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

International Journal of High Performance Computing Applications
XX(X):1–16

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Martin Schreiber[1], Pedro S. Peixoto[12], 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 independent 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 a 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 validated 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 co-design process concerned with 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 anymore. To achieve further scalability of performance 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).

[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

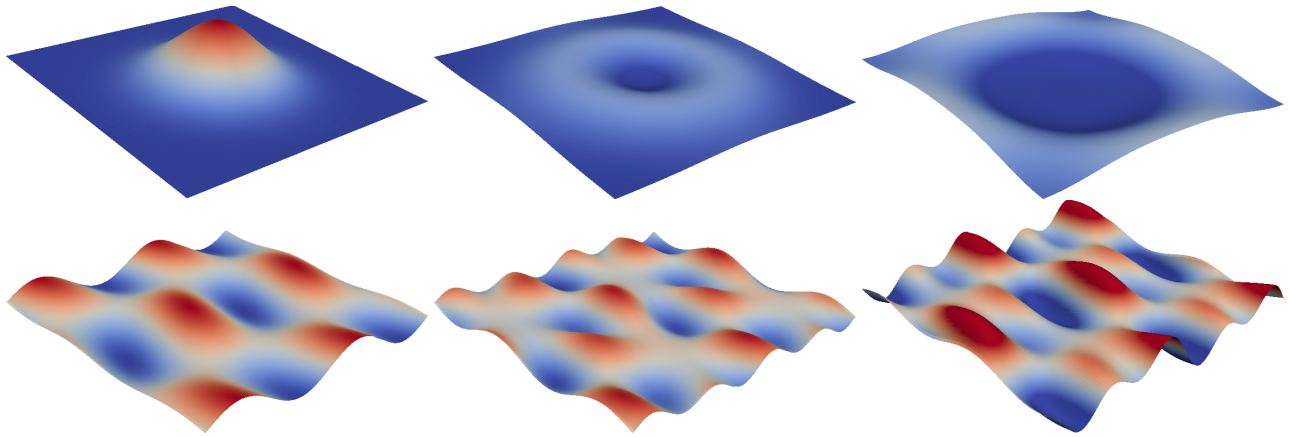

Figure 1. Examples of shallow-water simulations. Top row: Gaussian initial condition (top row) at time $t = 0.0, 0.2, 0.4$. Bottom row: Wave-like initial conditions (bottom row) at time $t = 0.0, 0.1, 0.2$. We solve these equations with a parallel-in-time approach which avoids time step restrictions.

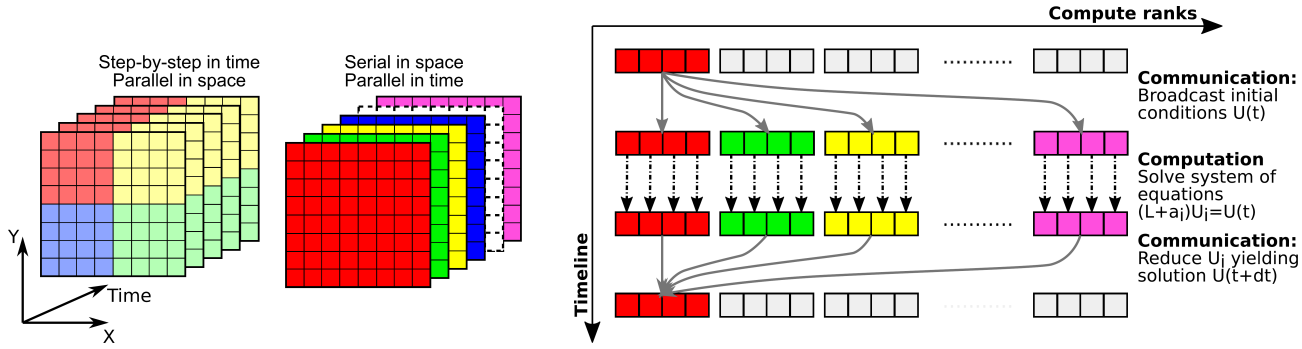

Figure 2. Left: Parallelization-in-space only, each color denotes one compute unit and each slice in time typically requires additional synchronization for the time stepping.
Center: Parallelization-in-time only, each time slice is assigned to a differently colored group of compute unit. Overheads in 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.

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 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, were the dominant method of achieving parallel speed-ups has been to use parallelization-in-space by using domain decomposition 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

It has been proposed that parallelism should be introduced into the time dimension—see Gander (2015) for a review—and this approach has had some successes (Samaddar et al. 2010; Berry et al. 2012). In this work, we focus on evaluating the time evolution operator $\exp(\tau\mathcal{L})$, where it is applied to problems meant for extreme parallelization. This operator is used for the solution to linear systems of equations, but is also a key element of exponential integrators (Hochbruck and Ostermann 2010; Cox and Matthews 2002) and has also been studied in the climate modelling community (Clancy and Pudykiewicz 2013; Garcia et al. 2014). Two important applications that are interesting for use on exascale computer architectures one of the Parareal methods for oscillatory stiffness (Haut and Wingate 2014) and the ParaExp method (Gander and Guettel 2013).

