^T). \tag{2.59}$$

Now, since $\mathbf{M}_{\sin^2}$ is symmetric positive definite, we have $\sigma_{\min}(\tilde{\mathbf{Q}}\mathbf{M}_{\sin^2}\tilde{\mathbf{Q}}^T) = \lambda_{\min}(\tilde{\mathbf{Q}}\mathbf{M}_{\sin^2}\tilde{\mathbf{Q}}^T) \geq \lambda_{\min}(\mathbf{M}_{\sin^2}(3 : m + 3, 3 : m + 3))$, where we used the nonzero structure of $\tilde{\mathbf{Q}}$ for the last inequality. Moreover, the eigenvalues of $\mathbf{M}_{\sin^2}(3 : m + 3, 3 : m + 3)$ are explicitly known to be

$$\lambda_j = \Big\{\tfrac{1}{2}\big(\cos(\pi j/(m/2+1))+1\big), \ 1 \leq j \leq \tfrac{m}{2}\Big\}\cup\Big\{\tfrac{1}{2}\big(\cos(\pi j/(m/2+2))+1\big), \ 1 \leq j \leq \tfrac{m}{2}+1\Big\}. \tag{2.60}$$

Thus $1/\sigma_{\min}(\tilde{\mathbf{Q}}\mathbf{M}_{\sin^2}\tilde{\mathbf{Q}}^T) \leq 1/\min_j \lambda_j = \mathcal{O}(m^2)$, and hence $\|\mathbf{T}_{\sin^2}^{-1}\| = 1/\sigma_{\min}(\mathbf{T}_{\sin^2}) = \mathcal{O}(m^2)$, as required. We conclude that $\|\mathbf{L}_i\| = \mathcal{O}(m^2(i^2 + m))$; since this holds for every i, it follows that

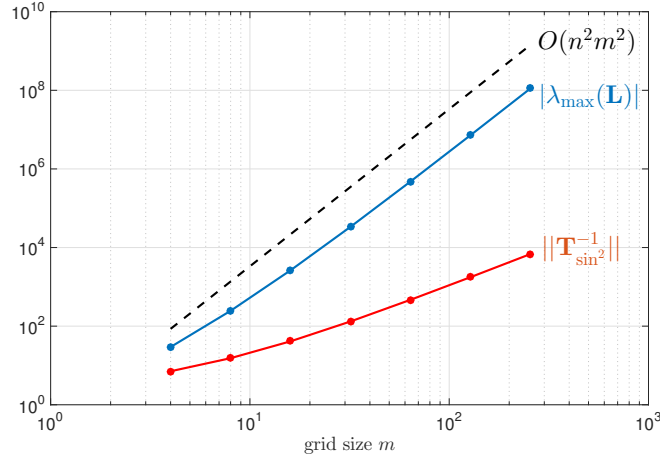$$|\lambda_{\max}(\mathbf{L})| = \mathcal{O}(n^2 m^2 + m^3). \tag{2.61}$$

Fig. 2.4: *Variation of $|\lambda_{\max}(\mathbf{L})|$ with $m = n$. The bound (2.61) is accurately reflected.*

This bound scales as $\mathcal{O}(n^2m^2)$ when $m = n$ (our usual choice). We illustrate (2.61) in Figure 2.4, which suggests that it is sharp.

The fact that the largest eigenvalue has order n^2m^2 makes time-dependent PDEs on the sphere particularly stiff. It is a consequence of both the second order of the Laplacian operator and the clustering of the points near the poles in (2.60). It would imply severe restrictions on the time-steps for generic explicit algorithms—this is why we use exponential integrators and IMEX schemes, which we describe next. (In the literature, the severe time-stepping restrictions due to uniform longitude-latitude grids is sometimes called the *pole problem* [8, 37]). Note that the time-stepping restrictions resulting from the clustering near the poles can be addressed by truncating high-frequency terms in the space discretization (see, e.g., [19, Sec. 2.1.6] and [22]). However, this approach does not overcome stiffness resulting from the second-order operator.

3. Fourth-order time-stepping on the sphere. Using the DFS method, we seek a solution $\tilde{u}$ of the "doubled-up" version of (2.3),

$$\tilde{u}_t = \alpha\Delta\tilde{u} + \mathcal{N}(\tilde{u}), \quad \tilde{u}(t = 0, \lambda, \theta) = \tilde{u}_0(\lambda, \theta), \quad (\lambda, \theta) \in [-\pi, \pi]^2, \tag{3.1}$$

of the form (2.52). Discretizing the Laplacian operator with the Laplacian matrix (2.37), we obtain the system of nm ODEs (1.2) where $\mathbf{L}$ is (2.37) multiplied by α. Time is discretized with time-step h and the problem is to find the Fourier coefficients $\hat{u}^{n+1}$ of $\tilde{u}$ at $t_{n+1} = (n+1)h$ from the coefficients $\hat{u}^n$ at $t_n = nh$ and also coefficients at previous time-steps (for multistep schemes). Note that, in practice, nonlinear evaluations $\mathbf{N}(\hat{u}^n)$ are carried out in value space.

We present in this section four time-stepping algorithms for solving (1.2), and show how it is possible to achieve $\mathcal{O}(nm\log nm)$ complexity per time-step in most cases. Two of them are exponential integrators based on the ETDRK4 scheme with different strategies for computing the matrix exponential and related functions, while the two others are IMEX schemes. As before, we have observed numerically that these schemes combined with our spatial discretization preserve both the "doubled-up" symmetry (2.4) and the pole conditions (2.8), i.e., if one starts with a smooth

"doubled-up" initial condition, then the solution at time t is also a smooth "doubled-up" function.

3.1. Exponential integrators. Dozens of exponential integration formulas of order four and higher have been proposed over the last 15 years [13, 26, 27, 31, 32, 34, 38]. The first author recently demonstrated [36] that it is hard to do much better than the ETDRK4 scheme of Cox and Matthews [13]. The formula for this scheme is:

$$\hat{a}^n = \varphi_0(h\mathbf{L}/2)\hat{u}^n + (h/2)\varphi_1(h\mathbf{L}/2)\mathbf{N}(\hat{u}^n),$$

$$\hat{b}^n = \varphi_0(h\mathbf{L}/2)\hat{u}^n + (h/2)\varphi_1(h\mathbf{L}/2)\mathbf{N}(\hat{a}^n),$$

$$\hat{c}^n = \varphi_0(h\mathbf{L}/2)\hat{a}^n + (h/2)\varphi_1(h\mathbf{L}/2)\big[2\mathbf{N}(\hat{b}^n) - \mathbf{N}(\hat{u}^n)\big], \tag{3.2}$$

$$\hat{u}^{n+1} = \varphi_0(h\mathbf{L})\hat{u}^n + hf_1(h\mathbf{L})\mathbf{N}(\hat{u}^n) + hf_2(h\mathbf{L})\big[\mathbf{N}(\hat{a}^n) + \mathbf{N}(\hat{b}^n)\big] + hf_3(h\mathbf{L})\mathbf{N}(\hat{c}^n),$$

where the φ-functions are defined by

$$\varphi_0(h\mathbf{L}) = e^{h\mathbf{L}},$$

$$\varphi_1(h\mathbf{L}) = h^{-1}\mathbf{L}^{-1}(e^{h\mathbf{L}} - \mathbf{I}),$$

$$\varphi_2(h\mathbf{L}) = h^{-2}\mathbf{L}^{-2}(e^{h\mathbf{L}} - h\mathbf{L} - \mathbf{I}), \tag{3.3}$$

$$\varphi_3(h\mathbf{L}) = h^{-3}\mathbf{L}^{-3}(e^{h\mathbf{L}} - h^2\mathbf{L}^2/2 - h\mathbf{L} - \mathbf{I}),$$

and the coefficients f_1, f_2 and f_3 are linear combinations of the φ-functions,

$$f_1(h\mathbf{L}) = \varphi_1(h\mathbf{L}) - 3\varphi_2(h\mathbf{L}) + 4\varphi_3(h\mathbf{L}) = h^{-3}\mathbf{L}^{-3}[-4\mathbf{I} - h\mathbf{L} + e^{h\mathbf{L}}(4 - 3h\mathbf{L} + (h\mathbf{L})^2)],$$

$$f_2(h\mathbf{L}) = 2\varphi_2(h\mathbf{L}) - 4\varphi_3(h\mathbf{L}) = 2h^{-3}\mathbf{L}^{-3}[2\mathbf{I} + h\mathbf{L} + e^{h\mathbf{L}}(-2\mathbf{I} + h\mathbf{L})], \tag{3.4}$$

$$f_3(h\mathbf{L}) = -\varphi_2(h\mathbf{L}) + 4\varphi_3(h\mathbf{L}) = h^{-3}\mathbf{L}^{-3}[-4\mathbf{I} - 3h\mathbf{L} - (h\mathbf{L})^2 + e^{h\mathbf{L}}(4\mathbf{I} - h\mathbf{L})].$$

