International Journal of High Performance Computing Applications
XX(X):1–18
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems

Martin Schreiber[1], Pedro S. Peixoto[1,2], Terry Haut[3] and Beth Wingate[1]

Abstract
This paper presents, discusses and analyses a massively parallel-in-time solver for linear oscillatory PDEs, which is a key numerical component for evolving weather, ocean, climate and seismic models. The time parallelization in this solver allows us to significantly exceed the computing resources used by parallelization-in-space methods and results in a correspondingly significantly reduced wall-clock time. One of the major difficulties of achieving Exascale performance for weather prediction is that the strong scaling limit – the parallel performance for a fixed problem size with an increasing number of processors – saturates. A main avenue to circumvent this problem is to introduce new numerical techniques that take advantage of time parallelism. In this paper we use a time-parallel approximation that retains the frequency information of oscillatory problems. This approximation is based on (a) reformulating the original problem into a large set of independent terms and (b) solving each of these terms independently of each other which can now be accomplished on a large number of HPC resources. Our results are conducted on up to 3586 cores for problem sizes with the parallelization-in-space scalability limited already on a single node. We gain significant reductions in the time-to-solution of 118.3 for spectral methods and 1503.0 for finite-difference methods with the parallelization-in-time approach. A developed and calibrated performance model gives the scalability limitations a-priory for this new approach and allows us to extrapolate the performance method towards large-scale system. This work has the potential to contribute as a basic building block of parallelization-in-time approaches, with possible major implications in applied areas modelling oscillatory dominated problems.

Keywords
Parallelization in time, oscillatory problem, exponential integrator, rational approximation, scalability limitation

1 Introduction

In this paper, we investigate a massively parallel matrix exponential algorithm for simulating stiff oscillatory PDEs, which arises in applications such as magnetohydrodynamics, plasma physics, weather and climate.

With the current trend of HPC towards massive parallelism, existing solvers are facing the strong scalability limitation. Here, a key performance limitation is the saturation of scalability in space and a sequential step-by-step approach in time. Achieving additional parallel speedup on future architectures requires novel mathematical approaches to overcome this scalability limitation. Therefore, our strategy for this paper is to focus on the aspect of developing algorithms to suit the technology's strengths and limitations, rather than modifying the architecture to fit the algorithm.

In this work, we first discuss HPC and mathematical issues before describing the details of how and when a reduction in wall-clock time is possible. We are primarily concerned with the solution of equations that are stiff due to a separation of time scales in the mathematical description of the physics. This creates fast oscillations in the system of equations that, when numerical accuracy is required, bind the magnitude of the time step to the spatial resolution which is traditionally solved sequentially in time. Instead of using sequential-in-time solvers, we explore the viability of time-parallelism by investigating the rational approximation of exponential integrators (REXI).

1.1 Challenges in HPC

The trend for performance increase on all silicon-based hardware components changed one decade ago. Due to a breakdown of Dennard's scaling (Dennard et al. 1974) in about 2005, additional performance by increasing the frequency for singe-core CPUs was not possible any more. To achieve further scalability according to Moore's law[*] (Moore 2006), the only way to gain more performance was an increase in on-chip parallelism: This resulted in a change from single-cores to multi-core processors which are nowadays omnipresent even on mobile devices and an increase in data processable in one instruction (SIMD).

The reduction of the time-to-solution is therefore limited by the parallelizable part which itself is limited by the scalability in space with standard time stepping schemes. This is the case since the parallelizable part is reduced

[1]College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK
[2]Instituto de Matemática e Estatística, Universidade de São Paulo, Brazil
[3] Computer, Computational and Statistical Sciences Division, Los Alamos National Laboratory, USA

Corresponding author:
Martin Schreiber, College of Engineering, Mathematics and Physical Sciences - University of Exeter, Exeter, UK.
Email: M.Schreiber@exeter.ac.uk

[*]Here, we interpret Moore's law via performance scaling

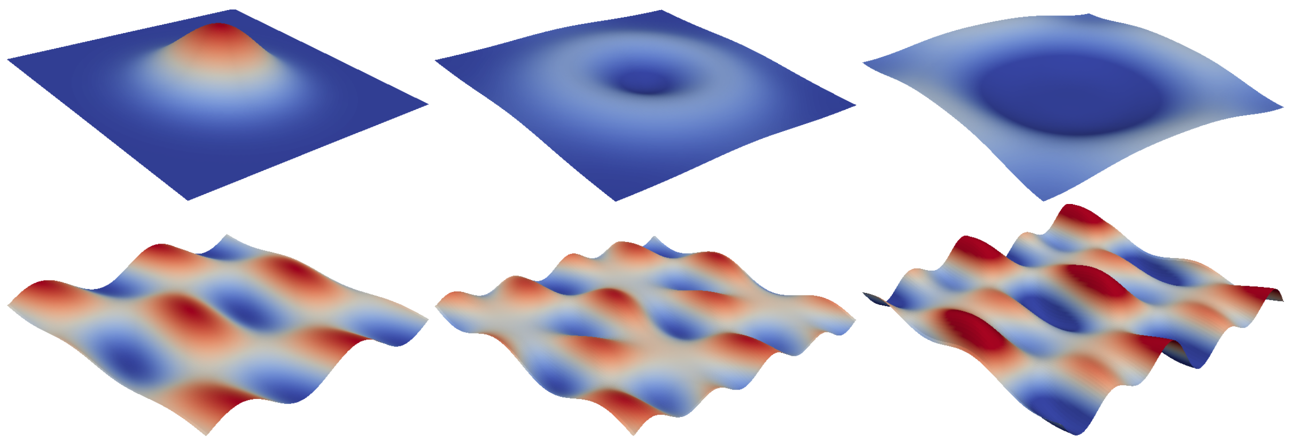


Figure 1. Examples of shallow-water simulations. Top row: Gaussian initial condition (top row) at time $t = \{0.0, 0.2, 0.4\}$. Bottom row: Wave-like initial conditions (bottom row) at time $t = \{0.0, 0.1, 0.2\}$. We solve these equations with a parallel-in-time approach which avoids time step restrictions.

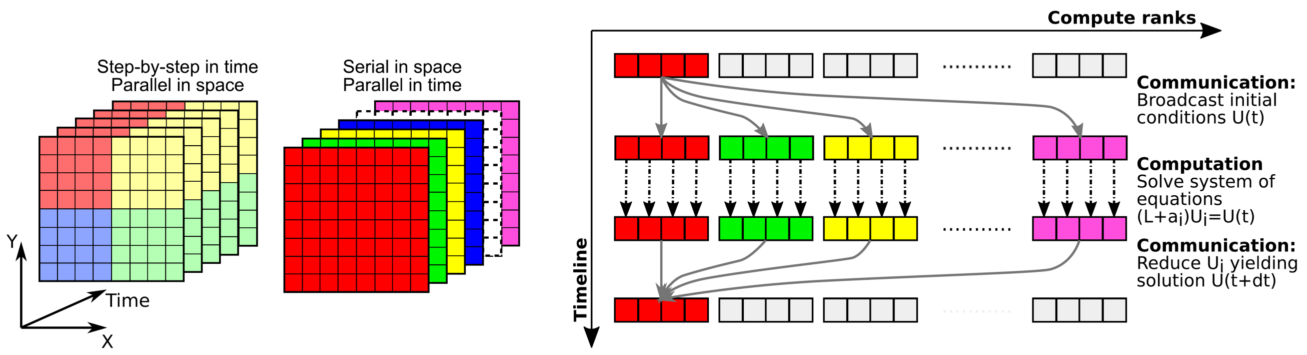

Step-by-step in time
Parallel in space
Y
Time
X
Serial in space
Parallel in time
Compute ranks
Timeline
Communication: Broadcast initial conditions U(t)
Computation Solve system of equations (L+a$_i$)U$_i$=U(t)
Communication: Reduce U$_i$ yielding solution U(t+dt)

Figure 2. Left: Parallelization-in-space only, each color denotes one compute unit and each slice in time typically requires additional synchronization for the time stepping.
Center: Parallelization-in-time only, each time slice is assigned to a differently colored group of compute unit. Overheads in space are non-existent, but a final synchronization in time is required.
Right: Schematic view on the realization of the REXI parallelization. The first group of compute units broadcasts the initial conditions to all other compute ranks. Second, each group solves the corresponding REXI terms. Third, all solutions are reduced to the compute units on the first rank.

and at a certain number of processors, the serial part and communication dominate the computation time. A potential way to tackle these issues is work in an interdisciplinary way among mathematics and HPC science.

This is also the case for the types of simulations of interest in this paper, like climate and weather prediction, magnetohydrodynamics, and fusion processes, where the dominant method of achieving parallel speed-ups has been to use parallelization-in-space by using domain partitioning techniques. This technique is prone to fail for strong-scalability on future Exascale architectures described above due to standard time stepping solvers and their inherent step-by-step application.

1.2 Parallelization-in-time and Exponential Integrators

Once the strong-scaling limitation of domain decomposition methods is reached, new parallelism paradigms need to be explored. One important avenue is the introduction of parallelism in the time dimension. Although this is not a new idea, it has recently gained a lot of attention (see Gander (2015) for a review). There are several different ways to explore the time dimension parallelism, with some successes in applied problems (e.g. Samaddar et al. (2010); Berry et al. (2012); Speck et al. (2012)).

For highly oscillatory nonlinear problems, such as the ones arising in climate and weather models, there is still a lot to be done to show that parallel-in-time methods can be effective. Haut and Wingate (2014) show a way to obtain an efficient parallel-in-time method for a simplified non-linear oscillatory problem, but this relies on having a time integrator that can permit very long time steps without damping the linear oscillations. Usual implicit time integrators, although allow long time steps, generally damp linear waves, so the time integration techniques needs to be rethought for parallel-in-time strategies. In this direction, the use of exponential integrators seems attractive.

Exponential integrators deals with the time evolution in its exponential form, usually making use of its semi-group properties (Hochbruck and Ostermann 2010; Cox and Matthews 2002). Numerically, the problem results in having to calculate exponential of matrices, which has been studied for many years and there are numerous solution strategies (see Moler and Van Loan (2003) for a review). The work of Güttel (2013) suggests using rational Krylov spaces to approximate the exponential of skew-Hermitian matrices, which arise in hyperbolic problems. However, adding a new Krylov subspace requires an additional matrix-vector multiplication for each additional subspace, which is inherently serial. Padé approximations (see Higham (2008); Moler and Van Loan (2003)) require computing power series with matrix-matrix multiplications, hence also have this inherently serial part, which gets critical for large matrices. Another method to reduce the $O(n^3)$ complexity in case of matrix-matrix multiplications for exponential integrators was presented in Bonaventura (2015) and is based on splitting the domain into sub-problems. Exponential integrators have also been used in the weather and climate community, with some success (Clancy and Pudykiewicz 2013; Garcia et al. 2014). Nevertheless, these methods are usually intrinsically serial, and we would like to explore methods that allow an extra degree of time-parallelism embedded in the exponential integrator, as a way to reduce time-to-solution.

Along the lines of building a parallel-in-time exponential integrators, Gander and Guettel (2013) propose a parallel-in-time method for general linear initial-value problems which makes use of polynomial and rational Krylov methods for the exponentiation part. In the present paper, we follow the ideas of Haut et al. (2015), which proposes a rational approximation of the exponential integrator, which we will denote as REXI. The key point is that this rational approximation is massively parallel. The downside is that it is less general, as it is built only for hyperbolic (oscillatory) linear initial-value problems, which fortunately is the class of problems we are interested in this work. As a direct comparison, for Krylov methods, reducing the approximation error requires computing more Krylov subspaces and hence leads to an increase of the serial parts. In contrast, as we will show, the REXI approach leads to an increase in parallelism, which is the main reason we focus on this approach in this work.

1.3 Overview

For sake of clarity, in Section 2 we review in detail the implementation of the rational function approach to compute the matrix exponential integration (REXI). We evaluate REXI with various different execution scenarios, different discretization schemes and show its competitiveness. This is done using various benchmark scenarios (described in Section 3) which are representative for basic building blocks of climate and weather simulations. We present our first results on the numerical accuracy of the method in Section 4. These results are important to understand the role of the different parameter regimes on the speed and accuracy of the method. For example, one of these parameters represents the new degree of parallelization which we can exploit with REXI to reduce the time-to-solution. The second type of results, presented in Section 5, is an evaluation of REXI using different parallelization aspects (space and/or time). Finally, we gain insight on the behavior of REXI on large-scale systems using a performance model derived in Section 6.