This operator has been studied for many years and there are numerous solution strategies (Moler and Van Loan 2003), but we are primarily concerned with if and how we can achieve time-parallel speed-ups for this operator and therefore focus on one recently proposed rational approximation (Haut et al. 2015).

1.3 Overview

For sake of clarity, in Section 2 we review in detail the implementation of the rational function approach to computing the matrix exponential integration (REXI). Those mainly interested in the performance may skip this section. We evaluate REXI with various different execution scenarios, different discretization schemes and

show its competitiveness. This is done by various benchmark scenarios which are representative for basic building blocks of climate and weather simulations in Section 3. We present our first results on the numerical accuracy of the method in Section 4. These results are important for understanding 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, are an evaluation of REXI using different parallelization aspects (space and/or time). Finally, we gain insight on the behaviour of REXI on large-scale system based on a performance model 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 in REXI with ODEs

To understand how parallel speed-ups can be attained beyond the strong-scaling limit it is important to review one of the fundamental issues with the application. First, consider the ordinary differential equation,

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

with exact solution

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

One standard way to approximate this numerically is by using first order Euler,

$$u(t+\Delta t) = (1-\lambda\Delta t)\, u(t) \quad \text{or} \tag{3}$$

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

This procedure is inherently serial and for real values of λ the time step is limited in size by $\Delta t < 1/\lambda$, for numerical stability. For oscillatory problems (where λ is purely imaginary) we may replace this inherently serial operation by a sum of rational functions. This transfers the serial nature of the problem to one that has more parallelism. To see this replace the term in the braces in Eq. (4) by,

$$(1-\lambda\Delta t)^n \approx \sum_{k=-N}^{N} Re\left(\frac{\beta_k}{\lambda\Delta t + \alpha_k}\right). \tag{5}$$

The sum in Eq. (5) can be distributed on different processors and can take longer time steps in contrast to Eq. (4). This represents a fundamental transformation from an inherently sequential step-by-step in time process to one that can also be parallelized. Rather than taking many small time steps to find the solution at time T, 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 highly parallizability is one of the main issues we address in this paper.

2.2 REXI with systems of ODEs and PDEs

Let the problem to be solved to be of the form

$$U_t = \mathcal{L}U, \tag{6}$$

with $\mathcal{L}$ a linear operator in space. The subscript t refers to the time derivative $\frac{\partial}{\partial t}$, and U is a vector of unknowns. We assume that the linear operator has been discretized in space, such as using the finite difference or Fourier method, so that the equation represents an ODE in time only. The solution in terms of exponential integrators may be written as

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

The goal of this section is to describe how to compute an approximation of $U(t)$ of the form

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

where γ_n and α_n are parameters to be specified. 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 simpler rational approximation procedure for the one dimensional case of e^{ix} (see Sec. 2.3), whose rational approximation is,

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

with complex coefficients α_n and β_n where the range of the sum is $n \in \{-N, \ldots, N\}$. Then we will extend these concepts to the general multi-dimensional case $e^{\tau\mathcal{L}}$ (see Sec. 2.4) followed by using symmetry effects which halve the range of the sum to $n \in \{0, \ldots, N\}$ (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 lays between $x - Mh$ and $x + Mh$, this approximation has been shown to be adequate for $|x| \lesssim Mh$.

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

$$\frac{\hat{f}(\xi)}{\hat{\psi}_h(\xi)} = \sum_{m=\infty}^{\infty} b_m e^{2\pi i m h \xi}, \tag{12}$$

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,

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

for m integer and $1/h$ defining the periodicity of the trigonometric basis function. The parameter h is then chosen to be small enough 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

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

where Re indicates the real part of the argument, and μ and a_l are constant given in Haut et al. (2015), Table 1. The number of coefficients (L) depend 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

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

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

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

and

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

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

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

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 for the oscillatory problem $\mathcal{L}$ is skew-Hermitian and therefore 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

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

where we used

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

where we have explicitly used the fact that the eigenvalues are imaginary, and 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 $x = \tau\lambda_n$, only now it is a matrix.

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

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

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

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

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 may be seen in Haut et al. (2015). Later, in section 4, we will numerically explore optimal choices for these parameters.

The coefficients α_n and β_n are skew-symmetric about the central pole, such that $\alpha_{-n} = \bar{\alpha}_n$ and $\beta_{-n} = \bar{\beta}_n$, and $Im(\alpha_0) = Im(\beta_0) = 0$. Furthermore, it holds that $\overline{(\mathcal{L} + \alpha)^{-1}U} = (\mathcal{L} + \bar{\alpha})^{-1}U$ 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

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

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

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

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. Otherwise, real eigenvalues due to the numerical procedure may produce an unstable scheme. There are strategies to deal with numerical methods that could potentially produce spurious real eigenvalues, for example Haut *et al* suggest a filter that may be applied as a rational approximation as well, which contributes to the overall parallelism in time possible with this scheme. 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 parallization 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 play an important role in determining the oscillatory behaviour (and time step limits) of the problem.