When all the eigenvalues of $\mathbf{L}$ are real (i.e., $\alpha \in \mathbb{R}$, $\alpha > 0$ corresponding to a diffusive PDE), matrix-vector products $\varphi_l(h\mathbf{L})v$ can be evaluated using rational approximations computed by the Carathéodory–Fejér (CF) method, as described in [45]. If $\mathbf{L}$ has some imaginary eigenvalues— the extreme case being when all the eigenvalues are imaginary (i.e., $\alpha \in i\mathbb{R}$, dispersive PDE)— methods based on rational approximations necessarily become expensive, and the φ-functions have to be precomputed before the time-stepping starts, e.g., using the eigenvalue decomposition of $\mathbf{L}$ and contour integrals.[9] We denote by ETDRK4-CF the method with CF approximations and by ETDRK4-EIG the method with eigenvalue decomposition. We shall give details about the precomputation of the coefficients for both ETDRK4-CF and ETDRK4-EIG below.

Note that Du and Zhu computed in [17] the stability region of (3.2) and showed that it includes parts of both the negative real axis and the imaginary axis.

---

[9]A comparison of methods for computing the φ-functions can be found in [4].

3.1.1. ETDRK4-CF. When all the eigenvalues of $\mathbf{L}$ are real, one can compute matrix-vector products $\varphi_l(h\mathbf{L})v$ in $\mathcal{O}(mn)$ operations using near-best rational approximations to the φ-functions on the negative real axis [45]. For an efficient implementation, we use the algorithm that uses common poles for approximating the exponential and other φ-functions [45], as we summarize below. Using the CF method for the negative real line [53], we obtain a rational approximant

$$e^z \approx r_\infty + \sum_{j=1}^{p} \frac{c_j}{z - z_j}, \tag{3.5}$$

which has error decaying like $\approx 9.28903^{-p}$ with a type (p, p) function. (We use the MATLAB `cf` code of Trefethen, Weideman and Schmelzer [54] in our experiments.) To obtain an approximant to $\varphi_l(z)$ using the approximant (3.5) to $e^z = \varphi_0(z)$, we use the fact [45, Prop. 4.1] that defining

$$B_z = \begin{pmatrix} z & 1 \\ 0 & 0 \end{pmatrix} \tag{3.6}$$

we have

$$\varphi_l(B_z) = \begin{pmatrix} \varphi_l(z) & \varphi_{l+1}(z) \\ 0 & \varphi_l(0) \end{pmatrix}, \quad l \geq 0. \tag{3.7}$$

Together with the identity

$$(B_z - z_j I)^{-1} = \begin{pmatrix} (z - z_j)^{-1} & (z - z_j)^{-1} z_j^{-1} \\ 0 & -z_j^{-1} \end{pmatrix}, \tag{3.8}$$

we obtain the approximation

$$\varphi_l(z) \approx \sum_{j=1}^{p} \frac{c_j z_j^{-l}}{z - z_j}, \quad l \geq 0. \tag{3.9}$$

As suggested in [45, Prop. 4.1], we further incorporate a shift 1 in (3.5), which with $p = 12$ (the default choice; the accuracy is $\approx 10^{-8}$ with $p = 10$) gives accuracy $\approx 10^{-10}$ on the negative real axis for all φ_l, $0 \leq l \leq 3$. Given (3.5) and (3.9), evaluating $\varphi_l(h\mathbf{L})b$ for a vector b as in (3.2) can be approximated as

$$\varphi_l(h\mathbf{L})b \approx \sum_{j=1}^{p} c_j z_j^{-l}(h\mathbf{L} - z_j\mathbf{I})^{-1}b, \quad 0 \leq l \leq 3, \tag{3.10}$$

which reduces to $p = 12$ shifted linear systems of the form (2.41), which we do with linear cost as described in Section 2.3. In practice, we compute and store the LU factorizations of the matrices that appear in (3.10) before the time-stepping starts. Note that computing different products $\varphi_l(hL)v$ at once with the same v requires no further linear systems.

Let us emphasize three aspects of this approach. First, it is not necessary to explicitly compute and store the φ-functions; instead, their action on vectors is directly computed via (3.10). Second,
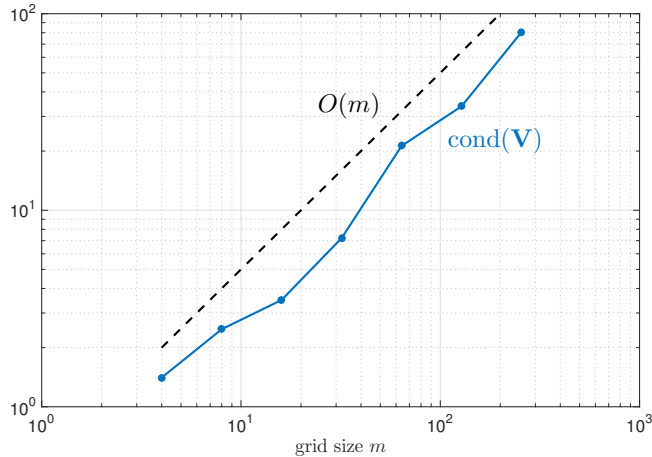
Fig. 3.1: *Variation of* $\mathrm{cond}(\mathbf{V})$ *with* $m = n$. *It is of order* m *and reasonably small; therefore, an approach based on eigenvalues and eigenvectors for the computation of the* φ-*functions is valid.*

the most expensive operation in (3.2)–(3.10) is the 2D FFT, which costs $\mathcal{O}(nm \log nm)$ operations; see Table 3.1. Third, this method is not applicable when $\mathbf{L}$ has some imaginary eigenvalues; this is because low-degree rational functions are unable to approximate the exponential on the imaginary axis, which is oscillatory. In this case one has to compute and store the φ-functions, and the complexity increases to $\mathcal{O}(nm^2)$ per time-step, as we describe next.

3.1.2. ETDRK4-EIG. To compute the φ-functions, one can use a method based on eigen-values and eigenvectors. The idea is to diagonalize $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$ and then apply the φ-functions to the eigenvalues,

$$\varphi_l(h\mathbf{L}) = \mathbf{V}\varphi_l(h\mathbf{\Lambda})\mathbf{V}^{-1}, \quad 0 \leq l \leq 3, \tag{3.11}$$

with

$$\varphi_l(h\mathbf{\Lambda}) = \begin{pmatrix} \varphi_l(h\lambda_1) & & & \\ & \varphi_l(h\lambda_2) & & \\ & & \ddots & \\ & & & \varphi_l(h\lambda_{nm}) \end{pmatrix}. \tag{3.12}$$

For a general $nm \times nm$ matrix $\mathbf{L}$, this would require $\mathcal{O}(n^3m^3)$ operations, but for (2.37), this can be done blockwise in $\mathcal{O}(nm^3)$ operations. Note that this corresponds to Method 14 of [35]. Theoretically, this approach only works when $\mathcal{L}$ is nondefective, that is, when it has a complete set of linearly independent eigenfunctions—this is a well known result for the Laplacian operator on the sphere [5]. In practice, difficulties occur when the discretizaion $\mathbf{L}$ is "nearly" defective, i.e., when $\mathrm{cond}(\mathbf{V}) = ||\mathbf{V}|| \, ||\mathbf{V}^{-1}||$ is large. Fortunately, we have observed numerically that the condition number is small and of order m as $m = n$ increases; see Figure 3.1.

