SIAM J. SCI. COMPUT.
Vol. 41, No. 1, pp. A220–A245

A DIRECT TIME PARALLEL SOLVER BY DIAGONALIZATION FOR THE WAVE EQUATION*

MARTIN J. GANDER†, LAURENCE HALPERN‡, JOHANN RANNOU§, AND JULIETTE RYAN§

Abstract. With the advent of very large scale parallel computers, it has become more and more important to also use the time direction for parallelization when solving evolution problems. While there are many successful algorithms for diffusive problems, only some of them are also effective for hyperbolic problems. We present here a mathematical analysis of a new method based on the diagonalization of the time stepping matrix proposed by Maday and Rønquist in 2007. Like many time-parallelization methods, at first this does not seem to be a very promising approach: the matrix is essentially triangular, or, for equidistant time steps, actually a Jordan block, and thus not diagonalizable. If one chooses however different time steps, diagonalization is possible, and one has to trade off between the accuracy due to necessarily having different time steps, and numerical errors in the diagonalization process of these almost nondiagonalizable matrices. We present for the first time such a diagonalization technique for the Newmark scheme for solving wave equations, and derive a mathematically rigorous optimization strategy for the choice of the parameters in the special case when the Newmark scheme becomes Crank–Nicolson. Our analysis shows that small to medium scale time parallelization is possible with this approach. We illustrate our results with numerical experiments for model wave equations in various dimensions and also an industrial test case for the elasticity equations with variable coefficients.

Key words. high performance computing, time parallelism, direct solver

AMS subject classifications. 65M55, 65M12

DOI. 10.1137/17M1148347

1. Introduction.

Using the time direction in evolution problems for parallelization is an active field of research; see [13] for an overview. Most of the methods developed for this purpose are iterative; see for example the parareal algorithm [33], whose convergence was analyzed in [26] for linear problems. It was also shown in [26] that for the transport equation no speed-up is possible with the parareal algorithm; for further analyses in the hyperbolic case, see [12]. A sharp convergence estimate of the parareal algorithm for nonlinear problems can be found in [9]. Later, a variation of the parareal algorithm using spectral deferred correction [38] led then to the parallel full approximation scheme in space and time algorithm [5], which is a multilevel method. Space-time multigrid methods were also developed (see [31] and references therein), and a new such method using only standard components can be found in [22] with excellent strong and weak scaling properties for parabolic problems. There is also the noninvasive multigrid reduction in time (MGRIT) algorithm [8, 6], and it was shown in [21] that MGRIT is equivalent to an overlapping parareal algorithm, which led to a detailed nonlinear convergence analysis for MGRIT [21]. Early versions of these space-time multigrid methods were based on waveform relaxation [34, 47], and there are also very successful Schwarz waveform relaxation methods for the time

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 19, 2017; accepted for publication (in revised form) September 17, 2018; published electronically January 8, 2019.

http://www.siam.org/journals/sisc/41-1/M114834.html

†Section de Mathematiques, Geneva University, CH-1211, Geneve, Switzerland (Martin.Gander@unige.ch).

‡LAGA, Universite Paris, Villetaneuse, 93430, France (halpern@math.univ-paris13.fr).

§ONERA, Chatillon, 92322, France (johann.rannou@onera.fr, ryan@onera.fr).

parallel solution of evolution partial differential equations [25, 28, 17, 11, 16, 1], and the more recent Dirichlet–Neumann and Neumann–Neumann waveform relaxation methods [37, 32, 19]. These are among the very few space-time iterative methods that are effective for hyperbolic problems [18, 15, 42, 20], since they use the finite speed of propagation for good space-time decompositions; see also the related tent-pitching approach [30] which removes the need for iteration completely by following in the decomposition the characteristics. A different approach for hyperbolic problems are the Krylov parareal methods [7, 23, 24, 45], which however have an important overhead due to orthogonalization.

As an alternative, one can use direct space-time parallel solvers, like revisionist integral deferred correction (RIDC) [4], which is ideal for multicore architectures with 4 to 8 cores and generates high order solutions in time in parallel at the runtime cost of an Euler time stepper. For linear hyperbolic problems, there is also the ParaExp algorithm [14] which is based on a fully overlapping decomposition and Krylov techniques. A further direct approach based on the diagonalization of the time stepping matrix was introduced in [36]. This approach contains however parameters which must be carefully selected for the method to be successful. A first estimate for these parameters for a backward Euler discretization of the heat equation was obtained in the short manuscript [10]. The idea of diagonalization can already be found in [2, 3] at the level of implicit Runge–Kutta methods, but not with parallelization in mind. Wave equations are however more delicate and require more refined techniques leading to more accurate estimates for the parameters.

We present a new such direct time parallel method based on diagonalization for wave equations discretized by the Newmark scheme. Our analysis for the special case of Crank–Nicolson shows that with this approach not an arbitrarily large number of processors can be used to parallelize in time, only about 3–4 times more than for RIDC, for example, but as for RIDC, this number becomes a multiplicative factor for the processors used for parallelization in space: for example, if 1000 processors were used for parallelization in space, and diagonalization permits the use of 20 processors in time, overall 20000 processors could then be used to solve the wave equation in parallel, instead of only 1000 without parallelization by diagonalization. We consider as our model partial differential equation (PDE) the second order wave equation,

$$\begin{aligned} \partial_{tt} u - \Delta u &= 0 && \text{in } \Omega \times (0,T), \\ u &= 0 && \text{on } \partial\Omega, \\ (u, \partial_t u)(\cdot, 0) &= (f, g) && \text{in } \Omega. \end{aligned} \tag{1.1}$$