For sake of reproducibility the source code is published as well (Schreiber et al. 2016).

2 Review of the Rational Exponential Integrator (REXI)

This section is a comprehensive discussion of the rational function exponential integrator (REXI) for linear systems of equations. It provides a fundamental understanding of how the mathematics translates into time-parallel speed-ups and therefore illustrates the method's strengths and limitations. Even more importantly, the details published in this section are required for any other investigators to repeat and improve on our calculations.

We first review how time parallelism is introduced into what is often an inherently serial problem before we outline the details of the REXI approach (Haut et al. 2015).

2.1 Time-parallelism with REXI in a simplified ODE problem

Consider the ordinary differential equation (ODE),

$$\frac{du}{dt} = \lambda u, \qquad u(t_0) = u_0, \qquad \lambda \in \mathbb{C} \tag{1}$$

with exact solution

$$u(t) = u_0 e^{\lambda(t-t_0)}. \tag{2}$$

One standard way to approximate (1) numerically is, for example, using first order forward Euler,

$$\tilde{u}(t + \Delta t) = (1 + \lambda \Delta t)\, \tilde{u}(t), \tag{3}$$

where $\tilde{u}$ denotes a numerical approximation to $u(t)$.

Assuming integration up until time T, with n timesteps ($\Delta t = (T - t_0)/n$), we have that the Euler scheme approximates the exact solution with sequential applications of Eq. (3) resulting in

$$\tilde{u}(T) = \underbrace{(1 + \lambda \Delta t)^n}_{\approx e^{\lambda(T-t_0)}} u_0, \tag{4}$$

where the exponential approximation under the braces is adequate for large values of n, or equivalently, Δt sufficiently small.

This kind of step-by-step procedure is inherently serial and may have stability constraints (particularly for explicit schemes, such as the one described above). For oscillatory problems (where λ is purely imaginary) we may replace the inherently serial operation of approximating $e^{\lambda(T-t_0)}$ by a sum of independent rational functions. This transfers the serial nature of the problem to one that has more parallelism. This is accomplished using an approximation of the form

$$e^{\lambda t} \approx \sum_{k=-N}^{N} \frac{\beta_k}{\lambda t + \alpha_k} \tag{5}$$

where α_k, β_k are precomputed complex numbers and $2N + 1$ defines the number of terms in the summation to approximate the exponential, which will be discussed in details in the next subsections.

Each of the terms in the sum can be evaluated independently, therefore, can be distributed on different processors. One time step of this scheme (of size $T - t_0$) would be analogous to n time steps of size Δt of the forward

Euler scheme, but these terms in the sum may be calculated in parallel. We will show that the larger the time step (larger T), the larger the number of terms in the sum needs to be (larger N), but this can be absorbed in a parallel way. This represents a fundamental transformation from an inherently sequential step-by-step in time process to one that can be parallelized. Rather than taking many small time steps to find the solution at a certain time, one can take fewer, larger, parallel steps to reach the same time.

One issue is that each large time step is potentially more expensive, but highly parallel. This is advantageous if very large time steps are possible or if there is a gain in accuracy that can be absorbed in a parallel way. The trade-off between the extra expense of computing the rational function and its high degree of parallelization is one of the main topics we address in this paper.

2.2 REXI with hyperbolic systems

We now extend this approach to linear hyperbolic PDE problems of the form

$$U_t = \mathcal{L}U, \quad U(t_0) = U_0 \tag{6}$$

with $\mathcal{L}$ being the operator in space, the subscript t referring to the time derivative $\frac{\partial}{\partial t}$, and U being a vector of unknowns.

The solution may be written as

$$U(t) = e^{(t-t_0)\mathcal{L}}U(t_0), \tag{7}$$

where $e^{(t-t_0)\mathcal{L}}$ is the semi-group or exponential integrator of the operator $\mathcal{L}$. The goal of this section is to describe how to compute an approximation of $U(t)$ of the form

$$e^{(t-t_0)\mathcal{L}}U(t_0) \approx \sum_{n=0}^{N} \gamma_n((t-t_0)\mathcal{L} + \alpha_n I)^{-1}U(t_0), \tag{8}$$

where γ_n and α_n are parameters to be specified and I is the identity matrix. This will be done using the Rational Approximation of an Exponential Integrator (Haut et al. 2015) and we will review the method carefully so that other investigators will be able to understand and reproduce our results.

Since it is the oscillatory nature of the problem that we're investigating, we assume the problem is hyperbolic and thus that the linear operator has only purely imaginary eigenvalues. We begin by describing the approximation of e^{ix} (see Sec. 2.3), whose rational approximation for its real part (Re) is,

$$Re\left(e^{ix}\right) \approx \sum_{n=-N}^{N} Re\left(\frac{\beta_n}{ix+\alpha_n}\right), \tag{9}$$

with complex coefficients α_n and β_n to be defined, and range of the sum $n \in \{-N,\dots,N\}$. Then we will extend these concepts to the general multi-dimensional case $e^{\tau\mathcal{L}}$ (see Sec. 2.4) followed by the usage of symmetry properties which will halve the range of the sum to $n \in \{0,\dots,N\}$ (see Sec. 2.5).

2.3 Rational approximation of $\exp(ix)$

The approximation of the e^{ix} used in this paper relies on the combination of two approximation steps:

(A) Approximation of e^{ix} by a linear combination of Gaussian kernels.

(B) Approximation of Gaussian kernels by a linear combination of rational functions.

Step A: *Approximation of* $\exp(ix)$ *by Gaussian kernels*

Let

$$\psi_h(x) = (4\pi)^{-\frac{1}{2}}e^{-x^2/(4h^2)}, \tag{10}$$

be our Gaussian function to be used as basis for our approximation, where h is a parameter that specifies the width of the main support region of the basis function (a horizontal "stretching" measure, similar to the standard deviation of a non-normalized Gaussian distribution).

Given an arbitrary complex function $f(x)$, which will be later specialized to $\exp(ix)$, we use superimposition of translations of the basis functions $\psi_h(x)$ to build the following approximation,

$$f(x) \approx \sum_{m=-M}^{M} b_m\psi_h(x+mh), \tag{11}$$

where M controls the interval of approximation (relative to width of "domain of interest") and h defines the sampling rate (relative to the resolution of "domain of interest"). The accuracy of the approximation is highly dependent on proper choices of both h and M. Since the translated points lay between $x - Mh$ and $x + Mh$, this approximation has been shown to be adequate for $|x| \lesssim Mh$ (Haut et al. 2015).

To compute the coefficients b_m, we rewrite the previous equation in Fourier space with

$$\frac{\hat{f}(\xi)}{\hat{\psi_h}(\xi)} = \sum_{m=-\infty}^{\infty} b_m e^{2\pi imh\xi}, \tag{12}$$

where the $\hat{\cdot}$ symbols indicate the Fourier transforms of the respective functions. The b_m are now the Fourier coefficients of the series for the function $\frac{\hat{f}(\xi)}{\hat{\psi}_h(\xi)}$ and can be calculated as,

$$b_m = h\int_{-\frac{1}{2h}}^{\frac{1}{2h}} e^{-2\pi imh\xi}\frac{\hat{f}(\xi)}{\hat{\psi_h}(\xi)}d\xi, \tag{13}$$

for m integer and $1/h$ defining the periodicity of the trigonometric basis function. The parameter h is then chosen to be small enough (see Section 4.1) so that the support of the Fourier transform of f is mainly localised within $[-1/(2h), 1/(2h)]$, i.e. almost zero outside this interval.

Since we are interested in approximating $f(x) = e^{ix}$, we can simplify the above equation by using the response in frequency space $\hat{f}(\xi) = \delta(\xi - \frac{1}{2\pi})$, where here δ is the Dirac distribution, which leads to

$$b_m = h\, e^{-imh}\hat{\psi}_h\left(\frac{1}{2\pi}\right)^{-1}. \tag{14}$$

Using the Fourier transform of the Gaussian function we can finally build a clear expression for the coefficients b_m for $f(x) = e^{ix}$,

$$b_m = h\, e^{-imh}\frac{1}{h\, e^{-h^2}} = e^{-imh}e^{h^2}. \tag{15}$$

As shown in Haut et al. (2015), the approximation (11) is very accurate for $|x| \lesssim Mh$. The x variable in the $\exp(ix)$ function will play the role of representing different wavenumbers of the linear operator $\mathcal{L}$, therefore we expect that M and h can be appropriately chosen to allow an adequate representation of the relevant wavenumbers of the problem.

Step B: *Rational approximation of the Gaussian basis functions* The second step is the approximation of the basis function $\psi_h(x)$ itself with a rational approximation. A close-to-optimal rational approximation of $\psi_h(x)$ is given by

$$\psi_h(x) \approx \text{Re}\left(\sum_{l=-L}^{L} \frac{a_l}{i\frac{x}{h} + (\mu + i\,l)} \right), \tag{16}$$

where μ and a_l are constant given in Haut et al. (2015), Table 1. The number of coefficients $(2L+1)$ depends only on the desired accuracy for this approximation. With $L = 11$ an accuracy greater than single precision is already obtained.

Final step: *Rational approximation of* $\exp(ix)$
Combining the approximation (B) into the approximation (A), we have that

$$\begin{aligned} e^{ix} &\approx \sum_{m=-M}^{M} b_m \psi_h(x + mh) \\ &= \sum_{m=-M}^{M} b_m \sum_{l=-L}^{L} \text{Re}\left(\frac{h a_l}{ix + h(\mu + i(m+l))} \right). \end{aligned}$$

Now, defining $n = m + l$, $\alpha_n = h(\mu + in)$, and

$$\beta_n^{\text{Re}} = h \sum_{m=-M}^{M} \sum_{l=-L}^{L} \text{Re}(b_m) a_l \delta_{n,\,m+l}, \tag{17}$$

and

$$\beta_n^{\text{Im}} = h \sum_{m=-M}^{M} \sum_{l=-L}^{L} \text{Im}(b_m) a_l \delta_{n,\,m+l}, \tag{18}$$

where $\delta_{i,j} = 1$ if $i = j$ and zero otherwise, we finally obtain the rational approximation for $\exp(ix)$ as

$$e^{ix} \approx \sum_{n=-N}^{N} \text{Re}\left(\frac{\beta_n^{\text{Re}}}{ix + \alpha_n} \right) + i\,\text{Re}\left(\frac{\beta_n^{\text{Im}}}{ix + \alpha_n} \right), \tag{19}$$

where $N = L + M$.

2.4 Rational approximation of the linear operator

To see the relationship between the approximation of e^{ix} with $e^{\tau\mathcal{L}}$, with $\tau = (t - t_0)$, remember that the oscillatory problem $\mathcal{L}$ has only purely imaginary eigenvalues. Therefore it maybe decomposed as $\Sigma\Lambda\Sigma^{-1}$, where Λ is a diagonal matrix in complex space containing the purely imaginary eigenvalues of $\mathcal{L}$ and Σ is a unitary matrix related to the eigenvectors.

Consequently

$$\begin{aligned} e^{\tau\mathcal{L}} &= \sum_{k=0}^{\infty} \frac{(\tau\mathcal{L})^k}{k!} = \sum_{k=0}^{\infty} \frac{\tau^k \Sigma\Lambda^k\Sigma^{-1}}{k!} \\ &= \Sigma\left(\sum_{k=0}^{\infty} \frac{(\tau\Lambda)^k}{k!} \right)\Sigma^{-1} = \Sigma e^{\tau\Lambda}\Sigma^{-1}, \end{aligned} \tag{20}$$

where we used

$$e^{\tau\Lambda} = \begin{pmatrix} \ddots & & \\ & e^{i\lambda_j\tau} & \\ & & \ddots \end{pmatrix},$$

and the fact that the eigenvalues are imaginary (therefore λ_n are real). Since $e^{\tau\Lambda}$ is diagonal, it can be eigenvalue-wise approximated in the same way as the function e^{ix}, with $x = \tau\lambda_n$.

Although $\mathcal{L}$ has imaginary eigenvalues, we wish to evaluate $e^{\tau\mathcal{L}}U$, which is real valued. Therefore, we will use the real approximation of e^{ix} applied to the multi-dimensional linear operator

$$e^{\tau\mathcal{L}} \approx \text{Re}\left(\sum_{n=-N}^{N} \beta_n(\tau\mathcal{L} + \alpha_n I)^{-1} \right), \tag{21}$$

where β_n is given by equation (17) and $\alpha_n = h(\mu + in)$.