Once we have computed the eigenvalue decomposition, we follow the idea of Kassam and Trefethen [29] and evaluate the φ-functions at each scaled eigenvalue $h\lambda$ using Cauchy's integral

formula,

$$\varphi_l(h\lambda) = \frac{1}{2\pi i} \oint_\Gamma \frac{\varphi_l(z)}{z - h\lambda} dz \approx \frac{1}{M} \sum_{k=1}^{M} \varphi_l\big(h\lambda + e^{2\pi i (k-0.5)/M}\big), \quad 0 \le l \le 3. \tag{3.13}$$

The constant M is the number of points in the discretized contour integration; we take $M = 32$ in our experiments.

The precomputation step costs $\mathcal{O}(nm^3)$ operations, while the cost per time-step is $\mathcal{O}(nm^2)$ since one has to compute block diagonal matrix-vector products $\varphi_l(h\mathbf{L})v$; see Table 3.1.

3.2. Implicit-explicit schemes. We present in this section the two IMEX schemes we consider in this paper. The first one, IMEX-BDF4 [3], is a multistep scheme which is stable only for diffusive PDEs. The second one, LIRK4 [9], is a one-step scheme, stable for both diffusive and dispersive PDEs.

3.2.1. IMEX-BDF4. Following Kassam and Trefethen [29], we consider a scheme known either as SBDF4 (in [3]), AB4BD4 (in [13]) or IMEX-BDF4 (in [28]), which combines a fourth-order Adams–Bashforth formula and a fourth-order backward differentiation scheme in a nontrivial way. The method is given by:

$$(25\mathbf{I}_{nm} - 12h\mathbf{L})\hat{u}^{n+1} = 48\hat{u}^n - 36\hat{u}^{n-1} + 16\hat{u}^{n-2} - 3\hat{u}^{n-3} + 48h\mathbf{N}(\hat{u}^n) - 72h\mathbf{N}(\hat{u}^{n-1})$$
$$+ 48h\mathbf{N}(\hat{u}^{n-2}) - 12h\mathbf{N}(\hat{u}^{n-3}). \tag{3.14}$$

At each time-step, one has to solve a linear system to get the Fourier coefficients $\hat{u}^{n+1}$, which we do with linear cost by multiplying each block of (3.14) by $\mathbf{T}_{\sin^2}$, as explained in Section 2.3. Therefore, the dominant cost in (3.14) is the $\mathcal{O}(nm \log nm)$ 2D FFT for the nonlinear evaluations; see Table 3.1 at the end of this section.

Let us add three comments about (3.14). First, the LU factorization of the left-hand side of (3.14) is computed and stored before the time-stepping starts. Second, this is a multistep formula so it has to be started with a one-step scheme—in the numerical comparisons of Section 4, we initialize it with three steps of ETDRK4-CF. Third, it is unstable for dispersive PDEs since the stability region of the fourth-order backward differentiation formula does not contain the portion of the imaginary axis near the origin. In these cases, one can use IMEX Runge–Kutta schemes [9, 30, 39] or extrapolation-based IMEX schemes [10, 12]. We have decided to focus on the former.

3.2.2. LIRK4. IMEX Runge–Kutta schemes combine explicit Runge–Kutta formulas to advance the nonlinear part and implicit Runge–Kutta to advance the linear part [9, 30, 39]. In this paper, we use the fourth-order LIRK4 scheme of Calvo, de Frutos and Novo [9]. It combines an implicit L-stable five-stage fourth-order Runge–Kutta method, whose Butcher tableau is given in [24, Table 6.5], with a six-stage fourth-order explicit Runge–Kutta method, whose coefficients were

Table 3.1: *Computational costs (per time-step) of the time-stepping algorithms with $p = 12$ in the CF approximation. The IMEX-BDF4 scheme is particularly cheap while for ETDRK4-CF one needs to solve an extremely large number of linear systems. ETDRK4-EIG is the only scheme that has a $\mathcal{O}(nm^2)$ cost per time-step and a $\mathcal{O}(nm^3)$ precomputation step. Precomputations for the other schemes (LU factorizations) cost $\mathcal{O}(nm)$ operations.*

	ETDRK4		IMEX	
	CF	EIG	BDF4	LIRK4
# $\mathcal{O}(nm \log nm)$ **FFTs**	8	8	2	12
# $\mathcal{O}(nm)$ **linear solves**	$9p = 108$	0	1	5
# $\mathcal{O}(nm^2)$ **matrix-vector products**	0	9	0	0
diffusive PDEs	✓	✓	✓	✓
dispersive PDEs	✗	✓	✗	✓

derived in [9]. The formula for this scheme is:

$$(\mathbf{I}_{nm} - \tfrac{1}{4}h\mathbf{L})\hat{a}^n = \hat{u}^n + \tfrac{1}{4}h\mathbf{N}(\hat{u}^n),$$

$$(\mathbf{I}_{nm} - \tfrac{1}{4}h\mathbf{L})\hat{b}^n = \hat{u}^n + \tfrac{1}{2}h\mathbf{L}\hat{a}^n - \tfrac{1}{4}h\mathbf{N}(\hat{u}^n) + h\mathbf{N}(\hat{a}^n),$$

$$(\mathbf{I}_{nm} - \tfrac{1}{4}h\mathbf{L})\hat{c}^n = \hat{u}^n + \tfrac{17}{50}h\mathbf{L}\hat{a}^n - \tfrac{1}{25}h\mathbf{L}\hat{b}^n - \tfrac{13}{100}h\mathbf{N}(\hat{u}^n) + \tfrac{43}{75}h\mathbf{N}(\hat{a}^n) + \tfrac{8}{75}h\mathbf{N}(\hat{b}^n),$$

$$(\mathbf{I}_{nm} - \tfrac{1}{4}h\mathbf{L})\hat{d}^n = \hat{u}^n + \tfrac{371}{1360}h\mathbf{L}\hat{a}^n - \tfrac{137}{2720}h\mathbf{L}\hat{b}^n + \tfrac{15}{544}h\mathbf{L}\hat{c}^n - \tfrac{6}{85}h\mathbf{N}(\hat{u}^n) + \tfrac{42}{85}h\mathbf{N}(\hat{a}^n),$$

$$+ \tfrac{179}{1360}h\mathbf{N}(\hat{b}^n) - \tfrac{15}{272}h\mathbf{N}(\hat{c}^n),$$

(3.15)

$$(\mathbf{I}_{nm} - \tfrac{1}{4}h\mathbf{L})\hat{e}^n = \hat{u}^n + \tfrac{25}{24}h\mathbf{L}\hat{a}^n - \tfrac{49}{48}h\mathbf{L}\hat{b}^n + \tfrac{125}{16}h\mathbf{L}\hat{c}^n - \tfrac{85}{12}h\mathbf{L}\hat{d}^n + \tfrac{79}{24}h\mathbf{N}(\hat{a}^n) - \tfrac{5}{8}h\mathbf{N}(\hat{b}^n)$$
$$+ \tfrac{25}{2}h\mathbf{N}(\hat{c}^n) - \tfrac{85}{6}h\mathbf{N}(\hat{d}^n),$$

$$\hat{u}^{n+1} = \hat{u}^n + \tfrac{25}{24}h\mathbf{L}\hat{a}^n - \tfrac{49}{48}h\mathbf{L}\hat{b}^n + \tfrac{125}{16}h\mathbf{L}\hat{c}^n - \tfrac{85}{12}h\mathbf{L}\hat{d}^n + \tfrac{1}{4}h\mathbf{L}\hat{e}^n + \tfrac{25}{24}h\mathbf{N}(\hat{a}^n) - \tfrac{49}{48}h\mathbf{N}(\hat{b}^n)$$

$$+ \tfrac{125}{16}h\mathbf{N}(\hat{c}^n) - \tfrac{85}{12}h\mathbf{N}(\hat{d}^n) + \tfrac{1}{4}h\mathbf{N}(\hat{e}^n).$$

The LU factorization of the left-hand sides of (3.15) is computed and stored before the time-

stepping starts. Note that the most expensive operations in the computation of the internal stages $\hat{a}^n$, $\hat{b}^n$, $\hat{c}^n$, $\hat{d}^n$ and $\hat{e}^n$ are the nonlinear evaluations ($\mathcal{O}(nm \log nm)$ work). The other operations, i.e., linear solves and matrix-vector products in the right-hand side (each block being multiplied by $\mathbf{T}_{\sin^2}$), can be carried out in linear time. The computation of $\hat{u}^{n+1}$ from $\hat{u}^n$ requires matrix-vector products of the form $\mathbf{L}v$, which reduce to n dense matrix-vector products $\mathbf{L}_i v$; each can be done in $\mathcal{O}(m)$ operations since $\mathbf{L}_i v = \mathbf{T}_{\sin^2}^{-1}(\mathbf{T}_{\sin^2}\mathbf{L}_i)v$. (The LU factorization of $\mathbf{T}_{\sin^2}$ is also computed and stored before the time-stepping starts.)