A popular implicit time integrator for the wave equation (1.1) is the Newmark scheme [40]: let Δ_h be a space discretization of the Laplacian Δ, and we thus want to discretize $\ddot{u}_h - \Delta_h u_h = 0$ in time, where $\ddot{u}_h := \partial_{tt} u_h$. In the Newmark scheme, the additional unknown $\dot{u}_h := \partial_t u_h$ is introduced, and one then approximates both u_h and $\dot{u}_h$ on the time partition $0 = t_0 < t_1 < t_2 < \cdots < t_N = T$, $k_n := t_n - t_{n-1}$, by a time stepping rule involving the two parameters β and γ,

$$\begin{aligned} u_h^{n+1} &= u_h^n + k_{n+1}\dot{u}_h^n + k_{n+1}^2((\tfrac{1}{2} - \beta)\ddot{u}_h^n + \beta \ddot{u}_h^{n+1}), \\ \dot{u}_h^{n+1} &= \dot{u}_h^n + k_{n+1}((1-\gamma)\ddot{u}_h^n + \gamma \ddot{u}_h^{n+1}), \\ \ddot{u}_h^n - \Delta_h u_h^n &= 0 \end{aligned} \tag{1.2}$$

with given initial data u_h^0 and $\dot{u}_h^0$.

Let I_t be the $N\times N$ identity matrix associated with the time domain and I_x be the $J\times J$ identity matrix associated with the spatial domain. Setting $\mathbf{u} := (u_h^1,\dots,u_h^N)$, and similarly for $\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$, and defining the matrices

$$B_1 = \begin{pmatrix} \frac{1}{k_1} & & & \\ -\frac{1}{k_2} & \frac{1}{k_2} & & 0 \\ 0 & \ddots & \ddots & \\ & & -\frac{1}{k_N} & \frac{1}{k_N} \end{pmatrix}, \qquad B_2 = \begin{pmatrix} 0 & & & \\ 1 & 0 & & 0 \\ 0 & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix},$$

$$C_1 = \begin{pmatrix} \beta k_1 & & & \\ (\frac{1}{2}-\beta)k_2 & \beta k_2 & & 0 \\ 0 & \ddots & \ddots & \\ & & (\frac{1}{2}-\beta)k_N & \beta k_N \end{pmatrix}, \quad C = \begin{pmatrix} \gamma & & & \\ (1-\gamma) & \gamma & & 0 \\ 0 & \ddots & \ddots & \\ & & (1-\gamma) & \gamma \end{pmatrix},$$

the scheme (1.2) can be written in the compact form

$$\begin{aligned} &(B_1\otimes I_x)\mathbf{u} = (B_2\otimes I_x)\dot{\mathbf{u}} + (C_1\otimes I_x)\ddot{\mathbf{u}} + F,\\ &(B_1\otimes I_x)\dot{\mathbf{u}} = (C\otimes I_x)\ddot{\mathbf{u}} + G,\\ &\ddot{\mathbf{u}} - (I_t\otimes \Delta_h)\mathbf{u} = 0 \end{aligned} \tag{1.3}$$

with the right-hand sides

$$F = (\frac{1}{k_1}u_h^0 + \dot{u}_h^0 + (\frac{1}{2}-\beta)k_1\Delta_h u_h^0, 0,\cdots,0), \quad G = (\frac{1}{k_1}\dot{u}_h^0 + (1-\gamma)\Delta_h u_h^0, 0,\cdots,0).$$

Solving the second equation in (1.3) for $\dot{\mathbf{u}}$ using that for the Kronecker product $(B_1\otimes I_x)(B_2\otimes I_x) = (B_1B_2\otimes I_x)$, we get $\dot{\mathbf{u}} = (B_1^{-1}\otimes I_x)((C\otimes I_x)\ddot{\mathbf{u}} + G)$, which we insert into the first equation to obtain

$$(B_1\otimes I_x)\mathbf{u} = ((B_2B_1^{-1}C + C_1)\otimes I_x)\ddot{\mathbf{u}} + (B_2B_1^{-1}\otimes I_x)G + F.$$

The matrix $B_3 := B_2B_1^{-1}C + C_1$ is a lower triangular matrix whose diagonal is that of C_1. Hence B_3 is invertible, and we can solve for $\ddot{\mathbf{u}}$ to obtain

$$\ddot{\mathbf{u}} = (B_3^{-1}\otimes I_x)\Big((B_1\otimes I_x)\mathbf{u} - (B_2B_1^{-1}\otimes I_x)G - F\Big).$$

Inserting this into the last equation in (1.3), we find

$$(B\otimes I_x - I_t\otimes\Delta_h)\mathbf{u} = \mathbf{f} \tag{1.4}$$

with

$$B = B_3^{-1}B_1, \quad \mathbf{f} = (B_3^{-1}\otimes I_x)((B_2B_1^{-1}\otimes I_x)G + F). \tag{1.5}$$

Equation (1.4) is the matrix tensor-product form of the Newmark scheme for the wave equation in mixed form. It is equivalent to (1.2), and since the matrices are well-conditioned, it gives numerically the same solution. The velocity vector can be recovered using the equation

$$\dot{\mathbf{u}} = (B_1^{-1}\otimes I_x)((C\otimes\Delta_h)\mathbf{u} + G).$$

The key idea to solve a space-time problem written in tensor-product form (1.4) in a time parallel fashion from [36] is to diagonalize B. Since all matrices are lower

triangular, B is lower triangular, and the diagonal of B equals $(\frac{1}{\beta k_1^2}, \cdots, \frac{1}{\beta k_N^2})$. If all the time steps are different, then B is indeed diagonalizable, $B = SDS^{-1}$, where the diagonal matrix D contains the eigenvalues $\frac{1}{\beta k_n^2}$ on the diagonal, and (1.4) can be written as

$$(S \otimes I_x)(D \otimes I_x - I_t \otimes \Delta_h)(S^{-1} \otimes I_x)\mathbf{u} = \mathbf{f}. \tag{1.6}$$