3.1 Problem description

We will solve the linear shallow water equations, linearised with respect to a rest state with mean water depth of $\bar{\eta}$, and defined for perturbations of height η. The linear operator $\mathcal{L}$ may be written as

$$\mathcal{L}U = \begin{pmatrix} 0 & -\bar{\eta}\partial_x & -\bar{\eta}\partial_y \\ -g\partial_x & 0 & f \\ -g\partial_y & -f & 0 \end{pmatrix}, \tag{25}$$

and we wish to solve

$$U_t = \mathcal{L}U, \tag{26}$$

where

$$U = (\eta, u, v)^T. \tag{27}$$

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

$$(\mathcal{L} - \alpha I)U = U_0, \tag{28}$$

are obtained in the following way.

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

$$A_\alpha V = V_0 - g\nabla\eta \tag{29}$$

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

$$A_\alpha = \begin{pmatrix} \alpha & -f \\ f & \alpha \end{pmatrix}. \tag{30}$$

The solution is

$$V = A_\alpha^{-1}(V_0 - g\nabla\eta). \tag{31}$$

Further reformulations with the assumption of a constant f lead to

$$\Delta\eta - \kappa^2\eta \quad = \quad r_0 \tag{32}$$

where

$$\kappa^2 = \frac{f^2 + \alpha^2}{\bar{\eta} g}, \tag{33}$$

$$r_0 = -\frac{\kappa^2}{\alpha}\eta_0 + \frac{1}{g}\delta_0 + \frac{f}{\alpha g}\zeta_0, \tag{34}$$

$\delta = u_x + v_y$ 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. (28), we first solve the associated Helmholtz problem for η (from equation (32)), and then calculate the directly velocities from equation (31).

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 have 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 SWE and avoiding computational spurious modes. On the other hand, collocated grids (A-grids) allow the use of efficient FFT solvers.

Solver for $(\mathcal{L} - \alpha)^{-1}$*:* For the REXI timestepping scheme, a linear system solver is required. More precisely, we need to solve a Helmholtz problem, given by equation 32. 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 general solvers is required (e.g. with unstructured grids).

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),

ID	Space discretization	Grid	Time integration	Solver for $(\mathcal{L}-\alpha)^{-1}$
(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.

$$\eta(x,y) = \sin(2\pi x\omega^x)\cos(2\pi y\omega^y) - \frac{1}{5}\cos(2\pi x\omega^x)\sin(4\pi y\omega^y) \quad (35)$$

$$u(x,y) = \cos(4\pi x\omega^x)\cos(2\pi y\omega^y) \quad (36)$$

$$v(x,y) = \cos(2\pi x\omega^x)\cos(4\pi y\omega^y) \quad (37)$$

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.

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

$$\eta(x,y) = e^{-50(x^2+y^2)}, \quad (38)$$

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, therefore allows the investigation of the dependency of the REXI scheme on the solver used, even when the spectral solver is used.

3.4 Error analysis

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

$$E_{rms}(f) = \sqrt{\frac{\sum_{ij}\left(f_{ij}-\tilde{f}_{ij}\right)^2}{N_x N_y}}, \quad (39)$$

which is based on the discrete computed solution f_{ij} for the points given by (x_i,y_j) and a 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

$$E_{max}(f) = \max_{ij}|f_{ij}-\tilde{f}_{ij}|. \quad (40)$$

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 understating 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, and we will briefly recall their meaning.

The parameter h defines the sampling of the $\exp(ix)$ with respect to the a sum a Gaussian functions. The parameter M sets the number of terms which are involved in the approximation. Together, they define the interval for adequate approximation of the relevant frequencies of the system. Recalling Sec. 2, adequate solutions are obtained if $\tau\bar{\lambda} < hM$, where τ is the time step size and $\bar{\lambda}$ is the maximum absolute eigenvalue of the linear operator. Therefore, smaller h would require larger M and vice-versa.

The parameter L is related to the number of terms for the approximation of the Gaussian function with a rational function. With $L=11$, sufficient accuracy (within single precision accuracy) is obtained. Since this will not be the leading error in the analysis, increasing L would not increase the accuracy of the simulations. Also, larger values for this term would not significantly change the computational workload, since $M \gg L$.

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 expect, smaller values of h require larger values of M to preserve accuracy, which reveal 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

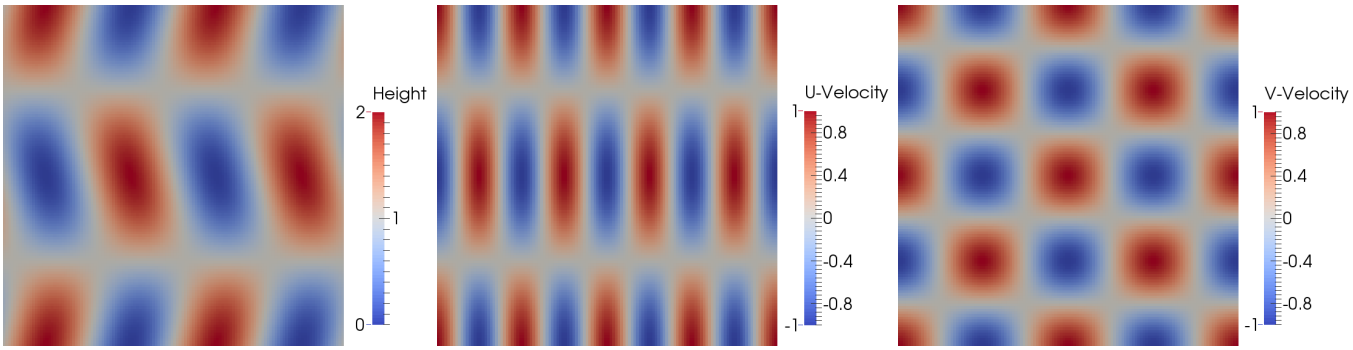

Figure 3. Initial conditions for the wave scenario given by parameters $(\omega^x, \omega^y) = (2, 1)$, see Sec. 3.3. The variables shown are the height perturbation (η), velocity in x-direction (u), velocity in y-direction (v) from left to right.