4. Numerical comparisons.

4.1. Methodology. To compare time-stepping schemes, we follow the methodology of [29]. We solve a given PDE up to $t = T$ for various time-steps h and a fixed number of grid points. We estimate the "exact" solution $u^{ex}(t = T, \lambda, \theta)$ by using a very small time-step (half the smallest time-step h) and ETDRK4-EIG (the most accurate time-stepping scheme). We then measure the relative L^2-error E at $t = T$ between the computed solution $u(t = T, \lambda, \theta)$ and $u^{ex}(t = T, \lambda, \theta)$,

$$E = \frac{\|u(t = T, \lambda, \theta) - u^{ex}(t = T, \lambda, \theta)\|_2}{\|u^{ex}(t = T, \lambda, \theta)\|_2}. \tag{4.1}$$

For both u and u^{ex} we use $m = n = 256$ grid points. (With these grid sizes, the error due to the spatial discretization is small compared to the error due to the time discretization.) We use $p = 12$ in the CF approximation for ETDRK4-CF and $M = 32$ points to compute the contour integrals (3.13) for ETDRK4-EIG. We plot (4.1) against relative time-steps h/T and computer times on a pair of graphs.[10] The former gives a measure of the accuracy of the time-stepping scheme for various time-steps or, equivalently, for various number of integration steps. (If the relative time-step is 10^{-3}, it means that the scheme performed 10^3 steps to reach $t = T$.) However, it is possible that each step is more costly, so it is the latter that ultimately matters.

4.2. Results for the diffusive case. The *Allen–Cahn equation*, derived by Allen and Cahn in the 1970s, is a reaction-diffusion equation which describes the process of phase separation in iron alloys [2], studied in the ball and on the sphere in, e.g., [18]. It is given by

$$u_t = \epsilon \Delta u + u - u^3, \quad \epsilon \ll 1, \tag{4.2}$$

with linear diffusion $\epsilon \Delta u$ and a cubic reaction term $u - u^3$. The function u is the order parameter, a correlation function related to the positions of the different components of the alloy. The Allen–Cahn equation exhibits stable equilibria at $u = \pm 1$, while $u = 0$ is an unstable equilibrium. Solutions often display metastability where wells $u \approx -1$ compete with peaks $u \approx 1$, and structures remain almost unchanged for long periods of time before changing suddenly. We take $\epsilon = 10^{-2}$ and

$$u(t = 0, x, y, z) = \cos(\cosh(5xz) - 10y), \tag{4.3}$$

and solve up to $t = 10$. The initial condition and the solution at times $t = 1, 2, 10$ are shown in Figure 4.1. The initial condition quickly converges to a metastable $u \approx \pm 1$ solution (at around $t = 10$) and eventually to the stable constant solution $u = 1$ (at around $t = 60$).

---

[10]The precomputation of the coefficients of the exponential integrators, the LU factorizations for the IMEX schemes and the starting phase of IMEX-BDF4 are not included in the computing time. Timings were done on a 2.8 GHz Intel i7 machine with 16 GB of RAM using MATLAB R2015b and Chebfun v5.6.0.
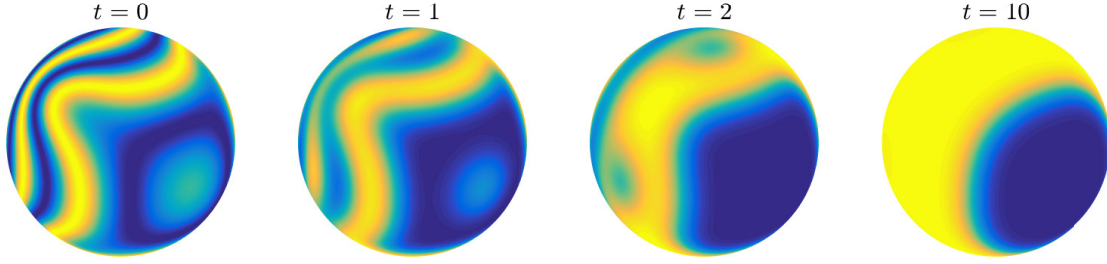
Fig. 4.1: *Initial condition* (4.3) *and solution at times* $t = 1, 2, 10$ *of the Allen–Cahn equation* (4.2).
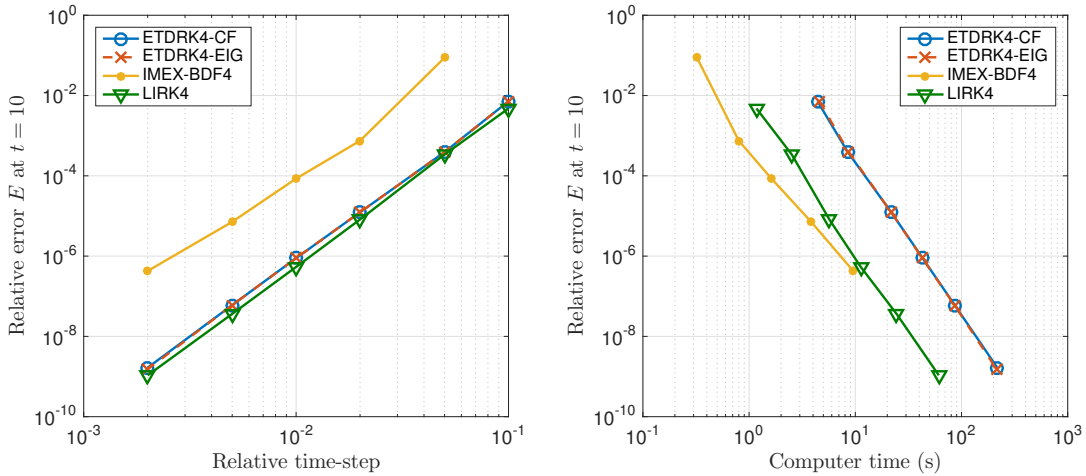

Fig. 4.2: *Relative error* E *at* $t = 10$ *versus relative time-step and computer time for the Allen–Cahn equation* (4.2).

The results are shown in Figure 4.2. All the schemes are stable for the time-steps we have considered, except IMEX-BDF4 for the largest time-step. The ETDRK4 schemes and LIRK4 have similar accuracy, while IMEX-BDF4 is significantly less accurate. However, IMEX-BDF4 is the most efficient scheme. This can be explained by looking at Table 3.1: IMEX-BDF4 requires very few operations per time-step. Note that in this experiment, ETDRK4-CF ($\mathcal{O}(nm \log nm)$ work per time-step) is not more efficient than ETDRK4-EIG ($\mathcal{O}(nm^2)$). Again, the reason can be found in Table 3.1: ETDRK4-CF requires the solution of 108 linear systems per time-step. (For $m = n$ sufficiently large, ETDRK4-CF will be indeed more efficient than ETDRK4-EIG.)