One can then solve (1.6) parallel in time performing the following three steps:

$$\begin{array}{lrcl} \text{(a)} & \mathbf{g} & = & (S^{-1} \otimes I_x)\mathbf{f}, \\ \text{(b)} & (\frac{1}{\beta k_n^2} - \Delta_h)\mathbf{w}^n & = & \mathbf{g}^n, \quad 1 \le n \le N, \\ \text{(c)} & \mathbf{u} & = & (S \otimes I_x)\mathbf{w}. \end{array} \tag{1.7}$$

The main work, namely the N linear systems in space in step (b), can all be solved in parallel using N processors, or if the spatial problems were already parallelized using M processors, one could then overall use $N \times M$ processors. We call this approach time parallelization by diagonalization. It was first introduced and analyzed numerically for wave propagation in [43, 44], motivated by the original invention in [35, 36] for diffusive problems the authors called tensor-product space-time solvers. We will analyze here specifically the Crank–Nicolson scheme, which corresponds to $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$ in the Newmark scheme (1.2), and leads to the unconditionally stable time-integration operator of maximum accuracy most often used in practice. To see in what sense the parameter choice $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$ corresponds to Crank–Nicolson, we first replace the last equation in (1.2) into the first two for these values of the parameters,

$$\begin{array}{rcl} u_h^{n+1} & = & u_h^n + k_{n+1}\dot{u}_h^n + \frac{k_{n+1}^2}{4}(\Delta_h u_h^n + \Delta_h u_h^{n+1}), \\ \dot{u}_h^{n+1} & = & \dot{u}_h^n + \frac{k_{n+1}}{2}(\Delta_h u_h^n + \Delta_h u_h^{n+1}). \end{array}$$

Now from the second equation, we have $\frac{1}{4}(\Delta_h u_h^n + \Delta_h u_h^{n+1}) = (\dot{u}_h^{n+1} - \dot{u}_h^n)/(2k_{n+1})$, which when inserted into the right hand side of the first equation gives

$$\begin{array}{rcl} u_h^{n+1} & = & u_h^n + \frac{k_{n+1}}{2}(\dot{u}_h^n + \dot{u}_h^{n+1}), \\ \dot{u}_h^{n+1} & = & \dot{u}_h^n + \frac{k_{n+1}}{2}(\Delta_h u_h^n + \Delta_h u_h^{n+1}), \end{array} \tag{1.8}$$

which is clearly Crank–Nicolson applied to the mixed formulation of the wave equation, which with $v_h := \dot{u}_h$ is given by

$$\begin{array}{rcl} \dot{u}_h & = & v_h, \\ \dot{v}_h & = & \Delta_h u_h. \end{array}$$

Crank–Nicolson is second order accurate in time and unconditionally stable. With a similar computation as for the Newmark scheme, we also obtain the tensor-product form (1.4) for the Crank–Nicolson scheme with

$$B = (C^{-1}B_1)^2, \quad \mathbf{f} = (C^{-1} \otimes I_x)((B_1 C^{-1} \otimes I_x)F + G), \tag{1.9}$$

$\beta = \frac{1}{4}$, $\gamma = \frac{1}{2}$, and the same G, but with $F = (\frac{1}{k_1}u_h^0 + \frac{1}{2}\dot{u}_h^0, 0, \cdots, 0)$.

2. Analysis of time parallelization by diagonalization. Using the time parallel solver (1.7) based on diagonalization requires some care: first, the time stepping matrix B is only diagonalizable if the time steps are all different, and this can lead to a larger discretization error compared to using equidistant time steps. Second, the

condition number of the eigenvector matrix S increases exponentially with the number of time steps N to be done in parallel, which leads to inaccurate results in steps (a) and (c) of (1.7) because of roundoff error. One therefore needs to carefully and accurately estimate these two errors to determine how many time steps can indeed be performed in parallel using (1.7) without loosing accuracy.

In order to obtain such estimates in the wave equation case, we use a Fourier transform in space with Fourier variable ξ and obtain $\frac{\partial^2 \hat{u}}{\partial t^2} + |\xi|^2 \hat{u} = 0$, which indicates that we need to study for fixed T and $a > 0$ the algorithm (1.7) applied to the ordinary differential equation (ODE)

$$\ddot{u} + a^2 u = 0, \quad t \in (0, T) \tag{2.1}$$

with initial conditions $u(0) = f$ and $\dot{u}(0) = g$. The Crank–Nicolson scheme for (2.1) is

$$\begin{aligned} u^n &= u^{n-1} + \tfrac{1}{2} k_n (\dot{u}^n + \dot{u}^{n-1}), \\ \dot{u}^n &= \dot{u}^{n-1} - \tfrac{a^2}{2} k_n (u^n + u^{n-1}) \end{aligned} \tag{2.2}$$

with initial conditions $u^0 = f$ and $\dot{u}^0 = g$. Let $U := (u, \frac{1}{a}\dot{u})$ be the solution of (2.1) including the derivative term which will be important for our estimates, and $U^n := (u^n, \frac{1}{a}\dot{u}^n)$ be the discrete approximation defined by (2.2). We can then rewrite the continuous and discretized problems as

$$d_t U + aJU = 0, \quad \left(I + \frac{ak_n}{2} J\right) U^n = \left(I - \frac{ak_n}{2} J\right) U^{n-1}, \quad J := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \tag{2.3}$$

with initial condition $U(0) = U^0 = (f, \frac{g}{a})$. For a given N, we now consider time steps given by a geometric partition[1] $\mathcal{T}_q := (k_1, \cdots, k_N)$, $k_n(q) := q^{n-1} k_1$ as it was suggested in [35]. The constraint $\sum_{n=1}^N k_n = T$ determines uniquely the value of the geometric time steps,

$$k_n = \frac{q^n}{\sum_{j=1}^N q^j} T. \tag{2.4}$$

We call $U^n(\mathcal{T}_q)$ the corresponding sequence of solutions. The equidistant timestep $\tilde{k} = \frac{T}{N}$ is obtained for $q = 1$. An example of such a geometric time mesh is given in Figure 2.1 for a typical value $q = 1.1$ our analysis will determine and for comparison also an equidistant time mesh is shown.