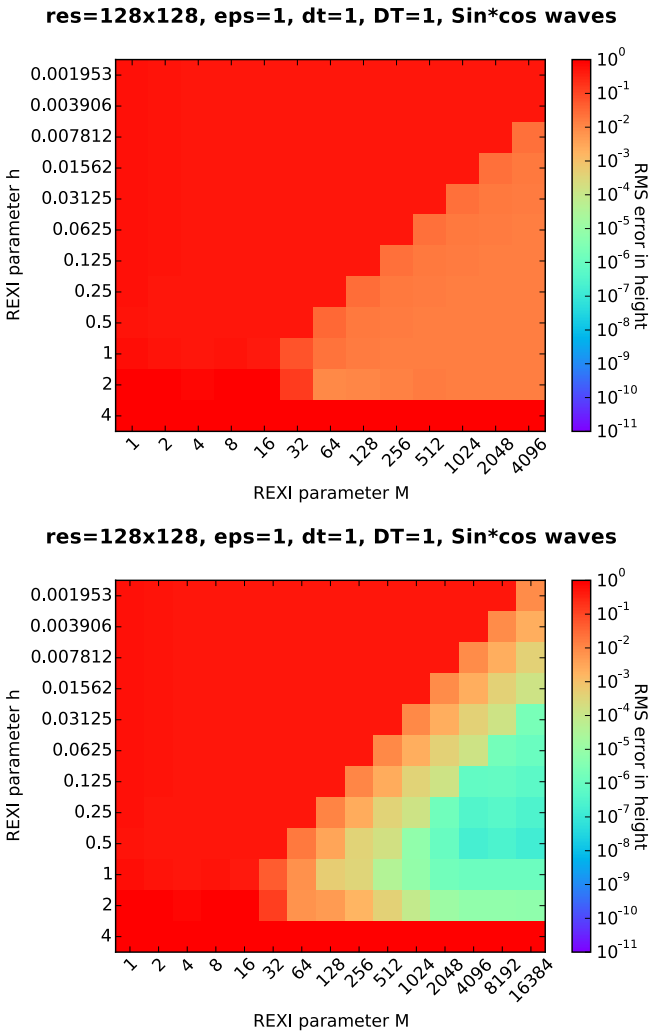

Figure 4. Error analysis (RMS) of the REXI scheme with respect to varying h and M using two methods. Top image shows method (B) which uses a finite-difference solver. Bottom image shows method (D) which uses spectral solver. The errors are those of a single large time step of size, $\tau = 1$, for the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

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 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 in 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.

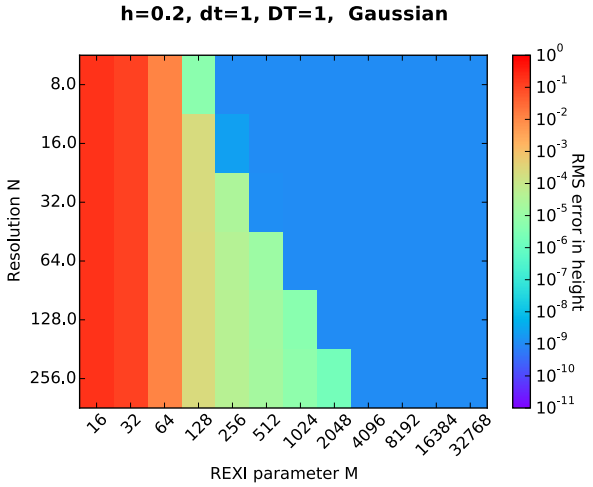

Figure 5. Error analysis (RMS) of the REXI scheme with respect to varying resolution N and M using method (D)-spectral solver. The errors are those of a single large time step of size $\tau = 1$, for the Gaussian initial conditions scenario. We can observe that M linearly depends on the resolution, so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

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,

of this work, which is large enough to reduce the workload, but small enough to produce accurate results given $M > 32$.