4.3. Results for the dispersive case. The *nonlinear Schrödinger (NLS) equation*,
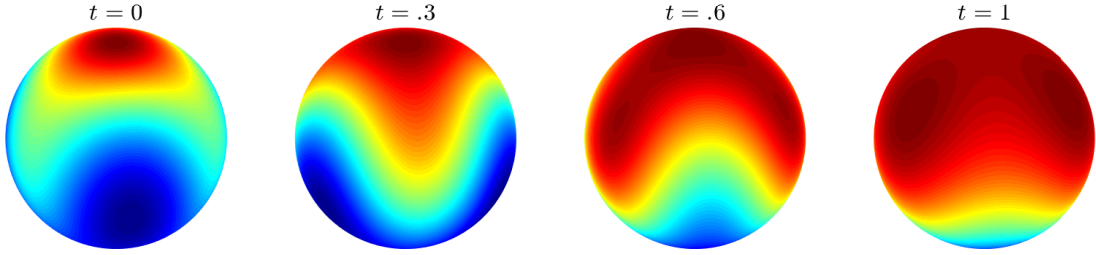
$$u_t = i\Delta u + i|u|^2 u, \tag{4.4}$$

Fig. 4.3: *Initial condition* (4.5) *and real part of the solution at times* $t = 0.3, 0.6, 1$ *of the NLS equation* (4.4).
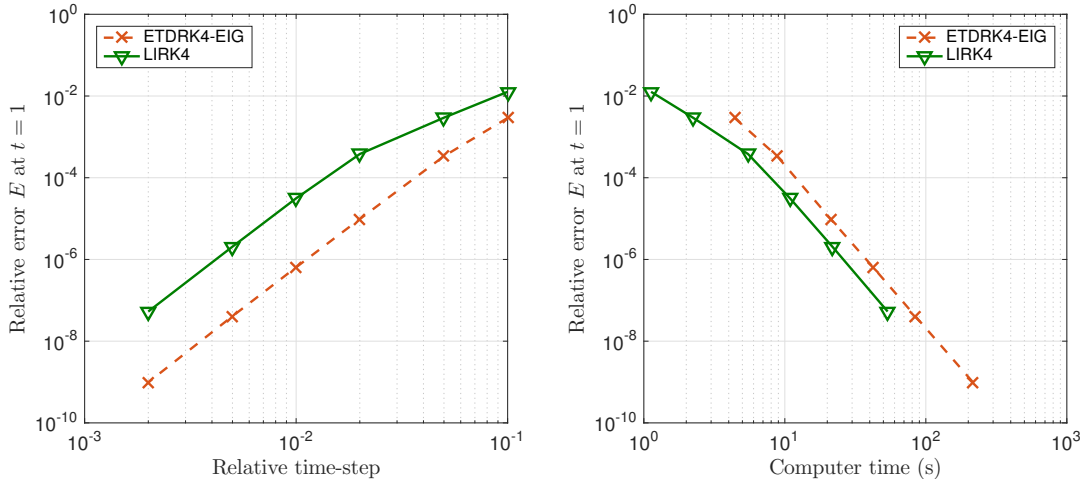

Fig. 4.4: *Relative error E at* $t = 1$ *versus relative time-step and computer time for the NLS equation.*

models a variety of physical phenomena, including the propagation of light in optical fibres; on the sphere, a recent theoretical study of its solutions can be found in [49]. A nonlinear variant of the Schrödinger equation, it couples dispersion $i\Delta u$ with a nonlinear potential $i|u|^2 u$. Note that the wave function u is complex-valued. We take

$$u(t = 0, \lambda, \theta) = A\left(\frac{2B^2}{2 - \sqrt{2}\sqrt{2 - B^2}\cos(AB\theta)} - 1\right) + Y_3^3(\lambda, \theta) \qquad (4.5)$$

with $A = B = 1$ and solve up to $t = 1$. The initial condition and the real part of the solution at times $t = 0.3, 0.6, 1$ are shown in Figure (4.3). The initial condition is the superposition of two nonlinear waves in which energy concentrates in a localized and oscillatory fashion, a *breather* and a spherical harmonic.

The results are shown in Figure 4.4. Both schemes are stable for the time-steps we have

considered. LIRK4 is less accurate than ETDRK4-EIG in this case, but since it has a much lower cost per time-step, it is more efficient. Let us emphasize that timings do not include the precomputation step, which takes significantly longer for ETDRK4-EIG ($\mathcal{O}(nm^3)$ versus $\mathcal{O}(nm)$). Also, since LIRK4 and ETDRK4-EIG have different scaling in $m = n$, the results ultimately depend on the size of the discretization. For example, for $m = n = 128$ instead of 256, ETDRK4-EIG is slightly more efficient.

5. Discussion. We have presented algorithms for solving stiff PDEs on the sphere with spectral accuracy in space and fourth-order in time. For the spatial discretization, we have used a variant of the DFS method in coefficient space. The main advantages of our method are that it is not necessary to numerically impose the pole conditions (2.8), while operating in coefficient space avoids the coordinate singularity without using shifted grids and leads to matrices $\mathbf{L}$ that can be computed and inverted efficiently. We have tested our method with the Poisson and heat equations and obtained excellent results.

For solving nonlinear time-dependent PDEs, we have used IMEX schemes and exponential integrators to circumvent the time-stepping restrictions due to the large eigenvalues of the Laplacian matrix. For diagonal problems, exponential integrators are particularly efficient since the computation and the action of the matrix exponential can trivially be computed in linear time. For problems that allow for fast numerical linear algebra (fast sparse direct solver), as in this paper, we have given numerical evidence that IMEX schemes outperform exponential integrators. The IMEX-BDF4 time-stepping scheme is remarkably efficient for diffusive PDEs but since it is unstable for dispersive PDEs and needs to be started with another algorithm, it might be preferable to use LIRK4 for both diffusive and dispersive PDEs. For problems that generate dense matrices $\mathbf{L}$, it is not clear which one would perform best. On a grid with N points, both exponential integrators (dense matrix-vector products) and IMEX schemes (triangular systems from precomputed LU factorizations) would have a $\mathcal{O}(N^2)$ cost per time-step. Note that dense matrices correspond to, e.g., the DFS method applied to (1.1) with highly oscillatory variable coefficients or a Chebyshev discretization in value space of PDEs of the form (1.1) in 1D/2D/3D. However, as recently shown by Aurentz in [6], it seems that all spectral differentiation matrices (even the dense ones!) might allow for fast numerical linear algebra, which would render IMEX schemes more efficient for value-based Chebyshev discretizations—this is a story to be continued. We summarize these observations in Table 5.1.

We have not considered the "sliders" of Fornberg and Driscoll [15, 20]. While this can be an efficient alternative to IMEX schemes and exponential integrators for diagonal problems [29], it is not clear how it can be extended to non-diagonal ones.

Our method can in principle deal with nonsmooth initial conditions for diffusive problems, as long as the diffusion is strong enough to smooth out the solution. Also, as we mentioned in the introduction, our method could be applied to more general PDEs, including linear operators consisting of powers of the Laplacian operator. These would have a larger bandwidth, but could still be inverted efficiently. Therefore, we believe that the same conclusions would hold. Analogues of the matrices $\mathbf{P}$ and $\mathbf{Q}$ would be involved.