2.1. Estimation of the error due to the geometric time stepping. We now present an accurate estimate for the change in the truncation error caused by using a geometric time stepping with

$$q = 1 + \varepsilon, \quad \varepsilon \text{ a tunable parameter}, \tag{2.5}$$

compared to using equal time stepping, by studying the distance between $U^N(\mathcal{T}_q)$ and $U^N(\mathcal{T}_1)$.

[1] Nick Higham, after a presentation of the first author, and later also an anonymous referee, suggested considering a random partition. We tested this and did not find any advantage over the geometric partition in our numerical experiments. The geometric partition allows us to provide a complete analysis for the best choice of the parameters in the method, while for the analysis of random time steps it is not clear how one would have to proceed.

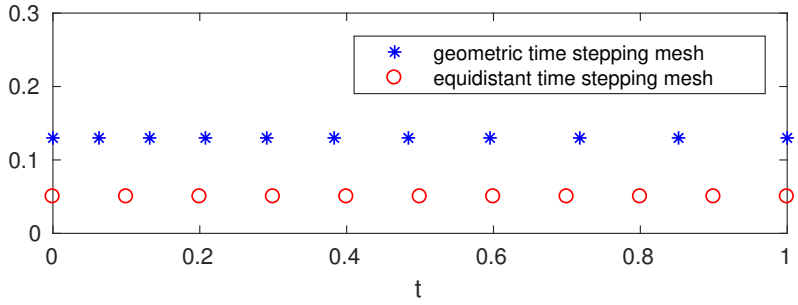

FIG. 2.1. *Geometric time stepping mesh for $q = 1.1$, a typical value of the parameter we will determine with our analysis, compared to an equidistant time stepping mesh.*

THEOREM 2.1 (asymptotic truncation error estimate). *For fixed a, T, and N, we obtain for ε small the error estimate*

$$\|U^N(\mathcal{T}_{1+\varepsilon}) - U^N(\mathcal{T}_1)\| = \phi(\frac{aT}{2N}, N)\varepsilon^2\|U^0\| + \mathcal{O}(\varepsilon^3), \tag{2.6}$$

where $\phi(y, N) := \frac{N(N^2-1)}{6}\frac{y^3}{(1+y^2)^2}$.

Proof. We first diagonalize the two systems (2.3) in the complex plane: let

$$P := \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}, \quad D := \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},$$

which implies that $P^{-1}JP = iD$ and $P^{-1}(I - \alpha J)P = I - i\alpha D$. Defining

$$\mu(t) := e^{-iat}, \quad \lambda_n := \frac{1 - i\frac{ak_n}{2}}{1 + i\frac{ak_n}{2}}, \quad \text{and} \quad \mu_n := \prod_{p=1}^{n} \lambda_p,$$

we obtain the diagonalized continuous and discrete solutions

$$U(t) = P \begin{pmatrix} \bar{\mu}(t) & 0 \\ 0 & \mu(t) \end{pmatrix} P^{-1}U_0, \quad U^n = P \begin{pmatrix} \bar{\mu}_n & 0 \\ 0 & \mu_n \end{pmatrix} P^{-1}U^0.$$

For a given N, we now estimate the change induced by the nonequidistant time steps. For any partition $\mathcal{T} = (k_1, \cdots, k_N)$ of the interval $(0, T)$, we compare the solution $U^N(\mathcal{T})$ to the solution $U^N(\widetilde{\mathcal{T}})$ obtained by the equidistant mesh $\widetilde{\mathcal{T}} = (\tilde{k}, \cdots, \tilde{k})$ with $\tilde{k} = T/N$. Defining

$$\mu(\mathcal{T}) := \prod_{n=1}^{N} \frac{1 - i\frac{ak_n}{2}}{1 + i\frac{ak_n}{2}},$$

we find that

$$U^N(\mathcal{T}) - U^N(\widetilde{\mathcal{T}}) = P \begin{pmatrix} \overline{\mu(\mathcal{T}) - \mu(\widetilde{\mathcal{T}})} & 0 \\ 0 & \mu(\mathcal{T}) - \mu(\widetilde{\mathcal{T}}) \end{pmatrix} P^{-1}U^0. \tag{2.7}$$

Taking norms and noting that $\|PV\|^2 = 2\|V\|^2$ and $\|P^{-1}V\|^2 = \frac{1}{2}\|V\|^2$, we get

$$\begin{aligned} \|U^N(\mathcal{T}) - U^N(\widetilde{\mathcal{T}})\| &= |\mu(\mathcal{T}) - \mu(\widetilde{\mathcal{T}})|\|U^0\|, \\ \|U^N(\widetilde{\mathcal{T}}) - U(T)\| &= |\mu(\widetilde{\mathcal{T}}) - e^{-iaT}|\|U^0\|. \end{aligned} \tag{2.8}$$

The second formula is the truncation error at the end of the time interval for equidistant timesteps. We next need to estimate the error in μ, for which we use the following result.