We verify that directly substituting $\tau\mathcal{L}$ as ix in the rational approximation for the real part of $\exp(ix)$ and treating it as a matrix operator gives a valid approximation for $e^{\tau\mathcal{L}}$, since

$$\begin{aligned} &\sum_{n=-N}^{N} \text{Re}\left(\beta_n(\tau\mathcal{L} + \alpha_n I)^{-1}\right) = \\ &= \sum_{n=-N}^{N} \text{Re}\left(\beta_n(\tau\Sigma\Lambda\Sigma^{-1} + \alpha_n I)^{-1}\right) \\ &= \sum_{n=-N}^{N} \text{Re}\left(\beta_n \Sigma(\tau\Lambda + \alpha_n I)^{-1}\Sigma^{-1}\right) \\ &= \Sigma\left[\sum_{n=-N}^{N} \text{Re}\left(\beta_n(\tau\Lambda + \alpha_n I)^{-1}\right) \right]\Sigma^{-1} \\ &\approx \Sigma e^{\tau\Lambda}\Sigma^{-1} = e^{\tau\mathcal{L}}, \end{aligned} \tag{22}$$

where I is the identity operator and we used that the expression in brackets is a rational approximation for the real part of the exponential of each eigenvalue, $e^{i\tau\lambda_j}$, and therefore an approximation to $e^{\tau\Lambda}$.

2.5 Properties of the method

To obtain an accurate representation of $e^{\tau\mathcal{L}}$, the rational approximation must take into account the spectra of the operator $\mathcal{L}$, as shown in equation (22). If $\bar{\lambda} = \max_n |\lambda_n|$ is the maximum absolute eigenvalue of $\mathcal{L}$, then we need to build the rational approximation in such way that $\tau\bar{\lambda} < hM$. Assuming a fixed value of h, given by the accuracy desired for the step A of the procedure to build the rational approximation, M has to be chosen large enough to cover the whole spectra of $\mathcal{L}$. Also, if a larger time step (τ) is required, larger M values are also expected. Analytical expressions relating the error incurring from different choices of h and M can be found in Haut et al. (2015). In section 4, we will numerically explore optimal choices for these parameters.

The coefficients α_n and β_n are conjugate symmetric (Hermitian) about the central pole, such that $\alpha_{-n} = \bar{\alpha}_n$ and $\beta_{-n} = \bar{\beta}_n$, and $\text{Im}(\alpha_0) = \text{Im}(\beta_0) = 0$. Furthermore, it holds that $\overline{(\mathcal{L} + \alpha I)^{-1}U} = (\mathcal{L} + \overline{\alpha}I)^{-1}U$ with the overbar denoting the complex conjugate. This allows us to reduce the computational amount almost by a factor of two giving the real valued solution

$$e^{(t-t_0)\mathcal{L}}U(t) \approx \sum_{n=0}^{N} \mathrm{Re}\left(\gamma_n(\tau\mathcal{L}+\alpha_n I)^{-1}U(t_0)\right) \quad (23)$$

with

$$\gamma_n = \begin{cases} \beta_0, & n=0 \\ 2\beta_n, & n>0 \end{cases}$$

Therefore, the rational approximation requires the solution of $N+1$ linear systems for U of the form

$$\left(\mathcal{L}+\tau^{-1}\alpha_n I\right)U = \tau^{-1}U(t_0), \quad (24)$$

where the time step is given by $\tau = (t-t_0)$.

Some care needs to be taken in the choice of the method to be used to solve these systems. Since the rational approximation requires all eigenvalues to be purely imaginary, the numerical scheme should be able to preserve this property. We will discuss stability constraints and strategies to deal with numerical methods that could potentially produce spurious real eigenvalues in the next subsection.

2.6 Stability

We briefly analyze the stability of the time-stepping method based on repeated applications of $R_N(\tau\mathcal{L}) \approx e^{\tau\mathcal{L}}$, where the construction of the rational function,

$$R_N(\lambda) = \sum_{k=-N}^{N} \frac{\beta_k}{\lambda+\alpha_k}, \quad (25)$$

is discussed in the previous sections.

We first assume that $\mathcal{L}$ is a matrix with a purely imaginary spectrum. To do so, suppose that

$$\max_{|\lambda|\le\|\mathcal{L}\|_2} \left|R_N(\tau\lambda) - e^{i\tau\lambda}\right| \le \delta, \quad (26)$$

where $\|\mathcal{L}\|_2$ is the operator 2-norm, and δ is an error bound that can be as small as required depending on parameter choices of the rational approximation.

It then follows that

$$\begin{aligned}\max_{\lambda\le\|\mathcal{L}\|_2} |R_N(\tau\lambda)| &= \max_{\lambda\le\|\mathcal{L}\|_2} \left|e^{i\tau\lambda} + \left(R_N(\tau\lambda) - e^{i\tau\lambda}\right)\right| \\ &\le 1+\delta. \end{aligned} \quad (27)$$

Therefore, repeated applications (n) of R_N are bounded by

$$\max_{\lambda\le\|\mathcal{L}\|_2} \|(R_N(\tau\lambda))^n\| \le (1+\delta)^n \le e^{n\delta}. \quad (28)$$

Consequently, as long as the number of time steps n satisfies $n \lesssim 1/\delta$, the method is stable (regardless of the size of τ).

The time stepping error grows linearly in the number n of time steps, so there is a natural accumulation of errors $\mathcal{O}(n\Delta T)$ when more time-steps are taken. Therefore, an accuracy degradation is expected if more than $n = \mathcal{O}(1/\Delta t)$ time steps are taken, so the stability constraint for the scheme is not overly restrictive for reasonably accurate solutions.

As discussed in Haut et al. (2015), it is possible to avoid the above restriction on n - as well as to extend the above analysis to the case where $\mathcal{L}$ is a possibly unbounded operator - by using a rational function filter $S(\lambda)$ such that

$$\max_{\lambda\in\mathbb{R}} |S(\lambda)R_N(\tau\lambda)| \le 1. \quad (29)$$

Defining $\tilde{R}_N = SR_N$ as the filtered rational approximation, it then follows that

$$\left\|\tilde{R}_N(\tau\mathcal{L})\right\|_2 \le \max_{\lambda\le\|\mathcal{L}\|_2} \left|\tilde{R}_N(\tau\lambda)\right| \le 1, \quad (30)$$

and the time stepping method is unconditionally stable. See Haut et al. (2015) for further details. Importantly, this also allows one to use a rational approximation $R_N(t\lambda)$ that only approximates $e^{i\lambda}$ on part of the spectrum of $\mathcal{L}$.

For numerical solvers of equation (24) that do not preserve purely imaginary eigenvalues, unstable modes can also be treated with the rational filter. The filter may be applied as a rational approximation as well, which contributes to the overall parallelism-in-time. For the discretization methods adopted in this work, it was not necessary to apply any filtering for stability reasons.

3 Benchmark description

The purpose of this section is to examine the feasibility and limitations of gaining more parallelization with REXI for simple, but realistic problems. We therefore use the linear rotating shallow water equations for our benchmarks because they are widely used in the development of algorithms for the weather and climate models. Although full predictive models (e.g. weather and climate ones) are non-linear, the linear part of the systems plays an important role in determining the oscillatory behaviour (and time step limits) of the problem.

3.1 Problem description

We will solve the rotational linear shallow water equations (see also Haut et al. (2015)), linearized with respect to a rest state with mean water depth of $\bar{\eta}$, and defined for perturbations of height η and velocity components (u,v). The linear operator $\mathcal{L}$ may be written as

$$\mathcal{L} = \begin{pmatrix} 0 & -\bar{\eta}\partial_x & -\bar{\eta}\partial_y \\ -g\partial_x & 0 & f \\ -g\partial_y & -f & 0 \end{pmatrix}, \quad (31)$$

which results in a matrix with purely imaginary eigenvalues for the discretisation methods used in this work. A bi-periodic simulation domain is used and we wish to solve

$$U_t = \mathcal{L}U, \quad (32)$$

where

$$U = (\eta, u, v)^T. \quad (33)$$

If not otherwise stated, we set $g = \bar{\eta} = f = 1$ and the domain will be given by the bi-periodic unit square $\Omega = [0,1]^2$.

For the rational approximation, linear solutions of problems of the form

$$(\mathcal{L}+\alpha I)U = U_0, \quad (34)$$

are obtained in the following way.

First, we note that the momentum equations lead to a problem of the form

$$A_\alpha V = V_0 + g\nabla\eta \tag{35}$$

with $V = (u, v)$ and analogously $V_0 = (u_0, v_0)$, and

$$A_\alpha = \begin{pmatrix} \alpha & f \\ -f & \alpha \end{pmatrix}. \tag{36}$$

The solution is

$$V = A_\alpha^{-1}(V_0 + g\nabla\eta). \tag{37}$$

Further reformulations with the assumption of a constant f and using $\kappa = f^2 + \alpha^2$ lead to

$$\Delta\eta - \kappa^2\eta \quad = \quad r_0 \tag{38}$$

where

$$r_0 = \frac{\kappa}{\alpha}\eta_0 - \bar{\eta}\frac{f}{\alpha}\zeta_0 + \bar{\eta}\delta_0, \tag{39}$$

$\delta = u_x + v_y$ is the wind divergence, $\zeta = v_x - u_y$ is the wind (relative) vorticity and $\Delta\eta = \eta_{xx} + \eta_{yy}$ is the Laplacian of the fluid depth.

Therefore, to solve the linear problems, given as in Eq. (34), we first solve the associated Helmholtz problem for η (from equation (38)), and then calculate the velocities directly from equation (37).

3.2 Numerical methods

In this section we discuss the numerical methods used for the studies in this paper. An overview of the methods is presented in Table 1.

Time integration: Two different time integration methods are used in this work:

- RK4 refers to a Runge-Kutta 4th order time-stepping method.
- REXI denotes the time integration with the rational approximations, see Sec. 2.

The Runge-Kutta schemes are well understood methods for time integration and serve as the baseline for the comparisons with the alternative time stepping methods.

Space discretization: The benchmarks were conducted based on two different (and relevant) discretization strategies in space. The derivatives in the linear operator L can be based on:

- *Finite differences*: Here, the standard second order differences are used (e.g. a $[-1, 0, 1]$ stencil for $\frac{d}{dx}$)
- *Spectral derivatives*: The derivatives computed in spectral space. In this case, the variables are transformed to Fourier space, the derivatives are calculated, and the variable is reverted to physical space (e.g. $\frac{d}{dx}e^{i2\pi x} = 2i\pi e^{i2\pi x}$).

We made these choices for the following reasons. First, finite difference schemes are one common way for solving PDE problems in general, and are particularly relevant for weather, climate and ocean models (Wood et al. 2014; Madec 2014). Second, spectral methods allow high order accuracy with great efficiency if used with optimized fast Fourier transform algorithms and some state-of-the-art weather forecasting systems use spectral methods based on spherical harmonics (Barros et al. 1995; Mozdzynski et al. 2015).

Grid: For the spatial grids we use the following schemes:

- *Collocated A-grid*: All variables are placed at the cell center. This method is used for *all REXI time stepping methods* and for the *RK4 time stepping method with spectral spatial discretization*.
- *Staggered C-grid*: The potential is placed at the cell center and the velocity components at the cell edges. Furthermore, computations are based on the vector invariant formulation. This placement is used for *RK4 time stepping methods with finite-differences*.

The A/C naming convention follows Arakawa and Lamb (1977) convention used in ocean and atmospheric models and has been discussed for many years. As an example, Randall (1994) describes how the use of staggered C grids enable better representation of fast waves existing in the shallow water equations and avoiding computational spurious modes. On the other hand, collocated grids (A-grids) allow the use of efficient FFT solvers.

Solver for $(\mathcal{L} + \alpha I)^{-1}$: For the REXI timestepping scheme, a linear system solver is required. More precisely, we need to solve a Helmholtz problem, given by equation 38. For this purpose, we implemented two different solvers:

- *Finite differences*: Finite difference operators are implemented via a convolution operation of the finite difference stencil in Fourier space which is then solved directly.
- *Spectral basis functions*: Spectral derivative operators are computed in Fourier space.