Future directions include the application of our method to hyperbolic problems, e.g., the barotropic vorticity equation or the shallow water equations. Such problems involve nonlinear differential operators with large eigenvalues. While stiffness in the linear part (as in the Allen–Cahn and NLS equations) can be treated by using IMEX schemes or exponential integrators, it is

Table 5.1: *Computation costs per time-step for a grid with N points and most efficient time-stepping scheme in each case. Diagonal problems were considered in* [29]. *In this paper, we investigated non-diagonal problems that allow for fast numerical linear algebra (i.e., linear systems can be solved in linear time). In the latter case, IMEX schemes outperform exponential integrators. It is not clear which scheme would perform best in the dense numerical linear algebra case.*

	diffusive PDEs	dispersive PDEs
diagonal problems	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$
	ETDRK4	ETDRK4
non-diagonal problems	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$
fast sparse direct solver	IMEX-BDF4	LIRK4
non-diagonal problems	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
dense solver	TBD	TBD

not obvious how to deal with a stiff nonlinear operator. For the barotropic vorticity equation,

$$u_t = \mathcal{N}(u) = -\frac{(\Delta^{-1}u)_\theta}{\sin\theta}u_\lambda + \frac{(\Delta^{-1}u)_\lambda}{\sin\theta}(u_\theta - 2\Omega\sin\theta), \tag{5.1}$$

a possible approach would be to use the EPIRK schemes [50], e.g., the EPIRK2 scheme given by

$$\hat{u}^{n+1} = \hat{u}^n + \mathbf{J}^{-1}(e^{h\mathbf{J}} - \mathbf{I})\mathbf{N}(\hat{u}^n), \quad \mathbf{J} = \frac{d\mathbf{N}}{d\hat{u}}(\hat{u}),$$

with Arnoldi iteration for the matrix-vector products involving the Jacobian matrix $\mathbf{J}$ of the discretization $\mathbf{N}$ of $\mathcal{N}$ in Fourier space—the cost per time-step would also be $\mathcal{O}(N \log N)$ operations.

Appendix A. Eigenvalues of the Laplacian matrix. The Laplacian matrix (2.37) is a discretized Laplacian, so one might expect that the eigenvalues are all real and nonpositive. For example, Gottlieb and Lustman [23] give a nontrivial proof that the discretization of the second derivative operator in the Chebyshev collocation method on a real interval with separated boundary conditions has real and negative eigenvalues. Here we show that essentially the same holds on the sphere for (2.37) (a slight difference is that we have one zero eigenvalue since the Laplacian of a constant is zero).

THEOREM A.1. *The eigenvalues of $\mathbf{L}$ in* (2.37) *are all real and nonpositive.*

Proof. Clearly it suffices to examine each ith block $\mathbf{L}_i = \mathbf{D}_m^{(2)} + \mathbf{T}_{\sin^2}^{-1}\mathbf{T}_{\cos\sin}\mathbf{D}_m + \mathbf{D}_n^{(2)}(i,i)\mathbf{T}_{\sin^2}^{-1}$. We first note that the eigenvalues of $\mathbf{L}_i$ are equal to those of the matrix pencil

$$\mathbf{A}_i - \lambda\mathbf{B}_i := \mathbf{T}_{\sin^2}\mathbf{D}_m^{(2)} + \mathbf{T}_{\cos\sin}\mathbf{D}_m + \mathbf{D}_n^{(2)}(i,i)\mathbf{I}_m - \lambda\mathbf{T}_{\sin^2}, \tag{A.1}$$

which corresponds to the generalized eigenvalue problem $\mathbf{A}_i x = \lambda\mathbf{B}_i x$. We shall prove that this

pencil has negative real eigenvalues, regardless of i; we drop the subscript i for simplicity. Our proof proceeds as follows:

1. The eigenvalues of $\mathbf{A} - \lambda\mathbf{B}$ are the values of λ_0 for which the matrix $\mathbf{A} - \lambda_0\mathbf{B}$ has a zero eigenvalue.
2. For any fixed $\lambda_0 \in (-\infty, 0]$, all the eigenvalues of the matrix $\mathbf{A} - \lambda_0\mathbf{B}$ are real. Therefore we can define m real continuous functions $f_j(\lambda_0) := \lambda_j(\mathbf{A} - \lambda_0\mathbf{B})$ for $j = 1, \ldots, m$.
3. For every j, we have $f_j(0) \le 0$, and $f_j(\lambda_0) \ge 0$ for negative λ_0 with sufficiently large $|\lambda_0|$.
4. By the intermediate value theorem to each $f_j(\lambda_0)$, there is at least one root $f_j(\lambda_0) = 0$ in $\lambda_0 \in (-\infty, 0]$ for each j. It follows that $\mathbf{A} - \lambda\mathbf{B}$ has m real eigenvalues (counting multiplicities), hence so does $\mathbf{L}_i$.

The only nontrivial parts are the second step, and the claim $f_j(0) \le 0$.

We first prove the second step, that the matrix $\mathbf{C}(\lambda_0) := \mathbf{A} - \lambda_0\mathbf{B}$ has only real eigenvalues. To do this we apply the similarity transformation with the permutation matrix $\mathbf{P} = \mathbf{I}_m(:, [1 : 2 : m, 2 : 2 : m])$, which gives $\mathbf{P}^T\mathbf{C}(\lambda_0)\mathbf{P} = \mathrm{diag}(\mathbf{C}_1(\lambda_0), \mathbf{C}_2(\lambda_0))$, a block-diagonal matrix with two $\frac{m}{2} \times \frac{m}{2} := \ell \times \ell$ blocks. Here $\mathbf{C}_1(\lambda_0)$ is tridiagonal with extra elements in the upper-right and lower-left corners (as in (A.3) below), and $\mathbf{C}_2(\lambda_0)$ is tridiagonal.[11]

For $\mathbf{C}_2(\lambda_0)$, we can verify that the products of the neighboring off-diagonals are positive,

$$\mathbf{C}_2(\lambda_0)_{j,j+1}\mathbf{C}_2(\lambda_0)_{j+1,j} > 0, \tag{A.2}$$

for all j (when $\lambda_0 = 0$ the product can be 0; we exclude this case for the moment and assume $\lambda_0 < 0$). Hence, $\mathbf{C}_2(\lambda_0)$ is diagonally similar to a symmetric matrix, thus its eigenvalues are all real.

It remains to prove that the eigenvalues of $\mathbf{C}_1(\lambda_0)$ are all real. Note that $\mathbf{C}_1(\lambda_0)$ is of the form

$$\mathbf{C}_1(\lambda_0) = \begin{pmatrix} \alpha & \beta' & & & & & & \beta' \\ \beta & \alpha_1 & \beta_1 & & & & & \\ & \beta_{\ell-2} & \alpha_2 & \beta_2 & & & & \\ & & \beta_{\ell-3} & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \beta_{\ell-1} & \\ & & & & & \beta_2 & \alpha_2 & \beta_{\ell-2} \\ \beta & & & & & & \beta_1 & \alpha_1 \end{pmatrix}. \tag{A.3}$$

Several properties of $\mathbf{C}_1(\lambda_0)$ are worth noting: (i) the $(\ell - 1) \times (\ell - 1)$ submatrix obtained by removing the first row and column is symmetric about the antidiagonal, both in the diagonal and off-diagonal elements (note the double appearance of β_i) and (ii) the products of the neighboring off-diagonal blocks are all positive $\beta_j\beta_{\ell-j-1} > 0$ for all j, and $\beta\beta' > 0$. In Lemma A.2 below we prove that any matrix (A.3) with such structure has only real eigenvalues, establishing that $f_j(\lambda_0)$ is real for any $\lambda_0 < 0$. The claim extends to $\lambda_0 = 0$ by continuity of the eigenvalues, completing the second step in the proof.

It remains to show $f_j(0) \le 0$ for every j, that is, $\mathbf{A}$ has only nonpositive eigenvalues. Since $\mathbf{D}_n^{(2)}(i, i) \le 0$ for all i, it suffices to treat the case for which $\mathbf{D}_n^{(2)}(i, i) = 0$. We again examine

---

[11]It is possible to use this structure in the linear solvers. We did not do this in our experiments, as applying the permutation also requires $\mathcal{O}(nm)$ operations.

$\mathbf{C}_1(0)$ and $\mathbf{C}_2(0)$ separately. For each of these, after deflating the zero eigenvalue (if present) we can apply a diagonal similarity transformation so that the Gershgorin disks, whose centers lie on the negative half line, do not contain the origin, implying that all the eigenvalues are nonpositive. This completes the proof of Theorem A.1. $\square$

It remains to prove that the eigenvalues of matrices of the form in (A.3) are all real. A key fact is that a real tridiagonal matrix with the neighboring off-diagonals having the same sign is diagonally similar to a symmetric tridiagonal matrix, and an analogous result holds for arrowhead matrices.