LEMMA 2.2. *For fixed a, T, and N, we have*

$$|\mu(\mathcal{T}) - \mu(\widetilde{\mathcal{T}})| = \frac{a^3\tilde{k}}{4(1+\frac{a^2\tilde{k}^2}{4})^2}\,\|\mathcal{T}-\widetilde{\mathcal{T}}\|^2 + \mathcal{O}(||\mathcal{T}-\widetilde{\mathcal{T}}||^3). \tag{2.9}$$

Proof. We write $\mu(\mathcal{T})$ as

$$\mu(\mathcal{T}) = \prod_{n=1}^{N} \lambda(k_n), \quad \lambda(k) = \frac{1 - i\frac{ak}{2}}{1 + i\frac{ak}{2}}.$$

By the second order Taylor–Young formula, we have

$$\mu(\mathcal{T}) = \mu(\widetilde{\mathcal{T}}) + \mu'(\widetilde{\mathcal{T}})\cdot(\mathcal{T}-\widetilde{\mathcal{T}}) + \frac{1}{2}\mu''(\widetilde{\mathcal{T}})\cdot(\mathcal{T}-\widetilde{\mathcal{T}})\cdot(\mathcal{T}-\widetilde{\mathcal{T}}) + \mathcal{O}(||\mathcal{T}-\widetilde{\mathcal{T}}||^3).$$

We now compute the derivative,

$$\frac{\partial\mu}{\partial k_n}(\widetilde{\mathcal{T}}) = \mu(\widetilde{\mathcal{T}})\frac{\lambda'(k_n)}{\lambda(k_n)} = \mu(\widetilde{\mathcal{T}})\frac{-ia}{1+\frac{a^2\tilde{k}^2}{4}}, \tag{2.10}$$

and therefore obtain for the linear term

$$\mu'(\widetilde{\mathcal{T}})\cdot(\mathcal{T}-\widetilde{\mathcal{T}}) = \mu(\widetilde{\mathcal{T}})\frac{-ia}{1+\frac{a^2\tilde{k}^2}{4}}\sum_n (k_n - \tilde{k}) = 0.$$

For the quadratic term, we differentiate (2.10) again with respect to k_n,

$$\frac{\partial^2\mu}{\partial k_n^2}(\widetilde{\mathcal{T}}) = \mu(\widetilde{\mathcal{T}})a^2\frac{-1+i\frac{a\tilde{k}}{2}}{(1+\frac{a^2\tilde{k}^2}{4})^2}, \quad \frac{\partial^2\mu}{\partial k_n\partial k_j}(\widetilde{\mathcal{T}}) = \mu(\widetilde{\mathcal{T}})a^2\frac{-1}{(1+\frac{a^2\tilde{k}^2}{4})^2}, \tag{2.11}$$

and therefore obtain

$$\mu''(\widetilde{\mathcal{T}})\cdot(\mathcal{T}-\widetilde{\mathcal{T}})\cdot(\mathcal{T}-\widetilde{\mathcal{T}}) = -a^2\mu(\widetilde{\mathcal{T}})\frac{\sum_{n,j}(k_n-\tilde{k})(k_j-\tilde{k}) - i\frac{a\tilde{k}}{2}\sum_n(k_n-\tilde{k})^2}{(1+\frac{a^2\tilde{k}^2}{4})^2},$$

which gives

$$\mu(\mathcal{T}) = \mu(\widetilde{\mathcal{T}}) - \frac{a^2\mu(\widetilde{\mathcal{T}})}{2}\frac{\sum_{n,j}(k_n-\tilde{k})(k_j-\tilde{k}) - i\frac{a\tilde{k}}{2}\sum_n(k_n-\tilde{k})^2}{(1+\frac{a^2\tilde{k}^2}{4})^2} + \mathcal{O}(||\mathcal{T}-\widetilde{\mathcal{T}}||^3).$$

The first sum in the numerator is zero, and hence

$$\mu(\mathcal{T}) - \mu(\widetilde{\mathcal{T}}) = \frac{i\mu(\widetilde{\mathcal{T}})a^3\tilde{k}\sum_n(k_n-\tilde{k})^2}{4(1+\frac{a^2\tilde{k}^2}{4})^2} + \mathcal{O}(||\mathcal{T}-\widetilde{\mathcal{T}}||^3),$$

which gives (2.9) when taking the modulus, since the quadratic term is purely imaginary and $|\mu(\widetilde{\mathcal{T}})| = 1$. ∎

It remains to estimate $\|\mathcal{T}-\widetilde{\mathcal{T}}\|^2$, which is done in what follows.

LEMMA 2.3. *The difference between the geometric and equal time partition satisfies the estimate*

$$\|\mathcal{T} - \widetilde{\mathcal{T}}\|^2 = \sum_{n=1}^{N} (k_n - \tilde{k})^2 = \tilde{k}^2 \frac{N(N^2-1)}{12}\varepsilon^2 + \mathcal{O}(\varepsilon^3). \tag{2.12}$$

Proof. The time step k_n in (2.4) has for ε small the expansion $k_n = \tilde{k}(1 + \alpha_n\varepsilon + \beta_n\varepsilon^2 + o(\varepsilon^2))$, with $\alpha_n = n - \frac{N+1}{2}$ and $\beta_n = n(n-N-2) + \frac{(N+1)(N+5)}{6}$. These coefficients satisfy the relations $\sum_n \alpha_n = \sum_n \beta_n = 0$, $\sum_n \alpha_n^2 = \frac{N(N^2-1)}{12}$, which leads to (2.12). ∎

We can now finish the proof of Theorem 2.1 by inserting (2.4) into (2.8) to obtain

$$\|U^N(\mathcal{T}) - U^N(\widetilde{\mathcal{T}})\| = \frac{a^3\tilde{k}}{4(1+\frac{a^2\tilde{k}^2}{4})^2}\tilde{k}^2\frac{N(N^2-1)}{12}\varepsilon^2\|U^0\| + \mathcal{O}(\varepsilon^3).$$

∎

2.2. Estimation of the roundoff error due to diagonalization. The space-time system (1.4) becomes in the scalar case

$$(B + a^2 I)\mathbf{u} = \mathbf{f},$$

and to diagonalize this system, we need to diagonalize the matrix B, which will involve unipotent lower triangular Toeplitz matrices.