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

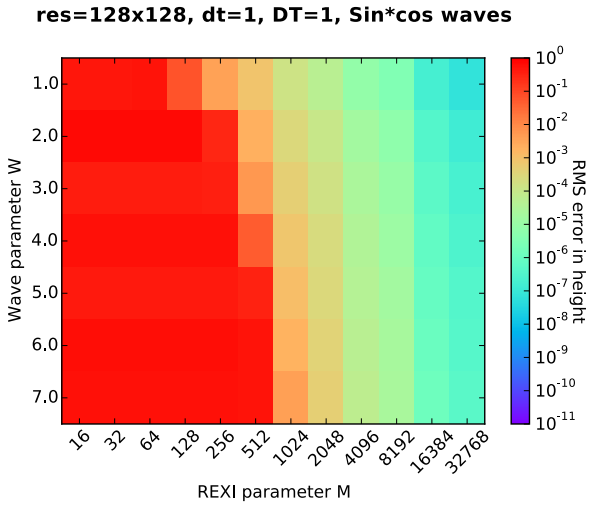

Figure 6. Error analysis (RMS) of the REXI scheme with respect to varying wave-numbers of the initial condition (ω using method (D)-spectral solver. The errors are those of a single large time step of size $\tau = 1$, for the waves initial conditions scenario. We can observe that M linearly depends on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

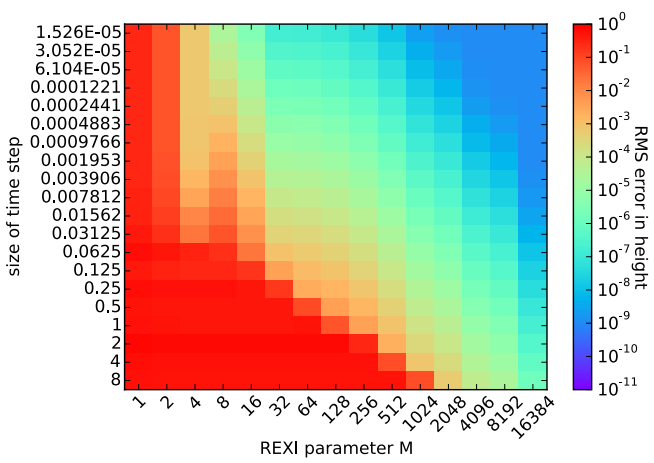

Figure 7. Dependency of REXI parameter M on varying time step size. To compute larger time steps, also M has to be increased.

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. This propagation speed also depend 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 L into a system of independent equations for each spatial frequency. Furthermore, we assume that an accurate solver is used for solving the inverse problem in each REXI term. The linear operator 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

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

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

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

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 $\sigma_{\pm 1} = \pm\sqrt{4\pi^2(\bar{\eta} g k_1^2 + \bar{\eta} g k_2^2) + f^2}$. 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.

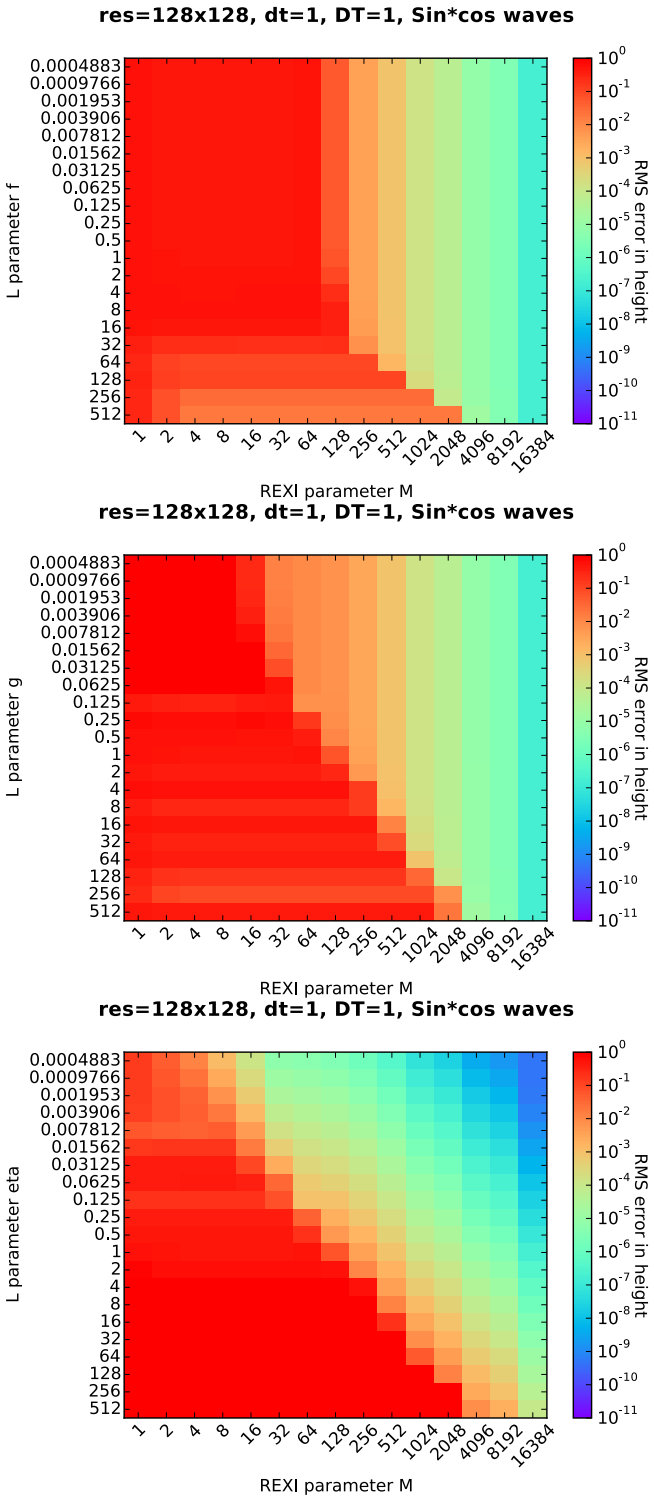

Figure 8. Error sensitivity study with varying parameters f (Coriolis frequency), g (gravitational constant), and $\bar{\eta}$ (average height) and parameter M on varying frequency ϵ of linear operator (larger ϵ leads to more oscillations) with resolution 128^2. The varying frequency requires increasing 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.