The Helmholtz problem is discretized into an algebraic linear system, which is then solved using a Fast Helmholtz solver (direct Fourier solver) using spectral basis functions for spectral methods or convolutions of the stencils for finite differences, see Swarztrauber and Sweet (1996). Because of its efficiency, we decided to use the Fast Helmholtz solver. However, further investigation on more generic solvers is required and we discuss this in Section 8.

3.3 Initial conditions

We use two different initial conditions for our evaluation of the REXI approach.

Wave scenario The wave scenario is initialized with the following values for the perturbation height η and velocity components (u,v),

$$\begin{aligned} \eta(x,y) &= \sin(2\pi x\omega^x)\cos(2\pi y\omega^y) \\ &\quad -\frac{1}{5}\cos(2\pi x\omega^x)\sin(4\pi y\omega^y) && (40) \\ u(x,y) &= \cos(4\pi x\omega^x)\cos(2\pi y\omega^y) && (41) \\ v(x,y) &= \cos(2\pi x\omega^x)\cos(4\pi y\omega^y) && (42) \end{aligned}$$

with mean water depth given by $\bar{\eta} = 1$ and $(\omega^x, \omega^y) = (2, 1)$ (see Fig. 3).

These initial conditions can be represented in spectral (Fourier) space, therefore the errors are very small when the spectral solver is used for the REXI approach. This ensures that we can use this set of initial conditions to understand the accuracy of the rational approximation method.

ID	**Space discretization**	**Grid**	**Time integration**	**Solver for** $(\mathcal{L}+\alpha I)^{-1}$
(A)	Finite differences	C-grid	RK4	-
(B)	Finite differences	A-grid	REXI	Finite differences in spectral space
(C)	Spectral methods	A-grid	RK4	-
(D)	Spectral methods	A-grid	REXI	Spectral basis functions in spectral space

Table 1. Overview of the different combinations of spatial discretizations, grids, timestepping methods and linear solvers for the REXI terms.

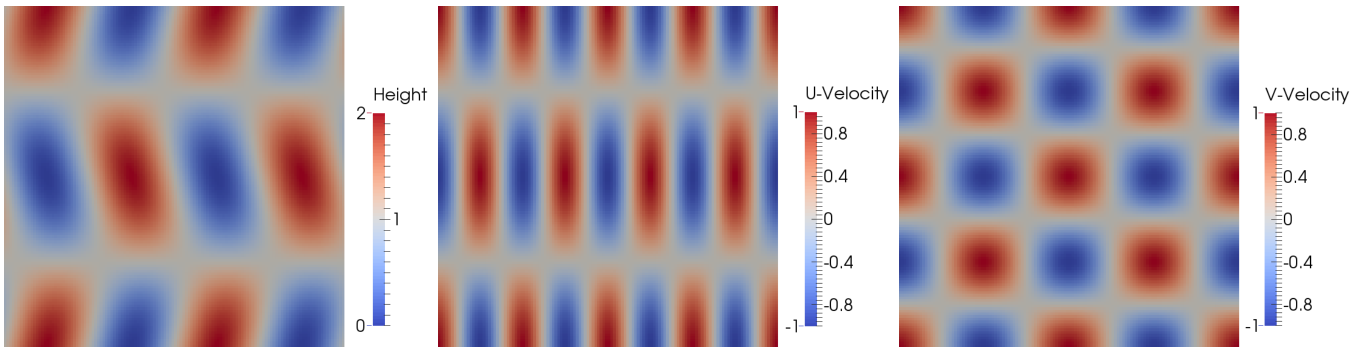


Figure 3. Initial conditions for the wave scenario given by parameters $(\omega^x, \omega^y) = (2, 1)$, see Sec. 3.3. The variables shown are the height perturbation (η), velocity in x-direction (u), velocity in y-direction (v) from left to right.

Gaussian scenario In this case we use an initial *Gaussian* function for the fluid height perturbation,

$$\eta(x, y) = e^{-50(x^2+y^2)}, \tag{43}$$

assuming a background constant height of $\bar{\eta}$ and zero-valued velocity fields as initial conditions. This initial condition is not exactly representable in spectral (Fourier) space.

3.4 Error analysis

For the comparisons of the numerical results in Section 4, we use the Root Mean Square (RMS) error norm,

$$E_{rms}(f) = \sqrt{\frac{\sum_{ij}\left(f_{ij} - \tilde{f}_{ij}\right)^2}{N_x N_y}}, \tag{44}$$

which is based on the discrete computed solution f_{ij} for the points given by (x_i, y_j) and on the reference solution calculated at grid points given by $\tilde{f}_{ij}$ the resolutions in x and y directions given respectively by N_x and N_y.

For the performance results in Section 5, we use the Maximum error norm which is given by

$$E_{max}(f) = \max_{ij} |f_{ij} - \tilde{f}_{ij}|. \tag{45}$$

The reference solution is given by a generalized analytical solution of the linear operator (see Embid and Majda (1996)) and we always compute the error with respect to the height (η) field.

4 Numerical evaluation

This paper is based on a new method to accelerate computations. However, the understanding of the relationship between different parameter/method choices is important to ensure accurate as well as efficient computations.

A key part is the understanding of how the REXI parameters M and h interact with each other and with different methods or model parameters (such as Coriolis parameter, gravity and mean water depth). In the following sections, we conduct a sequence of parameter studies with successively increasing complexity, with the aim of understanding the accuracy versus workload relationship.

4.1 REXI parameters (h, M)

We start by analysing the REXI parameters, h, M and L.

In Fig 4 we analyse the dependency between M and h for both solvers (finite differences (B) and spectral (D)) investigated with the REXI method. It shows that h needs to be chosen sufficiently small (at least $h < 2$) and M sufficiently large (at least $M > 32$). As expected, smaller values of h require larger values of M to preserve accuracy, which reveals a triangular region of most accurate results.

Also as expected, we note that REXI has higher accuracy using a spectral method (lower panel of Fig 4). For the finite-difference method (upper image), we compute the differential operators with finite differences, which results in a lower accuracy of the solution compared to the spectral method.

For efficiency reasons, we are interested in reducing the amount of total workload which is given by M. However, reducing M requires increasing h and the area with acceptable errors gives an upper limit of h. We will adopt a fixed parameter $h = 0.2$ throughout the remainder of this work, which is large enough to reduce the workload, but small enough to produce accurate results given $M > 32$.

4.2 Grid resolution (N)

Using standard time stepping methods, an increasing resolution would require additional computations due to the increase in spatial workload, and also an increase in the number of time steps, due to time step size restrictions (CFL condition).

With the REXI approach, an increasing resolution would also require additional computations due to the increase in

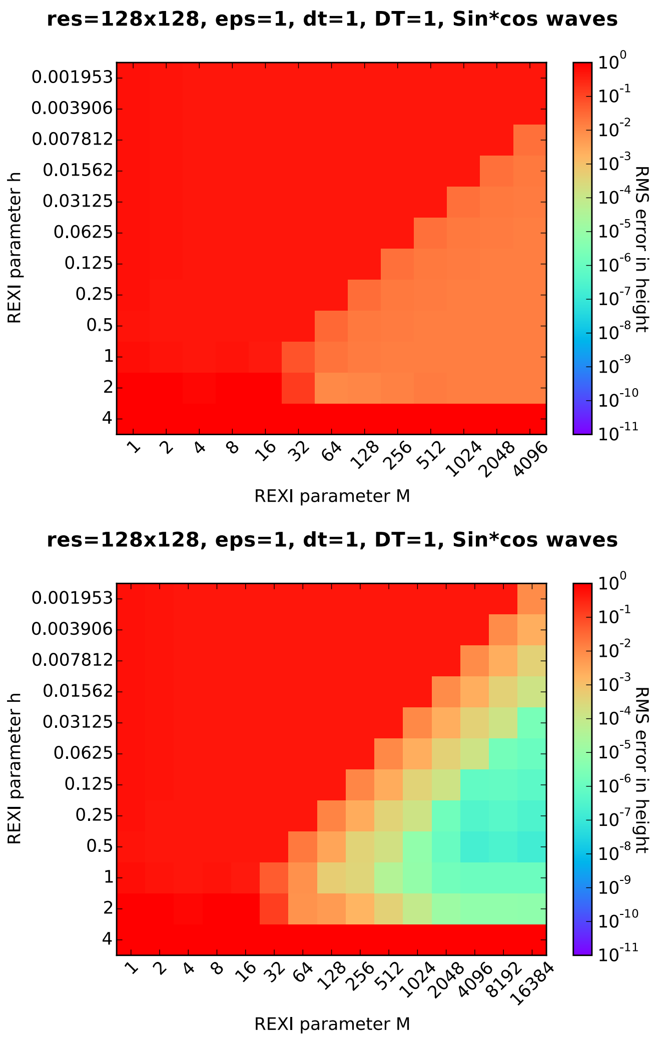


Figure 4. Error analysis (RMS) of the REXI scheme with respect to varying h and M using two methods. Top image shows method (B) which uses a finite-difference solver. Bottom image shows method (D) which uses spectral solver. The errors are those of a single large time step of size, $\tau = 1$, for the waves initial conditions scenario. We can observe limiting values for h and M and also that M is linearly inversely proportional to h.

spatial workload, but the time step size can be maintained the same as long as additional REXI terms (M) are included. We show in Fig. 5 the errors for varying resolution for the Gaussian scenario, which reveal that additional resolution implies a larger frequency spectrum to be represented (larger $\bar{\lambda}$) and therefore larger M. The main advantage of REXI is that the extra workload of having more REXI terms can be done massively in parallel, allowing to cope with wall-clock-time restrictions, whereas in standard time stepping methods the reduced time step size significantly impacts the wall-clock time of computation.

4.3 Initial condition wave-number (ω)

Next, we analyse a possible dependency of the REXI parameters depending on the initial condition. To do so, we vary the (ω^x, ω^y) parameters in our *waves* scenario by setting $(\omega^x, \omega^y) = (2\omega, \omega)$. The error plots are given in Figure 6 (top panel) and we see indeed a dependency on the frequency of the initial conditions. As expected, initial conditions

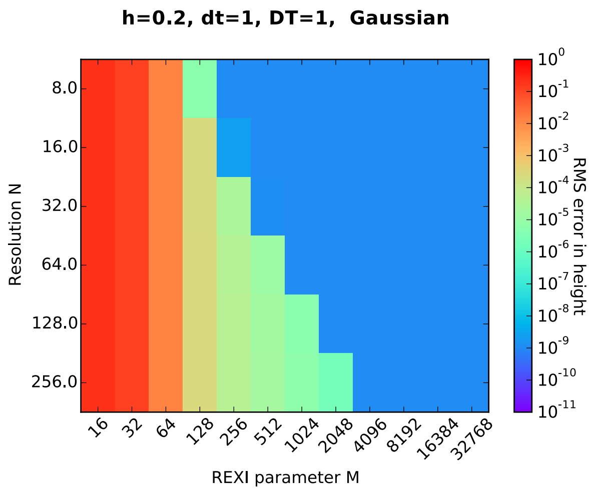


Figure 5. Error analysis (RMS) of the REXI scheme with respect to varying resolution N and M using method (D)-spectral solver. The errors are those of a single large time step of size $\tau = 1$, for the Gaussian initial conditions scenario. We can observe that M linearly depends on the resolution, so that higher resolutions require larger values of M.

containing higher wave-numbers require that REXI has more terms (larger M).

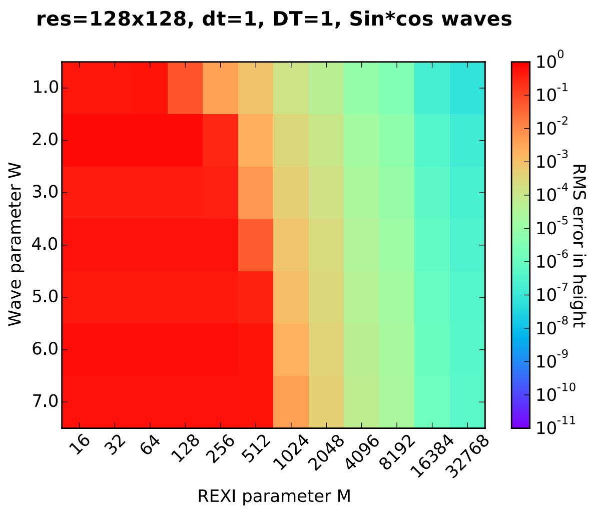


Figure 6. Error analysis (RMS) of the REXI scheme with respect to varying wave-numbers of the initial condition (ω using method (D)-spectral solver. The errors are those of a single large time step of size $\tau = 1$, for the waves initial conditions scenario. We can observe that M linearly depends on the wave-number, so that higher ω requires larger values of M.