LEMMA A.2. *For any real matrix of the form* (A.3), *with* $\beta_i \beta_{\ell-i-1} > 0$ *for all* i *and* $\beta\beta' > 0$, *all the eigenvalues are real.*

Proof. Denote the matrix by $\mathbf{C}$. We can apply a diagonal similarity transformation to the bottom-right $(\ell-1) \times (\ell-1)$ part $\mathbf{C}_2$, to obtain a symmetric matrix $\mathbf{D}^{-1}\mathbf{C}_2\mathbf{D}$. Since $\mathbf{C}_2$ is symmetric about the antidiagonal, so is the diagonal matrix $\mathbf{D}$; thus $\mathbf{D}^{-1}\mathbf{C}_2\mathbf{D}$ is both symmetric and symmetric about the antidiagonal, and so the transformation $\widehat{\mathbf{C}} = \begin{bmatrix} 1 & \\ & \mathbf{D}^{-1} \end{bmatrix} \mathbf{C} \begin{bmatrix} 1 & \\ & \mathbf{D} \end{bmatrix}$ preserves the property that the off-diagonal parts of the first row and column are parallel (when one is transposed). Now let $\mathbf{Q}$ be an orthogonal matrix of eigenvectors of $\mathbf{D}^{-1}\mathbf{C}_2\mathbf{D}$ such that $\mathbf{Q}^T(\mathbf{D}^{-1}\mathbf{C}_2\mathbf{D})\mathbf{Q}$ is diagonal, and consider the matrix $\widetilde{\mathbf{C}} = \begin{bmatrix} 1 & \\ & \mathbf{Q}^T \end{bmatrix} \widehat{\mathbf{C}} \begin{bmatrix} 1 & \\ & \mathbf{Q} \end{bmatrix}$. By the antidiagonal symmetry of $\mathbf{D}^{-1}\mathbf{C}_2\mathbf{D}$, each eigenvector (column of $\mathbf{Q}$) has the property that it is either in the form $[v_1, v_2, \ldots, -v_2, -v_1]^T$ or $[v_1, v_2, \ldots, v_2, v_1]^T$. Therefore, $\widetilde{\mathbf{C}}$ is an arrowhead matrix with the property that for every j, we have either $\widetilde{\mathbf{C}}_{j,1} = \widetilde{\mathbf{C}}_{1,j} = 0$, or $\widetilde{\mathbf{C}}_{j,1}\widetilde{\mathbf{C}}_{1,j} > 0$. It follows that there exists a diagonal similarity transformation that brings $\widetilde{\mathbf{C}}$ to symmetric form, hence has real eigenvalues. $\square$

REFERENCES

[1] J. C. ADAMS AND P. N. SWARZTRAUBER, *SPHEREPACK* 3.0: *a model development facility*, Mon. Wea. Rev., 127 (1999), pp. 1872–1878.

[2] S. M. ALLEN AND J. W. CAHN, *A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening*, Acta Metall., 27 (1979), pp. 1085–1095.

[3] U. M. ASCHER, S. J. RUUTH, AND B. T. R. WETTON, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823.

[4] H. A. ASHI, L. J. CUMMINGS, AND P. C. MATTHEWS, *Comparison of methods for evaluating functions of a matrix exponential*, Appl. Numer. Math., 59 (2009), pp. 468–486.

[5] K. ATKINSON AND W. HAN, *Spherical Harmonics and Approximations on the Unit Sphere: An Introduction*, Springer, Berlin Heidelberg, 2012.

[6] J. AURENTZ, *Fast algorithms for spectral differentiation matrices*, Electron. Trans. Numer. Anal., 44 (2015), pp. 281–288.

[7] S. BHATTACHARYA, *Galerkin model for Turing patterns on a sphere*, Phys. Rev. E, 72 (2005), p. 036208.

[8] J. P. BOYD, *The choice of spectral functions on a sphere for boundary and eigenvalue problems: A comparison of Chebyshev, Fourier and associated Legendre expansions*, Mon. Wea. Rev., 106 (1978), pp. 1184–1191.

[9] M. P. CALVO, J. DE FRUTOS, AND J. NOVO, *Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations*, Appl. Numer. Math., 37 (2001), pp. 535–549.

[10] A. CARDONE, Z. JACKIEWICZ, A. SANDU, AND H. ZHANG, *Extrapolation-based implicit-explicit general linear methods*, Numer. Algor., 65 (2014), pp. 377–399.

[11] H.-B. CHEONG, *Double Fourier series on a sphere: Applications to elliptic and vorticity equations*, J. Comput.

```matlab
% Parameters:
  m = 1024; n = m;                              % number of grid points
  h = 1e-1; T = 100;                            % time-step and final time
  u0 = @(x,y,z) cos(40*x)+cos(40*y)+cos(40*z);
  th = pi/8; c = cos(th); s = sin(th);
  u0 = 1/3*spherefun(@(x,y,z) u0(c*x-s*z,y,s*x+c*z)); % initial condition
  v0 = reshape(coeffs2(u0, m, n), m*n, 1);      % Fourier coefficients

% Nonlinear operator (evaluated in value space):
  g = @(u) u - (1+1.5i)*u.*(abs(u).^2);         % N(u) = u-(1+1.5)i*u*|u|^2
  c2v = @(u) trigtech.coeffs2vals(u);           % coeffs to values in 1D
  c2v = @(u) c2v(c2v(reshape(u,m,n)).').';      % coeffs to values in 2D
  v2c = @(u) trigtech.vals2coeffs(u);           % values to coeffs in 1D
  v2c = @(u) reshape(v2c(v2c(u).').',m*n,1);    % values to coeffs in 2D
  N = @(u) v2c(g(c2v(u)));                       % nonlinear operator

% Construct the Laplacian matrix (multiplied by Tsin2 and 1e-4):
  Dm = spdiags(1i*[0,-m/2+1:m/2-1]', 0, m, m);
  D2m = spdiags(-(-m/2:m/2-1).^2', 0, m, m);
  D2n = spdiags(-(-n/2:n/2-1).^2', 0, n, n);
  Im = speye(m); In = speye(n);
  P = speye(m+1); P = P(:, 1:m); P(1,1) = .5; P(m+1,1) = .5;
  Q = speye(m+1+4); Q = Q(3:m+2,:); Q(1,3) = 1; Q(1,m+3) = 1;
  Msin2 = toeplitz([1/2, 0, -1/4, zeros(1, m+2)]);
  Msin2 = sparse(Msin2(:, 3:m+3));
  Tsin2 = round(Q*Msin2*P, 15);
  Mcossin = toeplitz([0, 0, 1i/4, zeros(1, m+2)]);
  Mcossin = sparse(Mcossin(:, 3:m+3));
  Tcossin = round(Q*Mcossin*P, 15);
  Lap = 1e-4*(kron(In, Tsin2*D2m + Tcossin*Dm) + kron(D2n, Im));

% Compute LU factorizations of LIRK4 matrices:
  Tsin2 = kron(In, Tsin2);
  [L, U] = lu(Tsin2); [La, Ua] = lu(Tsin2 - 1/4*h*Lap);

% Time-stepping loop:
  itermax = round(T/h); v = v0;
  for iter = 1:itermax
      Nv = N(v); w = Tsin2*v;
      wa = w + h*Tsin2*1/4*Nv;
      a = Ua\(La\wa); Na = N(a);
      wb = w + h*Lap*1/2*a + h*Tsin2*(-1/4*Nv + Na);
      b = Ua\(La\wb); Nb = N(b);
      wc = w + h*Lap*(17/50*a - 1/25*b) + h*Tsin2*(-13/100*Nv + 43/75*Na + 8/75*Nb);
      c = Ua\(La\wc); Nc = N(c);
      wd = w + h*Lap*(371/1360*a - 137/2720*b + 15/544*c) ...
          + h*Tsin2*(-6/85*Nv + 42/85*Na + 179/1360*Nb - 15/272*Nc);
      d = Ua\(La\wd); Nd = N(d);
      we = w + h*Lap*(25/24*a - 49/48*b + 125/16*c - 85/12*d) ...
          + h*Tsin2*(79/24*Na - 5/8*Nb + 25/2*Nc - 85/6*Nd);
      e = Ua\(La\we); Ne = N(e);
      v = v + h*(U\(L\(Lap*(25/24*a - 49/48*b + 125/16*c - 85/12*d + 1/4*e)))) ...
          + h*(25/24*Na - 49/48*Nb + 125/16*Nc - 85/12*Nd + 1/4*Ne);
  end
  vals = c2v(v);                                % tramsform to value space
  vals = vals([m/2+1:m 1], :);                  % restrict to [-pi,pi]x[0,pi]
  u = spherefun(real(vals)); plot(u)            % output real(u) and plot
```

Fig. A.1: MATLAB *code to solve the Ginzburg–Landau equation on the sphere with the DFS method and the LIRK4 time-stepping scheme; this code can be used for both diffusive and dispersive PDEs.*

Phys., 157 (2000), pp. 327–349.