5 HPC performance evaluation

The current trends in high-performance computing goes 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≈ 0.3, this resulted in time step sizes of about 0.0017 seconds with RK4.

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.

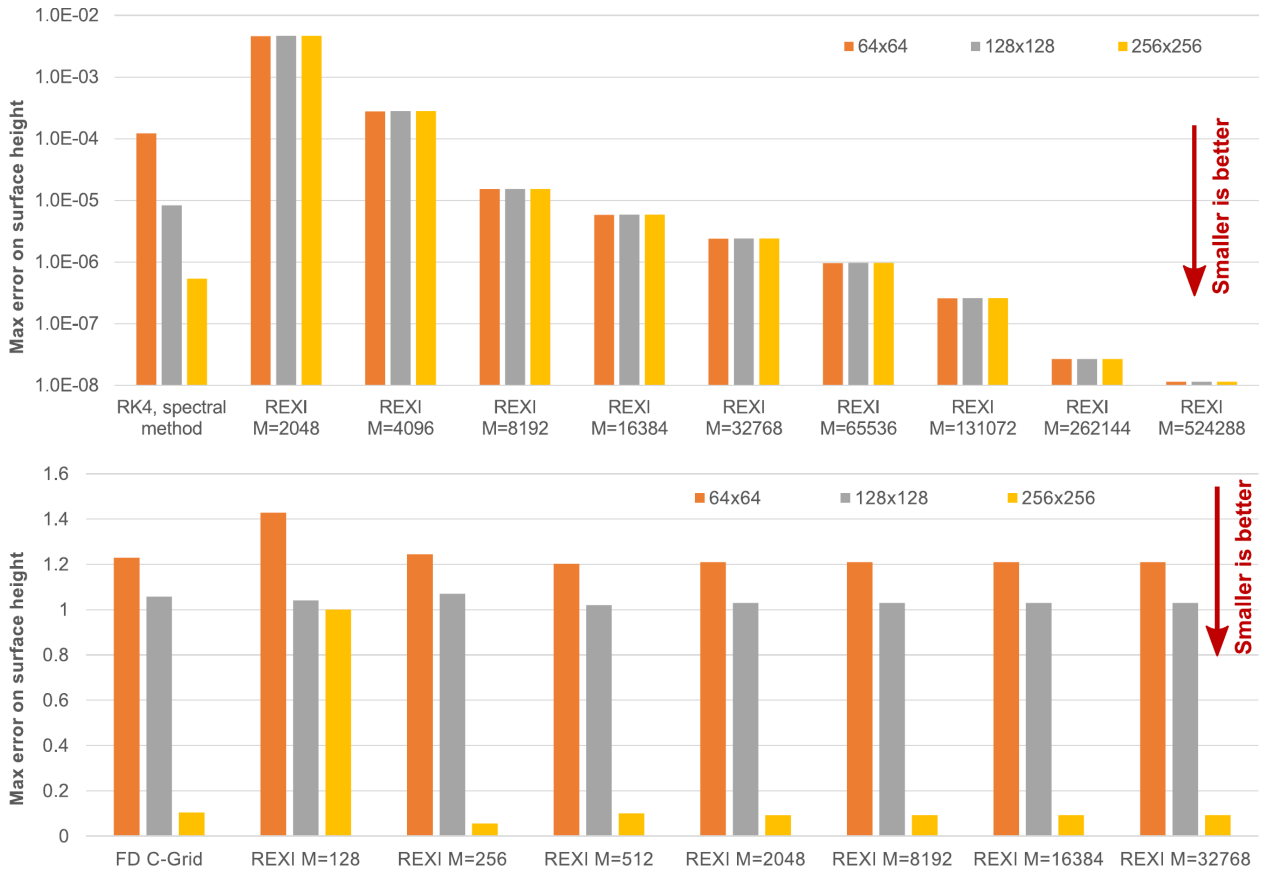

Figure 9. Maximum error norm computed on surface height for different REXI parameters M. Results are plotted for spectral methods (bottom) and finite-difference (top). The errors are identical for different resolutions since the initial condition can be error-free represented in spectral space.

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 and each rank. For an improved comparison 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.