4.4 Time-step size τ

We investigate the effect of different time step sizes (τ) in M. Error results are shown in Fig. 7. Due to the known relation that $\bar{\lambda}\tau < hM$, for fixed resolution and fixed h, larger time step sizes require more REXI terms (larger M).

4.5 $\mathcal{L}$ stiffness $(g, f, \bar{\eta})$

For standard explicit time stepping methods, the time step size is limited by the speed of the wave propagation (CFL condition needs to be respected). A faster wave propagation would then force smaller time step sizes for stability reasons.

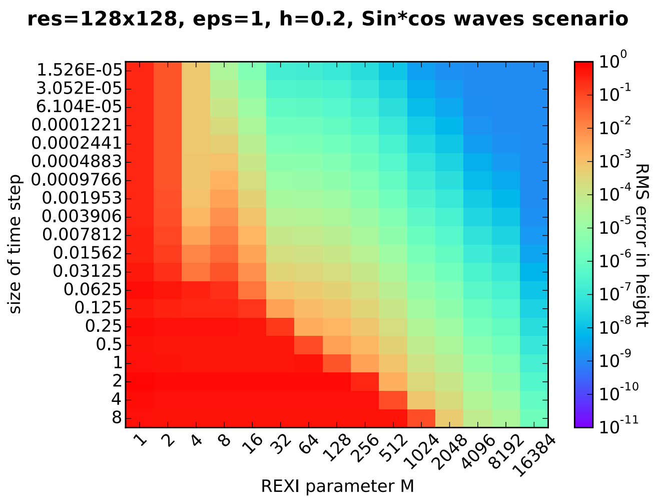


Figure 7. Dependency of REXI parameter M on varying time step size. To compute larger time steps, also M has to be increased.

This propagation speed also depends on the oscillatory stiffness of our operator, given by g, f and $\bar{\eta}$.

We test for such a potential dependency by varying one of the paramters f, g, $\bar{\eta}$ and keeping the other two values fixed. Error heatmaps with varying parameter of linear operator and REXI parameter M are given in Fig. 8. Here, we can observe again a dependency of the REXI parameter M on the frequencies which describe the stiffness of the system. Additionally, the varying frequency requires increasing M only after a certain point and we discuss this briefly: We account for that by the other oscillatory parts in the linear operator. These oscillations dominate the error as it can be seen by the constant error over a range of the varying parameter for a constant M. After the parameter which we vary reaches a frequency which is similar to the one in the other frequencies in the operator, the varying frequency starts dominating the error, hence also requires increasing the REXI parameters M.

Next, we have a look at dispersion effects of the REXI approximation. We use a spectral representation and decompose $\mathcal{L}$ into a system of independent equations for each spatial frequency. Furthermore, we assume that an accurate solver (up to machine precision e.g. with the fast Helmholtz solver) is used for solving the inverse problem in each REXI term. The linear operator $\mathcal{L}$ can be decomposed in spectral space to

$$e^{\tau\hat{L}} = \hat{\Sigma} e^{\tau\hat{\Lambda}} \hat{\Sigma}^{-1}$$

with the symbol $\hat{}$ denoting the spectral representation (see also Embid and Majda (1996)). Then, we can directly compute our solution by using

$$\begin{aligned} \hat{U}(\tau) &= e^{\tau\hat{\mathcal{L}}}\hat{U}(0) \\ &= \hat{\Sigma} e^{\tau\hat{\Lambda}} \hat{\Sigma}^{-1}\hat{U}(0) \end{aligned}$$

and obviously no error is introduced so far. We use REXI to approximate the frequencies given by $e^{\tau\hat{\Lambda}}$, yielding

$$\begin{aligned} \hat{U}_k(\tau) &= e^{\tau\hat{\mathcal{L}}_k}\hat{U}_k(0) \\ &\approx \hat{\Sigma}_k \left[\sum_{n=0}^{N} Re\left(\gamma_n^{Re}(\tau\hat{\Lambda}_k + \alpha_n I)^{-1}\right) \right] \hat{\Sigma}_k^{-1}\hat{U}_k(0) \end{aligned}$$

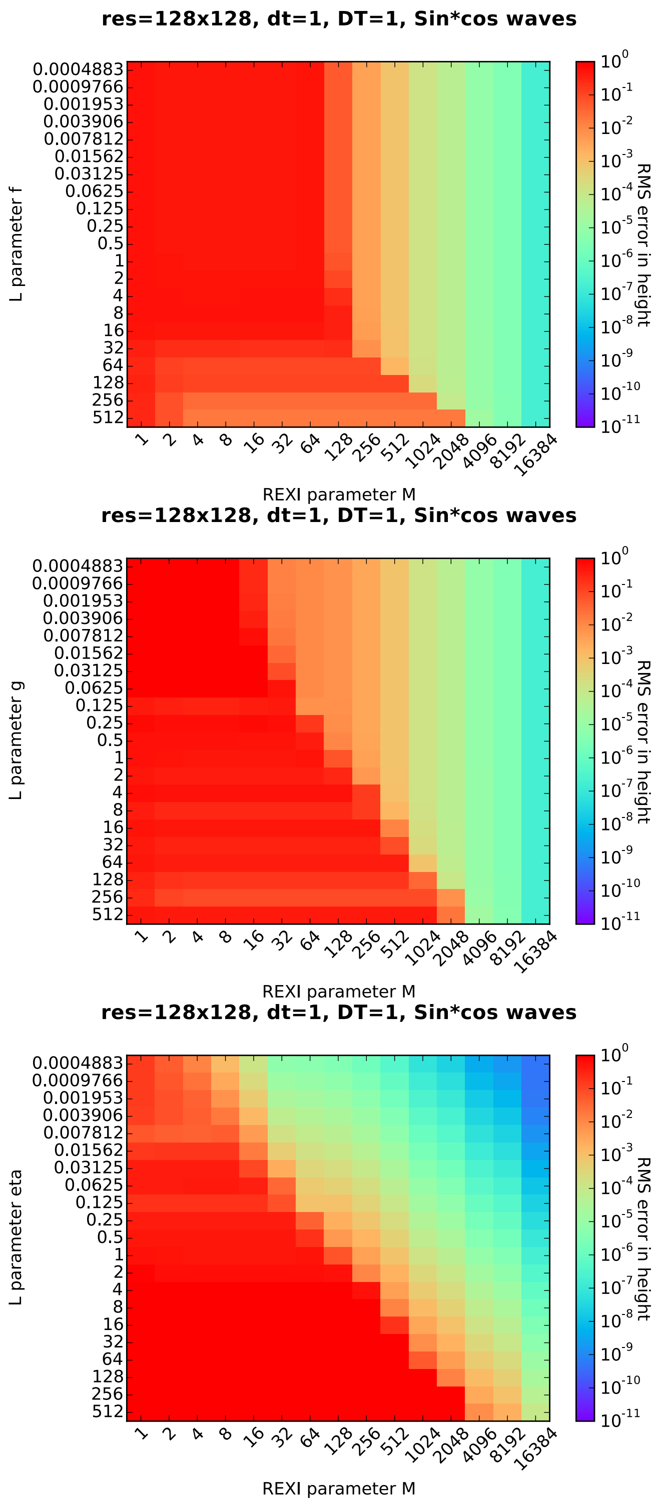


Figure 8. Error sensitivity study with varying parameters f (Coriolis frequency), g (gravitational constant), and $\bar{\eta}$ (average height) and parameter M on varying frequency ϵ of linear operator (larger ϵ leads to more oscillations) with resolution 128^2. The varying frequency requires increasing M only after a certain point. This is because the frequencies which are kept constant in the linear operator $\mathcal{L}$ are dominating the error.

with the selected spatial frequencies denoted by $k = (k_1, k_2)$ and time frequencies by σ. We focus on the case of $k_1 \neq 0$ and $k_2 \neq 0$, yielding the eigenvalues for the vortical/geostrophic mode $\sigma_0 = 0$ and inertia-gravity (Poincaré) modes $\sigma_{\pm1} = \pm\sqrt{4\pi^2(\bar{\eta}gk_1^2 + \bar{\eta}gk_2^2) + f^2}$. With the REXI parameter M being related to the fastest waves, we can explain the dependencies of the REXI parameter M on f, $\bar{\eta}$ and g as follows: The stiffness of the operator is linearly depending on the Coriolis frequency f, hence $M \propto f$ and we can observe this linear dependency in the top image in Fig. 8. Regarding the dependency on $\bar{\eta}$ and g, we expect that the REXI parameter is depending on both coefficients by $M \propto \sqrt{\bar{\eta}g}$. This behaviour can be also observed in the middle and bottom image in Fig. 8 by a steeper slope of the boundary of sufficiently accurate solutions.

5 HPC performance evaluation

The current trends in high-performance computing go towards massive parallelism and the parallelization of applications was so far mainly focusing on a parallelization-in-space. In this work, we consider problem sizes which are assumed to be not scalable at all (see Sec. 5.2). To overcome such scalability limitations, one solution was presented with REXI (see Sec. 2). This allows us to run simulations on thousands of cores and this section is on evaluating the potential reduction in wall-clock time.

We conducted all performance benchmarks on the CoolMUC2 cluster which is based on Intel(R) Xeon(R) CPU E5-2697. Each socket is equipped with 14 physical cores, 2 sockets are installed on each compute node and hyper-threading is deactivated. All computations were done in the SWEET software which is freely available (Schreiber et al. 2016). The Intel(R) icpc compiler was used for compilation with IntelMPI for distributed parallelization. For (hybrid MPI+X) threaded parallelization we pinned the threads to the cores with compact affinities. We used the FFTW (Frigo 1999) as a third party library compiled with Intel(R) icpc.

All simulations in this section were computed over $\Delta T = 50$ simulation seconds. We used Runge-Kutta 4 (RK4) time stepping method for the finite-difference and spectral method. Using a CFL ≈ 0.22, this resulted in time step sizes of about 0.0017 seconds with RK4. A reduction of the CFL would result in decreasing the error in time with order 4 but also an increase in the number of required time steps. These additional time steps would then directly lead to an increase in wall clock time with traditional time stepping methods.

For the REXI time stepping, the size of each time step is $\Delta t = 5$ simulation seconds, hence 10 time steps are done in total.

5.1 Determining REXI's M parameter for performance studies

We first conducted a parameter study with varying M and the wave-like initial conditions to assure that all parallel REXI computations are of sufficient accuracy. Results for spatial approximation with spectral methods and finite-differences are given in Fig. 9.

We first analyse the upper plot, which shows the errors for the finite-difference method. In contrast to the computations in the previous Section, these simulations are executed over a relatively long simulation time of 50 seconds with 10 seconds per time step. With the finite difference method, a reduction in the error only starts with a resolution $\geq 256 \times 256$. However, for a resolution of 128×128 which is of our interest, a convergence is reached for $M \approx 2048$. We link this to the results of the previous section: Figure 4 gives for $h = 0.2$ a lower limit of $M \geq 256$ poles for a single simulation second and same simulation parameters. Figure 7 shows that taking time steps which are 5 times larger also requires increasing M by a factor of 5. Hence, $M \geq 5 \cdot 256$ poles are required for a convergence which confirms the REXI results being sufficiently accurate for $M = 2048$.

The lower plot shows the errors for the spectral method. With this method, we can observe that the errors are steadily decreasing with increasing REXI parameter M. Furthermore, by adding more terms in the REXI approximations we can also gain results which are more accurate compared to spectral methods. Here, the spatial errors are obviously not dominating. We account for that by the errors in the RK4 time stepping scheme. Using the REXI approximation allows to overcome the errors introduced with a RKn time stepping method and we like to remind that this reduction is paid by an increase in the degrees of parallelization via the increasing M.

In contrast, reducing the errors with standard time stepping methods can be e.g. accomplished by decreasing the time step size. However, this would directly lead to an increase in total simulation time, whereas with REXI, this leads to an increase in parallelism.