[12] E. M. Constantinescu and A. Sandu, *Extrapolated implicit-explicit time stepping*, SIAM J. Sci. Comput., 31 (2010), pp. 4452–4477.

[13] S. M. Cox and P. C. Matthews, *Exponential time differencing for stiff systems*, J. Comput. Phys., 176 (2002), pp. 430–455.

[14] T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2006.

[15] T. A. Driscoll, *A composite Runge-Kutta method for the spectral solution of semilinear PDEs*, J. Comput. Phys., 182 (2002), pp. 357–367.

[16] T. A. Driscoll, N. Hale, and L. N. Trefethen, eds., *Chebfun Guide*, Pafnuty Publications, Oxford, 2014; see also www.chebfun.org.

[17] Q. Du and W. Zhu, *Analysis and applications of the exponential time differencing schmes and their contour integral modifications*, BIT, 45 (2005), pp. 307–328.

[18] Y. Du, *The heterogeneous Allen–Cahn equation in a ball: Solutions with layers and spikes*, J. Differential Equations, 244 (2008), pp. 117–169.

[19] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, 1998.

[20] B. Fornberg and T. A. Driscoll, *A fast spectral algorithm for nonlinear wave equations with linear dispersion*, J. Comput. Phys., 155 (1999), pp. 456–467.

[21] B. Fornberg and N. Flyer, *A Primer on Radial Basis Functions with Applications to the Geosciences*, SIAM, Philadelphia, 2015.

[22] B. Fornberg and D. Merrill, *Comparison of finite difference- and pseudospectral methods for convective flow over a sphere*, Geophys. Res. Lett., 24 (1997), pp. 3245–3248.

[23] D. Gottlieb and L. Lustman, *The spectrum of the Chebyshev collocation operator for the heat equation*, SIAM J. Numer. Anal., 20 (1983), pp. 909–921.

[24] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, New York, 1991.

[25] P. Henrici, *Applied and Computational Complex Analysis*, vol. 3, Wiley, New York, 1986.

[26] M. Hochbruck and A. Ostermann, *Explicit exponential Runge-Kutta methods for semilinear parabolic problems*, SIAM J. Numer. Anal., 43 (2005), pp. 1069–1090.

[27] ———, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.

[28] W. Hundsdorfer and S. J. Ruuth, *IMEX extensions of linear multistep methods with general monotonicity and boundedness properties*, J. Comput. Phys., 225 (2007), pp. 2016–2042.

[29] A.-K. Kassam and L. N. Trefethen, *Fourth-order time-stepping for stiff PDEs*, SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.

[30] C. A. Kennedy and M. H. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math., 44 (2003), pp. 139–181.

[31] S. Krogstad, *Generalized integrating factor methods for stiff PDEs*, J. Comput. Phys., 203 (2005), pp. 72–88.

[32] V. T. Luan and A. Ostermann, *Explicit exponential Runge-Kutta methods of high order for parabolic problems*, J. Comput. Appl. Math., 256 (2014), pp. 168–179.

[33] P. E. Merilees, *The pseudospectral approximation applied to the shallow water equations on a sphere*, Atmosphere, 11 (1973), pp. 13–20.

[34] B. V. Minchev, *Exponential integrators for semilinear problems*, PhD thesis, University of Bergen, 2004.

[35] C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.

[36] H. Montanelli and N. Bootland, *Solving periodic semilinear stiff PDEs in 1D, 2D and 3D with exponential integrators*, submitted (2016).

[37] S. A. Orszag, *Fourier series on spheres*, Mon. Wea. Rev., 102 (1974), pp. 56–75.

[38] A. Ostermann, M. Thalhammer, and W. M. Wright, *A class of explicit exponential general linear methods*, BIT, 46 (2006), pp. 409–431.

[39] L. Pareschi and G. Russo, *Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation*, J. Sci. Comput., 25 (2005), pp. 129–155.

[40] L. Pismen and J. Rubinstein, *Dynamics of disclinations in liquid crystals*, Quart. Appl. Math., 50 (1992), pp. 535–545.

[41] Y. Pomeau, S. Zaleski, and P. Manneville, *Dislocation motion in cellular structures*, Phys. Rev. A, 27 (1983), pp. 2710–2726.

[42] M. E. Rognes, D. A. Ham, C. J. Cotter, and A. T. T. McRae, *Automating the solution of PDEs on the sphere and other manifolds in FEniCS 1.2*, Geosci. Model Dev., 6 (2013), pp. 2099–2119.

[43] V. Rokhlin and M. Tygert, *Fast algorithms for spherical harmonic expansions*, SIAM J. Sci. Comput., 27 (2006), pp. 1903–1928.

[44] J. Rubinstein and P. Sternberg, *On the slow motion of vortices in the Ginzburg–Landau heat flow*, SIAM J. Math. Anal., 26 (1995), pp. 1452–1466.

[45] T. Schmelzer and L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals*, Electron. Trans. Numer. Anal., 29 (2007), pp. 1–18.

[46] J. Shen, *Efficient spectral-Galerkin methods IV. Spherical geometries*, SIAM J. Sci. Comput., 20 (1999), pp. 1438–1455.

[47] R. M. Slevinsky, *Fast and backward stable transforms between spherical harmonic expansions and bivariate Fourier series*, submitted (2017).

[48] K. Stewartson and J. T. Stuart, *A non-linear instability theory for a wave system in plane Poiseuille flow*, J. Fluid Mech., 48 (1971), pp. 529–545.

[49] H. Takaoka, *Local well-posedness of the nonlinear Schrödinger equations on the sphere for data in modulation spaces*, Commun. Part. Diff. Eq., 41 (2016), pp. 732–747.

[50] M. Tokman, *Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods*, J. Comput. Phys., 213 (2006), pp. 748–776.

[51] A. Townsend, H. Wilber, and G. B. Wright, *Computing with functions in spherical and polar geometries, I. The sphere*, SIAM J. Sci. Comput., 38 (2016), pp. C403–C425.

[52] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.

[53] L. N. Trefethen and M. H. Gutknecht, *The Carathéodory–Fejér method for real rational approximation*, SIAM J. Numer. Anal., 20 (1983), pp. 420–436.

[54] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, *Talbot quadratures and rational approximations*, BIT, 46 (2006), pp. 653–670.

[55] P. H. Trinh and M. J. Ward, *The dynamics of localized spot patterns for reaction-diffusion systems on the sphere*, Nonlinearity, 29 (2016), pp. 766–806.

[56] M. Tygert, *Fast algorithms for spherical harmonic expansions, II*, J. Comput. Phys., 227 (2008), pp. 4260–4279.

[57] ———, *Fast algorithms for spherical harmonic expansions, III*, J. Comput. Phys., 229 (2010), pp. 6181–6192.

[58] R. S. Varga, *Geršgorin and His Circles*, Springer, Berlin, 2004.

[59] G. B. Wright, M. Javed, H. Montanelli, and L. N. Trefethen, *Extension of Chebfun to periodic functions*, SIAM J. Sci. Comput., 37 (2015), pp. C554–C573.

[60] S. Y. K. Yee, *Solution of Poisson's equation on a sphere by truncated double Fourier series*, Mon. Wea. Rev., 109 (1980), pp. 501–505.