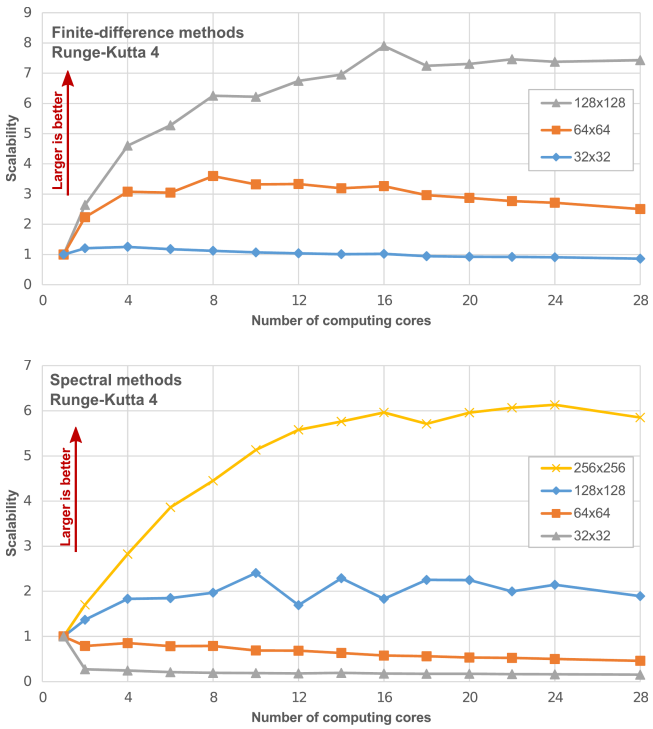

Figure 10. Example showing a strong scalability limitation with fixed number of RK4 time steps for a standard parallelization-in-space approach. Top: Speedup for finite-differences (staggered) C-grid (method id (A)), Bottom: Speedup for spectral method on A-grid (method id (B))

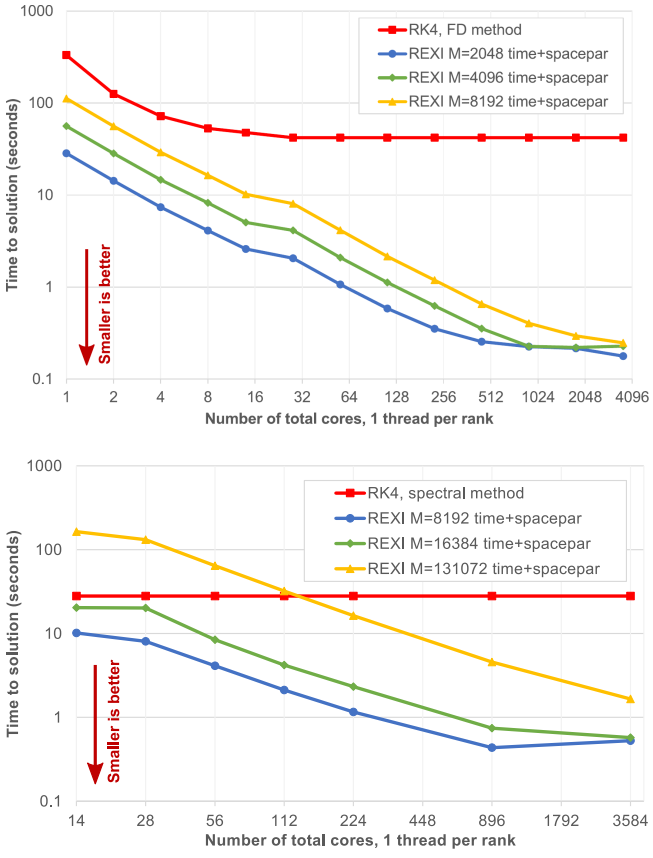

Figure 11. Scalability plot with only a parallelization-in-time. The grid resolution is 128×128. Time-to-solution (seconds) is plotted for finite-difference (top) and spectral methods (bottom).

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.
- "*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, independent 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

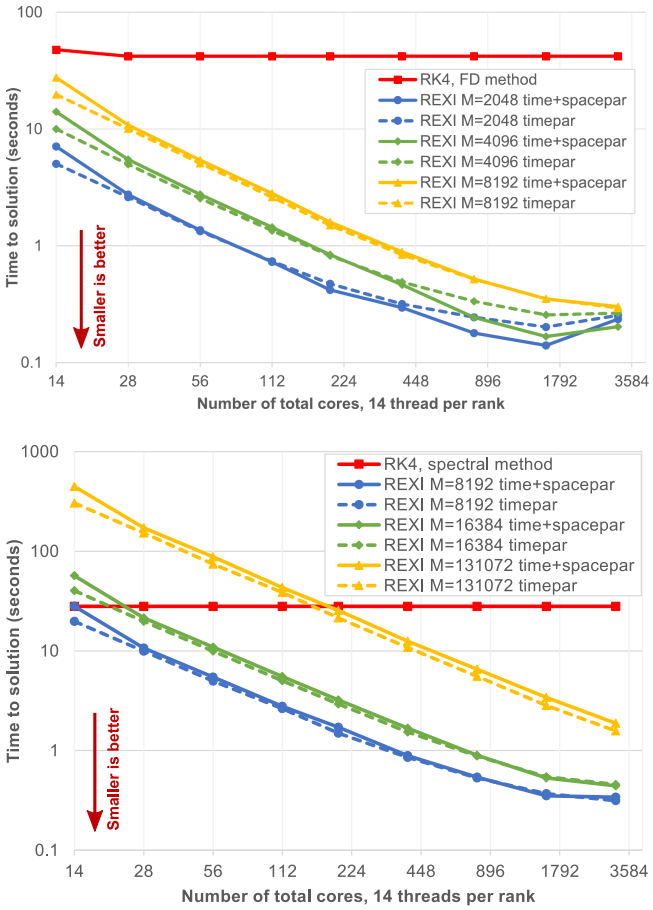

Figure 12. Scalability with a hybrid parallelization on simulation on a grid with a 128×128 resolution. *REXI M=... timepar* denotes a parallelization in space (OpenMP) and parallelization in time (MPI). *REXI M=... time+spacepar* denotes a parallelization in time only (MPI+OpenMP). Results are plotted for finite-difference (top) and spectral methods (bottom). For improved visualization, the time for the RK4 method is omitted for the finite difference method.