5.2 Space-parallelization results (non-REXI)

To show that we are tackling a strong scalability problem, we conducted scalability tests with resolutions of 32^2, 64^2 and 128^2.

The results for the finite-difference methods on a (staggered) C-grid are given in top image in Fig. 10. We omitted plotting results for larger resolutions due to caching effects resulting in a non-monotone scalability. Only problem sizes are considered in this work which are not influenced by these caching effects. These are limited in their scalability on a single shared-memory NUMA node.

Comparing these results with the ones for spectral methods on an A-grid (bottom image in Fig. 10), we can observe that spectral methods are by far more limited in their scalability. This can be explained by the computational complexity of spectral methods: First of all, the operations can be accomplished element-wise in spectral space without requiring (computational more intensive) stencil operations. This allows an efficient implementation with a parallel-for loop and element-wise operations (mainly multiply and add operations). Second, the afore mentioned issue also results in a low workload, hence leading to an increase of threading overheads.

For both methods, we can observe a clear scalability limitation even for a single shared-memory node with standard parallelization-in-space methods. In the next sections, we evaluate the REXI method as one of the parallelization-in-time strategies to overcome these scalability limitations.

5.3 Time-parallelization results (REXI)

We first evaluate the time-to-solution with a parallelization in the time dimension only. The simulation time for finite-difference and spectral methods with different REXI M parameters is plotted in Fig. 11. For an improved comparison

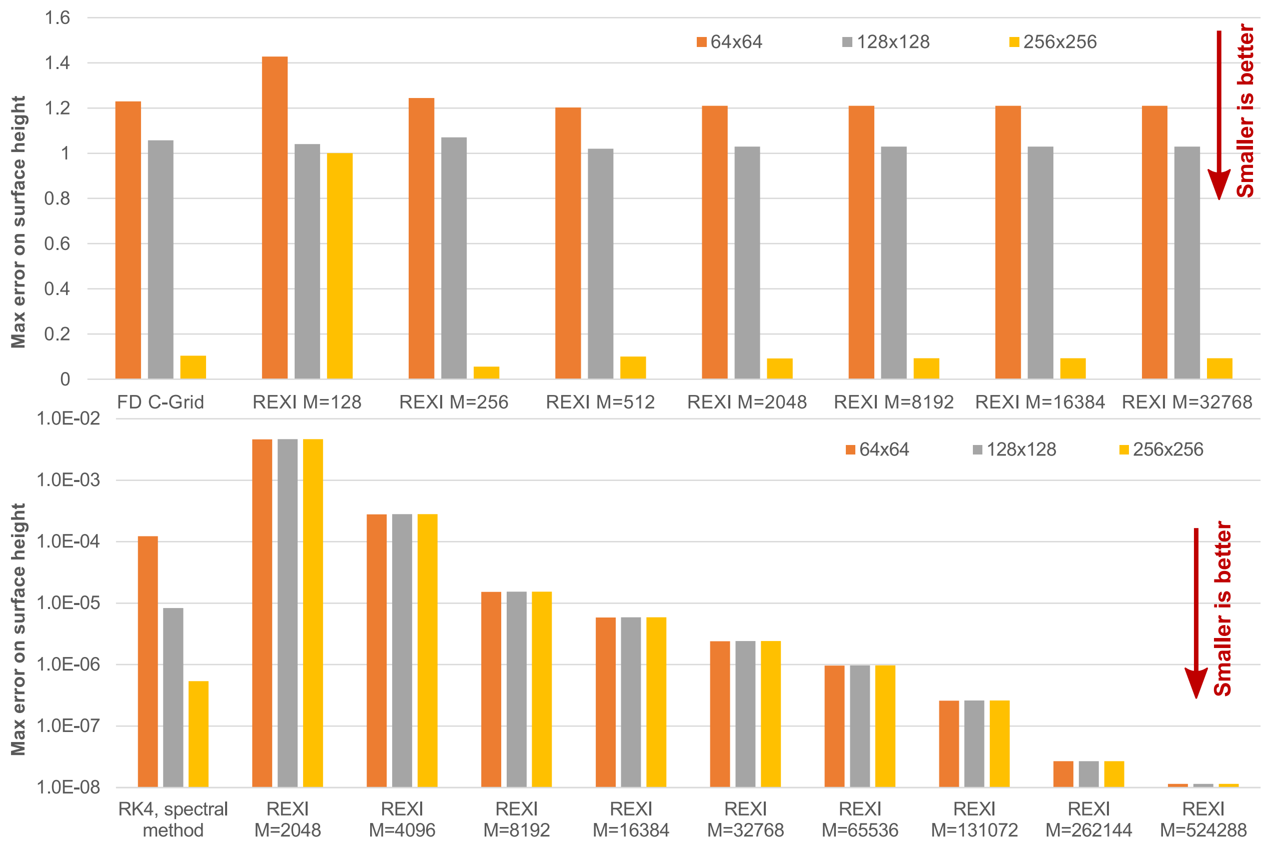


Figure 9. Maximum error norm computed on surface height for different REXI parameters M. Results are plotted for spectral methods (bottom) and finite-difference (top). The errors are identical for different resolutions since the initial condition can be error-free represented in spectral space.

with the RK4 time stepping method, we also plotted the time-to-solution. Here, we used the best time-to-solution for the numbers of cores which would typically lead to reduced scalability.

Regarding the REXI method, we can observe a convergence of the time-to-solution plots at certain number of cores. Using 112 cores, all the maximum amount of workload assigned to each compute core depends on the number of M. With this number of cores and for REXI parameters $M = (128, 256, 512)$, the maximum workload assigned to a compute rank is $(2, 3, 5)$ REXI terms, respectively. Therefore, we also observe different runtimes for the different REXI terms. However, this changes when increasing the number of cores. E.g. if the number of cores is exceeding the number of REXI terms (e.g. at 896 cores), the maximum workload on each compute rank is one and all time-to-solution plots are matching.

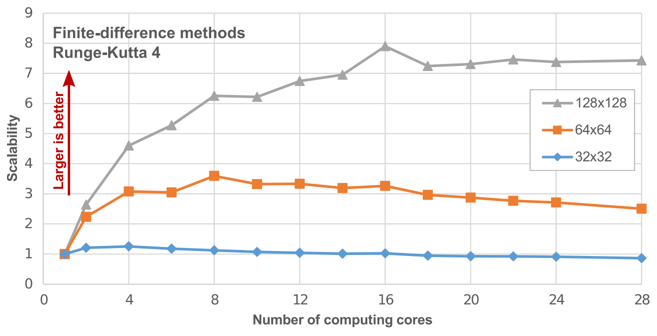
Finite-difference methods: For the finite-difference methods, we can observe a significantly faster computation time with the REXI approach compared to the RK4 time stepping method. Using only a single core, we get a simulation time of 332.19 secs with the RK4 time stepping and 28.71 secs with the REXI time stepping method. This results in a performance improvement of $11.57\times$. Activating the MPI parallelization with REXI, we get a reduction of the time-to-solution of $\frac{332.19}{0.2210} = 1503.0\times$.

Spectral methods: With the spectral method, the REXI approach is not able to be time-to-solution competitive with a single core execution. We account for that by the high spatial accuracy of the spectral method which also requires the REXI method representing more time-frequencies accurately in contrast to the finite-difference method. For the domain resolution of 128×128, we require at least $M = 16384$ REXI terms. Here, the REXI method is only competitive using at least 4 cores for a parallelization-in-time (58.2 secs with REXI vs. 67.4 secs. with RK4 time stepping). However, we can continue to increase the number of cores, since the scalability limitation is given at $M + L + 1 = 16384 + 11 + 1 = 16396$ cores. Using 3584 cores, we get a performance improvement of $\frac{67.4}{0.57} = 118.3\times$. These plots still indicate a further reduction in computation time, but we reached the resource manager limits of the compute cluster and we like to point out to Section 6 for a performance model.

5.4 Space-Time-hybrid-parallelization results (REXI)

In this section we analyze the combination of different parallelization concepts presented so far: parallelization-in-space and -in-time. Figure 12 shows plots for the finite-difference method (top) and spectral methods (bottom) and two different parallelization strategies are used:

- "*REXI M=... time+spacepar*" runs the computations with an OpenMP parallelization in space and an MPI parallelization in time. Hence, 14 threads are used in this case for a spatial parallelization on each socket and $\frac{\text{total cores}}{14}$ MPI ranks.



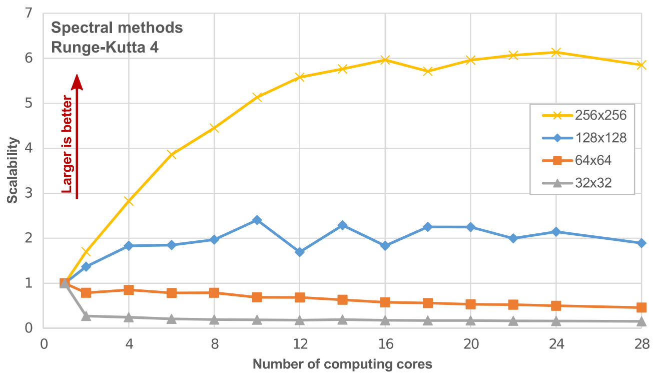
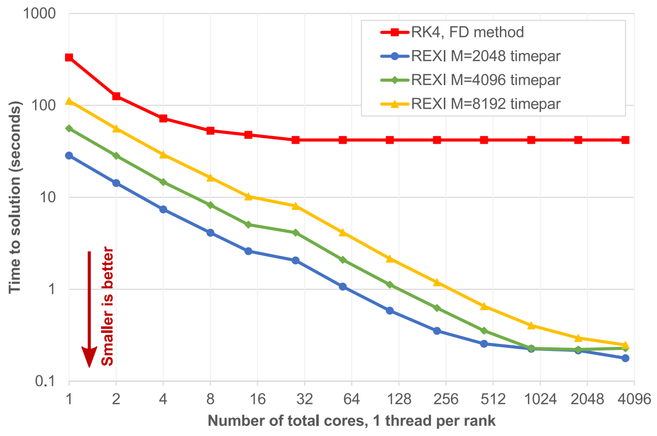


Figure 10. These examples show the classical strong-scaling limitation often found for parallelization in space only. The horizontal axes shows the number of computing cores, the vertical axes is the scalability for both (Top figure) finite differences (staggered) C-grid (method id (A)), and (Bottom figure) spectral methods on an A-grid (method id (B)). The strong scaling limitation can be seen in every line: for a fixed number of RK4 steps used in these examples, adding more processors does not lead to an increase in scalability.

- "*REXI M=... timepar*" uses an OpenMP and MPI parallelization only for the parallelization-in-time. Here, the reduce operations are executed first on each rank with a threaded parallelization. This is followed by a reduce operation on the REXI terms over all MPI ranks.

Similar to the parallelization-in-time method discussed in Section 5.3, we can observe a limitation in scalability if the number of cores exceeds 1000, independently of the utilized discretization in space (FD or spectral).

Finite-difference method: We start by discussing the results based on finite-differences in space for REXI $M = 2048$. Using only 14 threads, the parallelization-in-space-and-time ("*REXI M=2048 time+spacepar*") results in larger time-to-solution compared to using only a parallelization-in-time ("*REXI M=2048 timepar*"). Here, the overheads for the parallelization-in-space are larger than the ones for the parallelization-in-time. For increasing number of cores, this compensates (crossing of lines) after only quadrupling the number of cores to 56. With a parallelization-in-time computation on 1792 cores, we get a maximum performance improvement of $\frac{42.05}{0.2016} \approx 208.58\times$. For 1568 cores, switching from a parallel-in-time to a parallelization-in-space-and-time we get a performance improvement of $\frac{0.202}{0.140} = 1.44$. The total performance improvement is given by $\frac{42.05}{0.1401} \approx 300.14\times$ with a parallelization-in-space-and-time approach.