DEFINITION 2.4. *A unipotent lower triangular Toeplitz matrix of size N is of the form*

$$T(x_1, \ldots, x_{N-1}) = \begin{pmatrix} 1 & & & \\ x_1 & \ddots & & \\ \vdots & \ddots & \ddots & \\ x_{N-1} & \ldots & x_1 & 1 \end{pmatrix}. \tag{2.13}$$

The next Lemma gives a closed form eigendecomposition of the time stepping matrix B and is proved using the techniques of q-hypergeometric series (see the book of Gasper and Rahman, *Basic Hypergeometric series* [27]).

LEMMA 2.5 (eigendecomposition of B). *If $k_n = q^{n-1}k_1$ with $q \neq 1$, then B has the eigendecomposition $B = VDV^{-1}$, with $D = diag(\frac{4}{k_n^2})$, and V and its inverse are unipotent lower triangular Toeplitz matrices given by*

$$V = T(p_1, \ldots, p_{N-1}) \quad \textit{with} \quad p_n := \prod_{j=1}^{n} \frac{1+q^j}{1-q^j}\,, \tag{2.14}$$

$$V^{-1} = T(q_1, \ldots, q_{N-1}) \quad \textit{with} \quad q_n := q^{-n} \prod_{j=1}^{n} \frac{1+q^{-j+2}}{1-q^{-j}}. \tag{2.15}$$

Proof. The matrix $B = (C^{-1}B_1)^2$ is diagonalizable if and only if all time steps k_n are different. The eigenvalues are then $\frac{4}{k_n^2}$, and the eigenvectors are those of $C^{-1}B_1$. An eigenvector of $C^{-1}B_1$ is such that $(B_1 - \frac{2}{k_n}C)X^{(n)} = 0$, and the matrix $B_1 - \frac{2}{k_n}C$

is lower bidiagonal, which leads to a recursive formula for the coefficients $X^{(n)}$ of the eigenvectors,

$$\begin{cases} X_1^{(n)} = X_2^{(n)} = \cdots X_{n-1}^{(n)} = 0, \\ X_{n+1}^{(n)} = \frac{k_n+k_{n+1}}{k_n-k_{n+1}}\ X_n^{(n)}, \ldots, X_N^{(n)} = \frac{k_n+k_N}{k_n-k_N}\ X_{N-1}^{(n)}. \end{cases}$$

Since the normalization is arbitrary, we can choose $X_n^{(n)} = 1$, which gives

$$X_j^{(n)} = \begin{cases} 0 & \text{for } j < n, \\ \displaystyle\prod_{l=n+1}^{j} \frac{k_n+k_l}{k_n-k_l} & \text{for } j > n. \end{cases}$$

The matrix of eigenvectors $V := (X^{(1)}, \ldots, X^{(N)})$ is lower triangular and unipotent, i.e., the diagonal elements equal 1. In case of the geometric mesh, it has the particular structure

$$V_{ij} = X_i^{(j)} = \prod_{l=j+1}^{i} \frac{k_j + k_l}{k_j - k_l} = \prod_{l=j+1}^{i} \frac{q^j + q^l}{q^j - q^l} = \prod_{l=j+1}^{i} \frac{1 + q^{k-j}}{1 - q^{k-j}} = \prod_{l=1}^{i-j} \frac{1 + q^l}{1 - q^l},$$

which shows that V_{ij} is a function of $i-j$ only, and therefore V is a unipotent Toeplitz matrix. Consider now the inverse of V. First, it is easy to see that it is also unipotent Toeplitz. To establish (2.15) is equivalent to prove that

$$\text{for } 1 \le n \le N-1, \ \sum_{j=0}^{n} p_{n-j} q_j = 0, \ \text{with the convention that } p_0 = q_0 = 1. \tag{2.16}$$

To show this result, we need to define Heine's q-hypergeometric series [27],

$$ {}_2\varphi_1(a_1, a_2; b; q; x) := \sum_{n=0}^{\infty} \frac{(a_1; q)_n (a_2; q)_n}{(q; q)_n (b; q)_n} x^n, \tag{2.17}$$

where we assume that b is not an integer negative power of q, and the *q-rising factorial* is

$$(a; q)_k := \prod_{i=0}^{k-1} (1 - q^i a), \quad (a; q)_\infty := \prod_{i=0}^{\infty} (1 - a q^i).$$

Heine's summation formula states that for any two real numbers a_1 and a_2 and for any b with $|b/(a_1 a_2)| < 1$, we have

$$ {}_2\varphi_1(a_1, a_2; b; q; b/a_1 a_2) = \frac{(b/a_1; q)_\infty (b/a_2; q)_\infty}{(b; q)_\infty (b/a_1 a_2; q)_\infty}. \tag{2.18}$$

We now use Heine's summation formula to conclude the proof: let N be an integer larger than 1; setting $a_1 = q^{-N}$ and $a_2 = a$ yields the q-Chu–Vandermonde formula

$$ {}_2\varphi_1(q^{-N}, a; b; q; \frac{b}{a} q^N) = \frac{(b/a; q)_N}{(b; q)_N}. \tag{2.19}$$

Since $(q^{-N}; q)_n = 0$ as soon as $n \ge N+1$, the series on the left is a finite sum, and we obtain

$$\sum_{n=0}^{N} \frac{(q^{-N}; q)_n (a; q)_n}{(q; q)_n (b; q)_n} \left(\frac{b}{a} q^N\right)^n = \frac{(b/a; q)_N}{(b; q)_N}. \tag{2.20}$$

This is an equality between rational fractions in q, which is valid for all q different from $0, 1$, and all $a \neq 0$ and all b which are not an integer negative power of q. Choosing $a = -q$ and $b = -q^{-N+2}$, we have

$$(b/a;q)_N = (q^{-N+1};q)_N = (1-q^{-N+1})\cdots(1-q^{-N+1}q^{N-1}) = 0, \tag{2.21}$$

and thus the right hand side in (2.20) vanishes, and we get

$$\sum_{n=0}^{N} \frac{(-q;q)_n (q^{-N};q)_n}{(q;q)_n (-q^{-N+2};q)_n} q^n = 0. \tag{2.22}$$