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.

Spectral methods: We observe that for the spectral method the hybrid parallelization-in-space-and-time in 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.

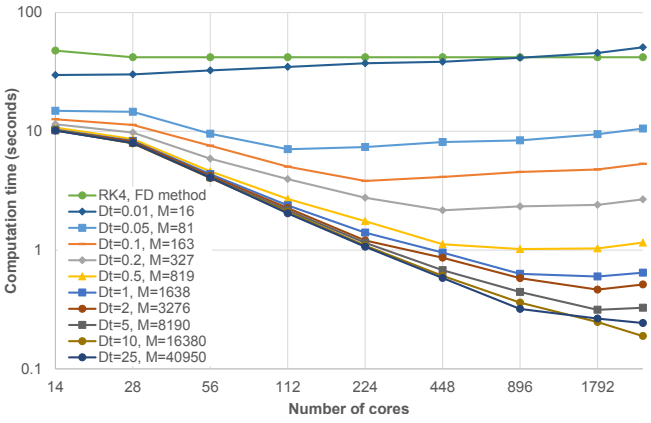

Figure 13. Time-to-solution for simulation over 50 seconds and different REXI time step sizes. Parallelization is done in time only.

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. 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

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

An overview of all involved components in the model is given in table in Table 2 and assume a constant problem size (resolution) for each problem.

Broadcast s_B: To spread the initial conditions $U(t_0)$, a broadcast is used. Regarding the parallel communication

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. $T(C)$ denotes the runtime for each specific part. The last column contains the determined model parameters s/p as discussed in this Section.

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 parallelizable part in toto. However, we can precompute several components with the reformulation to the Helmholtz problem (Eq. 32). 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 to 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

$$\begin{aligned} T(C,M) &= \underbrace{s_L}_{\text{serial part}} \\ &+ \underbrace{s_B \log(C) + s_R \log(C)}_{\text{serial/parallel part}} \\ &+ \underbrace{p_W \left\lceil \frac{W}{\min(C,W)} \right\rceil}_{\text{parallel part}} \end{aligned} \tag{41}$$

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

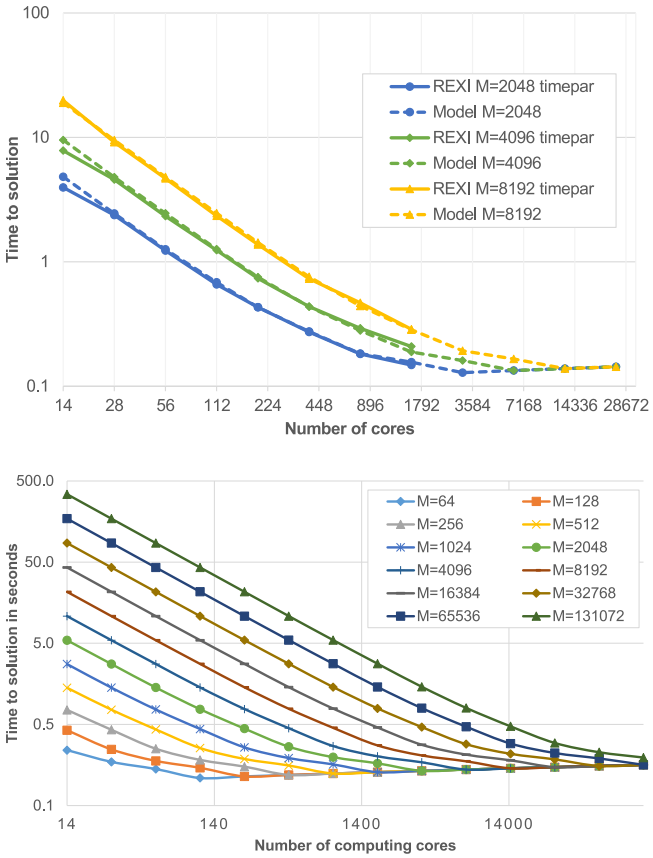

Figure 14. Top: Comparison of the performance results and the suggested model for the REXI parameters used in the finite-difference simulation with a resolution of 128^2. We see a close matching of the performance model with the wall-clock time of our simulations. Bottom: Application of the performance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

in the behaviour of the time-to-solution over a variety of 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

A new way of parallelism was discussed for oscillatory problems which are limited in their wall-clock time by their strong scalability and a step-by-step method in time. Here, a reformulation is required to tackle the curse of a sequential step-by-step approach in time and we used a rational approximation of an exponential integrator (REXI). This REXI approach allows to solve the same problem with a sum over independent subset problems. Each of these problems can then be solved independent to each other, hence paving a way towards massive parallel systems.

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.

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. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure.

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
$\mathcal{L}U$	Linear operator acting on U

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.

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.

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* : drv021.

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.

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

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

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.

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.