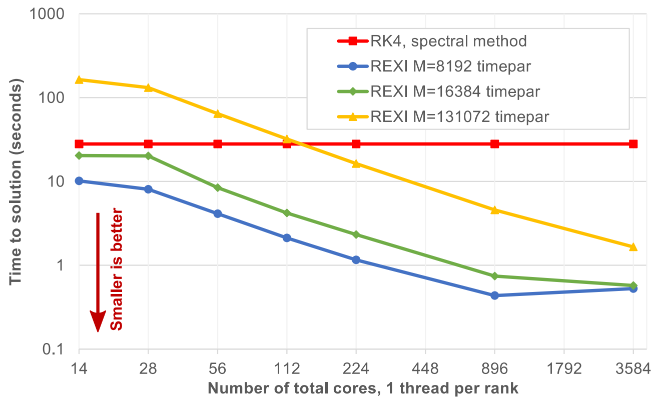
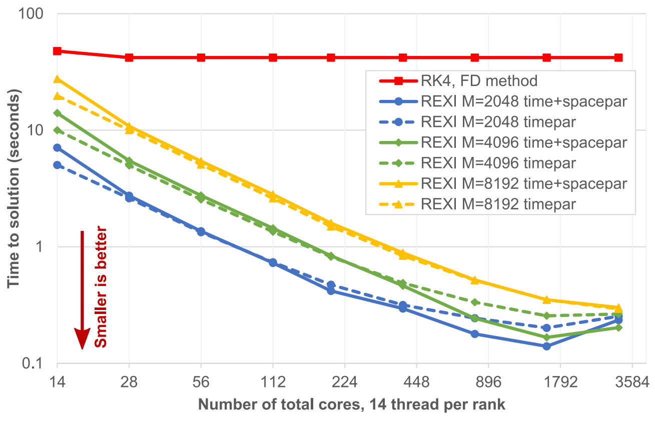


Figure 11. This figure shows REXI performance going beyond the strong scalability limit for time parallelzation only. The horizontal axes are the number of cores and the vertical axes is the time-to-solution. Top figure: This plot shows the results for finite differences. The red line shows the time-to-solution for RK4. As in Figure 10, the strong-scaling limitation can be easily seen since the reduction in time-to-solution stagnates. The other three lines are the REXI method which, for these benchmarks, always leads to an improvement in the time-to-solution. Bottom figure: Results for spectral methods are shown here. The green line with $M = 16384$ computes results of higher accuracy with REXI. This also leads to an improvement in the time-to-solution compared to the parallelization in space method which is given by the red line. Notice that the time-to-solution increases with a higher number of REXI terms, but that all three cases decrease the time-to-solution as soon as you add more processors.

Spectral methods: We observe that for the spectral method the hybrid parallelization in space and time is never or only hardly beneficial. Here, a pure parallelization-in-time clearly the best choice. We account for this in the following way: Comparing the number of REXI terms for the spectral and finite-difference methods, we require significantly more REXI terms for the spectral methods. As a consequence, this also results in compensating the parallelization overheads by the increasing amount of workload per core. The total performance improvement is $\frac{27.99}{0.3402} \approx 82.28\times$ with a parallelization-in-space-and-time approach and we expect a more significant performance improvement for a hybrid parallelization-in-space-and-time approach with a significantly larger number of computing cores.



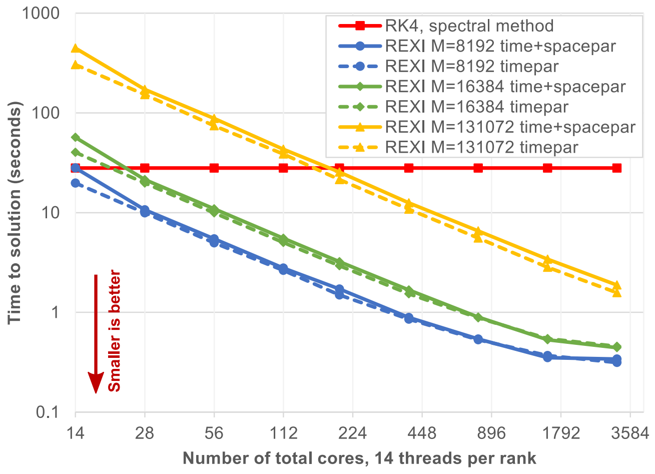


Figure 12. This figure shows REXI performance when using parallelization in both time and space (hybrid) on a grid with a 128×128 resolution. As in Figure 11, we plot time-to-solution versus the total number of processors for both the finite differences and spectral methods. *REXI M=... time+spacepar* denotes a parallelization in space (OpenMP) and parallelization in time (MPI). *REXI M=... timepar* denotes a parallelization in time only (MPI+OpenMP). Results are plotted for finite-difference (top) and spectral methods (bottom). We like to discuss the solid and dashed blue lines in the top plot: These lines show the time-to-solution for REXI with M=2048 terms. The dashed line uses only a single core for each REXI term and we can observe that they offer the best performance for a low number of computing cores. After about 2048 cores, adding more cores does not lead to further decrease of the time-to-solution. This is because no workload (in our case solving one REXI terms) can be assigned to the additionally added cores, hence they are idling. Here, activating also a parallelization in space to solve the REXI terms results in an additional reduction of the time-to-solution.

5.5 *Speedups depending on REXI time step size*

In this section we investigate the performance of REXI for varying time step sizes. Obviously, there are overheads included in running a massively parallel REXI approach. These overheads have to be compensated by the size of the time step. To get insight in the runtime behaviour for different time step sizes, we conducted several benchmarks with the time-to-solution shown in Fig. 13. The total simulation time was set to 50 seconds. With dt the time step size, $50/dt$ time steps are executed in total. We observe a very good scalability for time step sizes of at least 5.

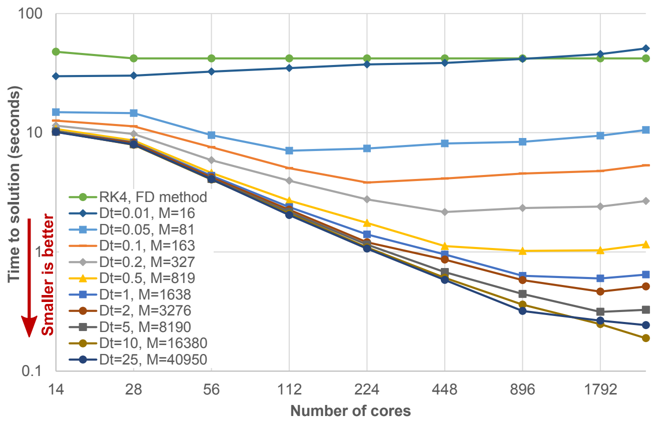


Figure 13. Time-to-solution for simulation over 50 seconds and different REXI time step sizes. Dt=... denotes the time step size for REXI and M=... the number of required REXI terms to generate competitive results. With REXI time stepping, $50/Dt$ REXI time steps are computed. The topmost green line shows the time-to-solution with a RK4 time stepping method. The dark blue line (Dt=0.01, M=16) below this one shows an increasing runtime for increasing number of cores. This is due to the low number of REXI terms (M=16) which avoids exploiting the additional number of cores since there is not enough workload. Due to parallelization overheads, the runtime is getting even longer. We see a competitive REXI time stepping method for all considered time step sizes.

For smaller time step sizes, the scalability gets successively reduced. For relatively tiny time step sizes of 0.1, the performance peak is reached at 224 cores. After this peak, the time-to-solution increases and therefore the scalability decreases. This is, because only $M = 163$ REXI terms are involved in the computation and using more than 224 cores would only result in additional communication overheads and idling MPI ranks.

Regarding the competitiveness of REXI with small time step sizes, we can consider e.g. the results with 14 cores and $Dt = 0.01$ for REXI. Using finite-differences in space and RK4 in time, we can take a time step size of about 0.0017 seconds. For this scenario, REXI is competitive for time step sizes about 5.88 times larger compared to RK4 time stepping methods.

6 HPC performance model

To develop a performance model for REXI, we separate its phases into serial and parallelizable components. We recall the REXI Equation 23 given by

$$e^{(t-t_0)\mathcal{L}}U(t) \approx \sum_{n=0}^{N} \mathrm{Re}\left(\gamma_n(\tau\mathcal{L} + \alpha_n I)^{-1}U(t_0)\right).$$

We assume a constant problem size (resolution) for each problem and denote the number of cores which are used with C. An overview of all involved components in the model is given in Table 2.

Broadcast s_B: To spread the initial conditions $U(t_0)$, a broadcast is used. Regarding the parallel communication time, we assume a logarithmic runtime for such operations and denote this by $s_B log(C)$.

Precomputation s_L *and solver* p_L: On the first glimpse, the reduce operation over the sum seems to be itself a

Serial/parallel	**Description of computation / communication**	**Runtime in model component**	**Performance model parameter**
Serial	Computation: Preprocessing for solving Helmholtz problems	$T_L(C) := s_L$	$s_L = 0.039167\ sec.$
Serial	Communication: Broadcast of initial conditions $U(t_0)$	$T_B(C) := s_B \log(C)$	$s_B = 0.004985\ sec.$
Serial / parallel	Communication: Reduce operation over sum	$T_R(C) := s_R \log(C)$	$s_R = 0.011398\ sec.$
Parallel	Computation: Solving Helmholtz problem and compute velocities	$T_W(C) := p_W \cdot \left\lceil \frac{W}{\min(C,W)} \right\rceil$	$p_W = 0.036657\ sec.$

Table 2. Description of parts in our performance model for the REXI parallelization. C denotes the number of cores and $T(C)$ the runtime for each specific part determined on the CoolMUC2 cluster. The last column contains the determined model parameters s/p as discussed in this Section.

purely parallelizable part. However, we can precompute several components with the reformulation to the Helmholtz problem (Eq. 38). We recall this equation here

$$\Delta\eta - \kappa^2\eta = r_0$$

and can observe, that the right-hand side given by r_0 is constant for each term in the REXI sum, hence can be precomputed. Therefore, such a precomputation belongs to the serial part of our computations and we denote this with s_L. We like to mention that precomputing these terms results in *less scalability, but improved overall performance*. Solving the Helmholtz problem itself and computing the velocities with the explicit formulation can then be done in parallel since all terms in the sum are entirely independent of each other. We assume that the number of REXI terms can be evenly spread across the computational resources and denote the time to compute each term by p_L (excluding the precomputable parts). Furthermore, we have to limit the parallelizable part in case that there are less REXI terms than the number of computation resources available. Let the number of REXI terms be given by

$$W = M + L + 1$$

which represents the independent systems of equations which are to be solved.

A perfect load balancing of a workload W to C would result in a runtime of $T \propto W/C$. However, each workload is assumed to be atomic, hence resulting in a runtime of $T \propto \lceil W/C \rceil$. Furthermore, there is a limitation $C \leq W$ on the maximum number of cores which yields $T \propto \left\lceil \frac{W}{min(C,W)} \right\rceil$. This finally yields

$$p_W \left\lceil \frac{W}{\min(C,W)} \right\rceil$$

for the massively parallel solver part p_L.

Reduce operation s_R: After all independent terms are solved, a reduce operation over an array of solutions is applied and we assume a runtime of $s_R \log(C)$ for these parallel computation and communication.

The overall runtime for computing one time step with REXI on C cores and REXI parameter M is then given by

$$\begin{aligned} T(C,M) &= \underbrace{s_L}_{\text{serial part}} \\ &+ \underbrace{s_B \log(C) + s_R \log(C)}_{\text{serial/parallel part}} \\ &+ \underbrace{p_W \left\lceil \frac{W}{\min(C,W)} \right\rceil}_{\text{parallel part}} \end{aligned} \tag{46}$$

with $p_L = p_W \cdot W = p_W \cdot (M + L + 1)$ describing the total parallelizable workload for REXI parameter M. The model parameters s_L, s_B, s_R and p_W are to be determined. A hybrid MPI+OpenMP parallelization and parallelization-in-time only with finite-difference discretization in space for $M \in \{2048, 4096, 8192\}$ is used. The wall-clock times $T(C)$ for each phase listed in Table 2 were measured separately. We computed the performance model parameters from Table 2 by inverting the corresponding equations. The parameters which describe all models were then determined with a weighted averaging: Since the number of cores for the performance studies are not evenly distributed, using a standard averaging would lead to a bias towards lower number of cores. Therefore, we used a weight $w(C,M)$ for each parameter which is based on the reciprocal of the total runtime $T(C,M)$, hence

$$w(C,M) = \frac{\frac{1}{T(C,M)}}{\sum_C \frac{1}{T(C,M)}}.$$

Then, e.g. the reduce parameter for all models is computed via

$$p_W = \sum_C p'_W(C,M)w(C,M)$$

with $p'_W(C,M)$ the parameter computed for the study on C cores and with REXI parameter M.