We express now p_n in terms of q-analogues,

$$p_n = \prod_{i=1}^{n} \frac{1+q^i}{1-q^i} = \prod_{i=0}^{n-1} \frac{1+qq^i}{1-qq^i} = \frac{(-q;q)_n}{(q;q)_n}. \tag{2.23}$$

Defining

$$\tilde{q}_{N-n} := \frac{(q^{-N};q)_n}{(-q^{-N+2};q)_n} q^n \quad \text{and} \quad q_{N-n} := \frac{\tilde{q}_{N-n}}{\tilde{q}_0},$$

(2.22) becomes

$$\sum_{n=0}^{N} p_n \tilde{q}_{N-n} = \sum_{n=0}^{N} p_n q_{N-n} = 0.$$

Introducing the definition of q_{N-n} and performing similar steps as in (2.23) backwards shows that q_n is equal to the value in (2.14), and the proof is complete. ∎

In the steps (a) and (c) of the direct time parallel solver (1.7) based on diagonalization, the condition number of the eigenvector matrix S has a strong influence on the accuracy of the results, and normalizing the eigenvectors from Lemma 2.5 with respect to the ℓ^2 norm leads to an asymptotically better condition number,

$$S := V\widetilde{D}, \quad \widetilde{D} = \operatorname{diag}\left(\frac{1}{\sqrt{1 + \sum_{i=1}^{N-n} |p_i|^2}} \right). \tag{2.24}$$

We now study the roundoff error when solving the system $(B + a^2 I)\mathbf{u} = \mathbf{f}$ numerically using the diagonalization

$$S(D + a^2 I)S^{-1}\mathbf{u} = \mathbf{f}. \tag{2.25}$$

Due to roundoff, we will obtain an approximate solution $\hat{\mathbf{u}}$, and the difference between the exact solution $\mathbf{u}$ and $\hat{\mathbf{u}}$ is classically related to the condition number of the matrix S. We first also give such an estimate, but later provide a more accurate one using the structure of the problem at hand.

LEMMA 2.6. *Let* $\mathbf{u}$ *be the exact solution of* $(B + a^2 I)\mathbf{u} = \mathbf{f}$*, and* $\hat{\mathbf{u}}$ *be the numerically computed solution using the factored form* (2.25)*, and let* $\underline{u}$ *denote the machine precision. Then for any norm,*

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|}{\|\mathbf{u}\|} \leq cond(B) \frac{\|\delta B\|}{\|B\|} \leq (2N+1)\underline{u}\, \|B^{-1}\|\, \|\, |S|\, |S^{-1}|\, \|\, \|D + aI\|, \tag{2.26}$$

where $|S|$ *denotes the matrix containing the elements of* S *in modulus.*

Proof. Using backward error analysis [29], the computed solution satisfies the perturbed systems

$$(S+\delta S_1)\hat{\mathbf{g}}=\mathbf{f},\quad (D+a^2I+\delta D)\hat{\mathbf{w}}=\hat{\mathbf{g}},\quad (S^{-1}+\delta S_2)\hat{\mathbf{u}}=\hat{\mathbf{w}},$$

and since S and S^{-1} are triangular and D is diagonal we get (see [29])

$$|\delta S_1|\le N\underline{u}|S|+\mathcal{O}(\underline{u}^2),\ |\delta S_2|\le N\underline{u}|S^{-1}|+\mathcal{O}(\underline{u}^2),\ |\delta D|\le \underline{u}|D+aI|+\mathcal{O}(\underline{u}^2).$$

Solving numerically $(B+a^2I)\mathbf{u}=\mathbf{f}$ using the factored form (2.25) is equivalent to solving exactly $(S+\delta S_1)(D+a^2I+\delta D)(S^{-1}+\delta S_2)\ \hat{\mathbf{u}}=\mathbf{f}$, which is of the form

$$(B+\delta B)\hat{\mathbf{u}}=\mathbf{f},\quad |\delta B|\le (2N+1)\underline{u}|S|\,|S^{-1}|\,|D+aI|+\mathcal{O}(\underline{u}^2).$$

The relative error then satisfies (see [29])

$$\frac{\|\mathbf{u}-\hat{\mathbf{u}}\|}{\|\mathbf{u}\|}\le \mathrm{cond}(B)\frac{\|\delta B\|}{\|B\|}\le \|B^{-1}\|\,\|\delta B\|, \tag{2.27}$$

and inserting $\|\delta B\|$ from before gives the result in (2.26). ∎

We next provide an asymptotic estimate for the term $\|\,|S|\,|S^{-1}|\|$ on the right in (2.26) in the case of a geometric time partition using the infinity norm.

LEMMA 2.7 (asymptotic condition number estimate). *For $q=1+\varepsilon$, we have*

$$\|\,|S|\,|S^{-1}|\,\|_\infty\sim \frac{2^{2(N-1)}}{(N-1)!}\,\varepsilon^{-(N-1)}, \tag{2.28}$$

$$cond_\infty(S)\sim \frac{2^{(N-1)}\,N}{\lfloor\frac{N}{2}\rfloor!\lfloor\frac{N-1}{2}\rfloor!}\varepsilon^{-(N-1)}. \tag{2.29}$$

Proof. Note first that $|q_n|\sim|p_n|\sim\frac{2^n}{n!\,\varepsilon^n}$. We next define $\gamma_n:=\sqrt{1+\sum_{j=1}^{N-n}|p_j|^2}$ and $\tilde{d}_n:=\frac{1}{\gamma_n}$, which implies that $\widetilde{D}=diag(\tilde{d}_n)$. Then $\gamma_n\sim|p_{N-n}|$, and we obtain

$$\|S\|_\infty=\max_n\sum_{j=1}^{n}\frac{|p_{n-j}|}{\gamma_j}\sim\max_n\sum_{j=1}^{n}\frac{|p_{n-j}|}{|p_{N-j}|}\sim\max_n\sum_{j=1}^{n}\frac{(N-j)!}{(n-j)!}\left(\frac{\varepsilon}{2}\right)^{N-n}\sim N. \tag{2.30}$$