The time-to-solution are plotted for the measured timings and the timings of our model in top image in Fig. 14. We can observe a very good match of our model with the real measured timings. This model also allows to get insight in the behaviour of the time-to-solution over a variety of

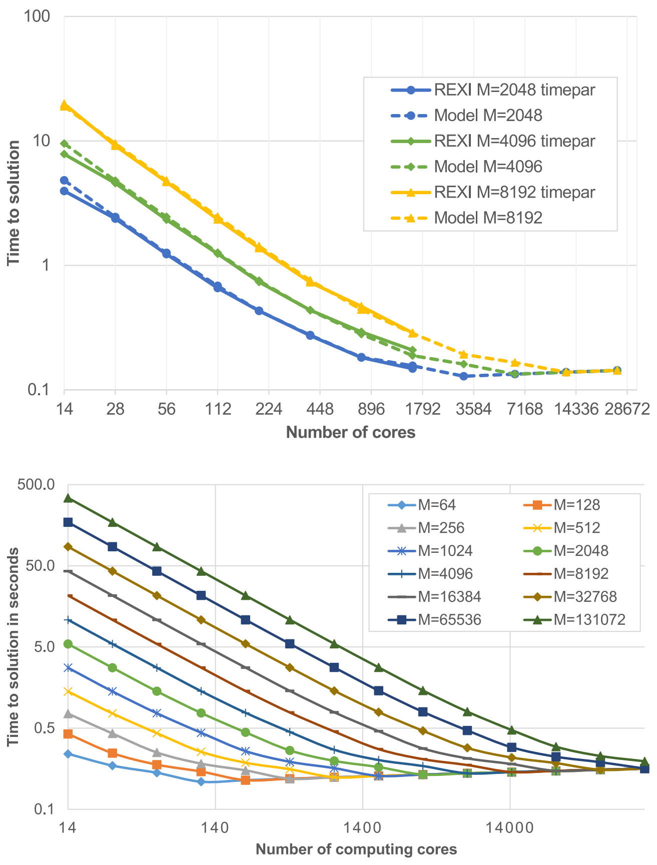


Figure 14. Top: Comparison of the performance results and the suggested model for the REXI parameters used in the finite-difference simulation with a resolution of 128^2. We see a close matching of the performance model with the wall-clock time of our simulations. Bottom: Application of the performance model to generic parameters. We see an excellent scalability for large REXI parameters M.

REXI parameter M and a significantly increased number of cores. Such studies are given in bottom image in Fig. 14. First, we can observe a very good scalability in case of very large REXI M. Second, we see a scalability limitation given at the lowest extrema. This lowest extrema always matches also the hard scalability limitation of REXI given by $C \geq M + L + 1$. At this extremum, adding more cores would not result in a reduce of time-to-solution. We can also study the behaviour for large-scale system with this model in Fig. 14 and the scalability limitations are clearly observable by the local extrema. Furthermore, these plots indicate a very good scalability on already existing systems with 100k cores. We can also see that communication required for the broadcast/reduce operation is still not the limiting part for 100k cores.

7 Summary and discussion

The parallel scalability of standard time stepping methods for oscillatory problems is limited due to the inherently sequential nature of the time stepping algorithm. This is particularly severe in the strong scaling limit, where parallelization in space only does not lead to further reductions in the run-time. In this paper we overcome this issue by developing an inherently parallel time stepping mechanism which is based on a rational approximation of an exponential integrator (REXI). Each of these problems can then be solved independently on a parallel machine. Introducing this additional level of parallelism allows the system to be solved on massively parallel machines.

Two new parameters, h and M, are introduced with REXI. These are related to the accuracy and the novel degree of time-parallelism of the approach, respectively. We first conducted several numerical studies about h and M and their relation to several simulation parameters such as the resolution, waves in the initial condition, linear stiffness, etc. These results allow to improve the understanding of this new parallelization and we like to briefly review one of the results: An increase of the stiffness of the simulation would traditionally lead to an increase in the number of time steps, hence increase in wall-clock time. In contrast, the REXI approach results in additional parallelism, hence has the potential to overcome the strong scalability limitations by allowing to use more computing resources.

To show the feasibility of this concept on HPC systems, a variety of parallel performance benchmarks for representative simulations were conducted. For the parallel-in-time comparisons, the results clearly show a significant reduction in the time-to-solution of $118.3\times$ for spectral methods and $1503.0\times$ for finite-difference methods. For the spectral methods, we were also able to gain results of higher accuracy with a time-to-solution reduced by one order of magnitude. Since the degree of parallelism is also based on the time step size, we conducted time-to-solution studies with robust performance improvements for varying time step sizes compared to non-parallel-in-time studies.

With this new degree of parallelization, a model was developed to improve the understanding of performance limiting factors. This performance model shows a close match with the measured data. Furthermore, we can foresee a potential scalability also on 100k core systems for the considered problems with an oscillatory stiffness which would be otherwise strong-scaling limited by their low resolution of 128×128 on a single compute node.

8 Outlook and future work

In our work we exploited the regular grid structure and used a Fast Helmholtz solver for efficiency reasons. However, this solver is not always applicable, for example, when unstructured grids are adopted. The strategy presented in this paper is general enough so that any preferred solution method for the linear solver could be adopted. In case of locally refined grids, preconditioned multigrid solvers (geometric or algebraic) are an interesting possibility. Multigrid solvers have low memory requirements, but may require several iterations if high accuracy is needed. Direct solvers, such as suggested in (Martinsson 2013), may require more data storage. Overall, the matter of which solver to be used in the linear spatial problem still requires further investigation, as it depends greatly on the specific discrete domain to the used.

The extension of the method to nonlinear hyperbolic equations is not necessarily straightforward. We are investigating two strategies. One is to apply REXI as part of aPinT (see Haut and Wingate (2014)), which is a combination of the Parareal (Lions et al. (2001)) scheme which assembles a coarse propagator, which could be based on REXI for the linear parts, and an averaging over the non-linearities, that allows greater effectiveness of the parallel-in-time method. The second strategy is a semi-Lagrangian method which treats advective non-linear parts in a Lagrangian sense.

We see parallelization-in-time, and methods such as REXI, as new ways to exploit modern architectures. For example, with the increasing sizes of vector registers, getting a vectorization in space is more and more challenging. With parallel-in-time approaches, each vector element can be used to run computations for each independently treated problem in time, hence resulting in a vectorization in the time domain rather than in space as it is usually the case.

Acknowledgements

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC at Leibniz Supercomputing Centre (LRZ, www.lrz.de). We also acknowledge use of Hartree Centre resources in this work on which the early evaluation of the parallelization concepts were done.

We like to thank Eike Mueller, Daniel Ruprecht and the two anonymous reviewers for their valuable feedback.

Funding

Peixoto acknowledges funding from the São Paulo Research Foundation, FAPESP, under grant 2014/10750-0 and the Brazilian National Council for Scientific and Technological Development (CNPq) under grant 441328/2014-8.

A Symbols for mathematical formulation

Symbol	Description
$U(t)$	Solution at time t
$\mathcal{L}$	Linear operator acting on U
I	Identity matrix

B Symbols for shallow water equations

η	Perturbation of surface height
$\bar{\eta}$	Mean surface height constant
f	Coriolis frequency
g	Gravity acceleration constant
(u, v)	Velocity components
δ	Divergence
ζ	Vorticity

C Abbreviations for REXI

Symbol	Description
h	Specifies the sampling density in the approximation of $\exp(ix)$ with a Gaussian basis function
L	Number of rational terms to approximate Gaussian basis function
M	Number of Gaussian basis functions used in the approximation of $\exp(ix)$
$X \times Y$	Spatial resolution of simulation domain
τ, dt	Time step size for REXI
DT	Overall simulation time
(ω^x, ω^y)	Parameters related to the wave-numbers of the initial conditions

References

Arakawa A and Lamb VR (1977) Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in computational physics* 17: 173–265.

Barros S, Dent D, Isaksen L, Robinson G, Mozdzynski G and Wollenweber F (1995) The IFS model: A parallel production weather code. *Parallel Computing* 21(10): 1621 – 1638. Climate and weather modeling.

Berry L, Elwasif W, Reynolds-Barredo J, Samaddar D, Sanchez R and Newman D (2012) Event-based Parareal: A data-flow based implementation of Parareal. *Journal of Computational Physics* 231(17): 5945–5954.

Bonaventura L (2015) Local Exponential Methods: a domain decomposition approach to exponential time integration of PDEs. *CoRR* abs/1505.02248. URL http://arxiv.org/abs/1505.02248.

Clancy C and Pudykiewicz JA (2013) On the use of exponential time integration methods in atmospheric models. *Tellus A* 65: 1–16.

Cox S and Matthews P (2002) Exponential Time Differencing for Stiff Systems. *Journal of Computational Physics* 176(2): 430 – 455.

Dennard RH, Rideout V, Bassous E and Leblanc A (1974) Design of ion-implanted MOSFET's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of* 9(5): 256–268.

Embid PF and Majda AJ (1996) Averaging over fast gravity waves for geophysical flows with arbitrary potential vorticity. *Communications in Partial Differential Equations* 21(3–4): 619–658.

Frigo M (1999) A Fast Fourier Transform Compiler. *SIGPLAN Not.* 34(5): 169–180.

Gander MJ (2015) 50 Years of Time Parallel Time Integration. In: Carraro T, Geiger M, Korkel S and Rannacher R (eds.) *Multiple Shooting and Time Domain Decomposition*. Springer-Verlag.

Gander MJ and Guettel S (2013) PARAEXP: A parallel integrator for linear initial-value problems. *SIAM Journal on Scientific Computing* 35(2): C123–C142.

Garcia F, Bonaventura L, Net M and Sánchez J (2014) Exponential versus IMEX high-order time integrators for thermal convection in rotating spherical shells. *Journal of Computational Physics* 264: 41–54.

Güttel S (2013) Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. *GAMM-Mitteilungen* 36(1): 8–31.

Haut T, Babb T, Martinsson P and Wingate B (2015) A high-order time-parallel scheme for solving wave propagation problems via the direct construction of an approximate time-evolution operator. *IMA Journal of Numerical Analysis* 36: 688–716.

Haut TS and Wingate BA (2014) As asymptotic parallel-in-time method for highly oscillatory PDEs. *SIAM Journal on Scientific Computing* 36(2): A693–A713.

Higham NJ (2008) *Functions of matrices: theory and computation*. Siam.

Hochbruck M and Ostermann A (2010) Exponential integrators. *Acta Numer* 19: 209–286.

Lions JL, Maday Y and Turinici G (2001) Résolution d'EDP par un schéma en temps pararéel. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* 332(7): 661 – 668.

Madec G (2014) NEMO ocean engine (Draft edition r6039). *Note du Pole de modelisation. Institut Pierre-Simon Laplace (IPSL)* 27: ISSN No 1288–1619.

Martinsson PG (2013) A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method. *Journal of Computational Physics* 242: 460–479.

Moler C and Van Loan C (2003) Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review* 45(1): 3–49.

Moore GE (2006) Cramming more components onto integrated circuits Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff. *IEEE Solid-State Circuits Newsletter* 3(20): 33–35.

Mozdzynski G, Hamrud M and Wedi N (2015) A Partitioned Global Address Space implementation of the European Centre for Medium Range Weather Forecasts Integrated Forecasting System. *International Journal of High Performance Computing Applications* 29(3): 261–273.

Randall DA (1994) Geostrophic adjustment and the finite-difference shallow-water equations. *Monthly Weather Review* 122(6): 1371–1377.

Samaddar D, Newman D and Sanchez R (2010) Parallelization in time of numerical simulations of fully developed plasma turbulence using the Parareal algorithm. *Journal of Computational Physics* 229(18): 6558–6573.

Schreiber M et al. (2016) URL https://github.com/schreiberx/sweet.

Speck R, Ruprecht D, Krause R, Emmett M, Minion M, Winkel M and Gibbon P (2012) A massively space-time parallel N-body solver. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, p. 92.

Swarztrauber PN and Sweet RA (1996) The Fourier and cyclic reduction methods for solving Poisson's equation.

Wood N, Staniforth A, White A, Allen T, Diamantakis M, Gross M, Melvin T, Smith C, Vosper S, Zerroukat M and Thuburn J (2014) An inherently mass-conserving semi-implicit semi-Lagrangian discretization of the deep-atmosphere global non-hydrostatic equations. *Quarterly Journal of the Royal Meteorological Society* 140(682): 1505–1520.