By definition $S^{-1}=\widetilde{D}^{-1}V^{-1}=\widetilde{D}^{-1}T(q_1,\cdots,q_{N-1})$; that is, the line n of $T(q_1,\cdots,q_{N-1})$ is multiplied by γ_n. Therefore

$$\begin{aligned}\|S^{-1}\|_\infty&=\max_n\gamma_n\sum_{j=0}^{n-1}|q_j|\sim\max_n\gamma_n|q_{n-1}|\sim\max_n\gamma_n|p_{n-1}|\\&\sim\max_n|p_{N-n}||p_{n-1}|\sim\left(\frac{2}{\varepsilon}\right)^{(N-1)}\max_{1\le n\le N-1}\frac{1}{(n-1)!(N-n)!}.\end{aligned} \tag{2.31}$$

The last term on the right hand side can be explicitly computed by noting that

$$\max_{1\le n\le N-1}\frac{1}{(n-1)!(N-n)!}=\frac{1}{(N-1)!}\max_{1\le n\le N-1}\binom{N-1}{n-1}=\frac{1}{(N-1)!}\binom{N-1}{\lfloor\frac{N-1}{2}\rfloor},$$

which follows from the Pascal triangle, and thus we get

$$\max_{1\le n\le N-1}\frac{1}{(n-1)!(N-n)!}=\frac{1}{\lfloor\frac{N}{2}\rfloor!\lfloor\frac{N-1}{2}\rfloor!}.$$

Combining (2.30) and (2.31) then yields (2.29). Similarly, we also obtain

$$\| \, |S| \, |S^{-1}| \, \|_\infty = \max_i \sum_j (|S| \, |S^{-1}|)_{ij} = \max_i \sum_{j=1}^{i} \sum_{k=j}^{i} |p_{i-k}| \, |q_{k-j}|$$
$$\sim \frac{2^{N-1}}{\varepsilon^{N-1}} \sum_{k=1}^{N} \frac{1}{(N-k)!(k-1)!}$$

and explicitly summing the last term on the right, $\sum_{k=1}^{N} \frac{1}{(N-k)!(k-1)!} = \frac{2^{N-1}}{(N-1)!}$, leads to the desired estimate (2.28). ∎

We are now ready to give a precise asymptotic error estimate of the roundoff error that is induced by the diagonalization in the direct time parallel solver (1.7).

THEOREM 2.8 (asymptotic roundoff error estimate). *Let* $\mathbf{u}$ *be the exact solution of* $(B + a^2 I)\mathbf{u} = \mathbf{f}$ *and* $\hat{\mathbf{u}}$ *be the computed solution from the direct time parallel solver* (1.7) *applied to* (2.1)*, and let* $\underline{u}$ *denote the machine precision. Then*

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lesssim \psi_1(\frac{aT}{2N}, N) \underline{u} \; \varepsilon^{-(N-1)}, \tag{2.32}$$

where $\psi_1(y, N) := \frac{2^{2(N+1)}}{(N-1)!}(1 + 2N(N-1))(1 + y^2)$.

Proof. The form of the matrices involved is very well adapted to estimates in the ℓ^∞ norm. We evaluate the various quantities in (2.26) starting with the norm of B^{-1}, which we can obtain by computing $B^{-1} = (B_1^{-1}C)^2$ explicitly and noting that the infinity norm comes from the last line, which gives

$$\|B^{-1}\|_\infty = (2k_1)^2 \left(1 + \frac{2q(q^{N-1} - 1)(q^N - 1)}{(q-1)^2}\right).$$

Now for $q = 1 + \varepsilon$, we get by expanding

$$\|B^{-1}\|_\infty \sim (2k_1)^2(1 + 2N(N-1)).$$

We now use that $D = diag(\frac{4}{k_i^2})$, and obtain

$$\|B^{-1}\|_\infty \|D + aI\|_\infty \sim (2k_1)^2(4/k_1^2 + a^2)(1 + 2N(N-1)) \sim 16 \left(1 + \frac{a^2 k_1^2}{4}\right)(1 + 2N(N-1)). \tag{2.33}$$

Inserting (2.33) into (2.26) gives

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lesssim 16 \left(1 + \frac{a^2 k_1^2}{4}\right)(1 + 2N(N-1)) \underline{u} \, \| \, |S| \, |S^{-1}| \, \|_\infty .$$

Replacing the last term on the right hand side using (2.28) leads to

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lesssim \frac{2^{2(N+1)}}{(N-1)!}(1 + 2N(N-1)) \left(1 + \frac{a^2 k_1^2}{4}\right) \underline{u} \, \varepsilon^{-(N-1)}. \tag{2.34}$$

Approximating k_1 by $\tilde{k}$ as ε goes to zero finally gives

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lesssim \frac{2^{2(N+1)}}{(N-1)!}(1 + 2N(N-1)) \left(1 + \left(\frac{aT}{2N}\right)^2\right) \underline{u} \, \varepsilon^{-(N-1)},$$

which proves the result. ∎

Remark 2.9. The more usual roundoff estimates use the condition number of the matrix S in (2.26) instead of the norm of $|S|\,|S^{-1}|$. Then using (2.29), the estimate (2.32) would become

$$\frac{\|\mathbf{u}-\hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lesssim \psi_2(\frac{aT}{2N},N)\underline{u}\;\varepsilon^{-(N-1)}, \tag{2.35}$$

where $\psi_2(y,N) := \frac{N(1+2N(N-1))\,2^{N+3}}{\lfloor \frac{N}{2}\rfloor!\lfloor\frac{N-1}{2}\rfloor!}(1+y^2)$, which we will see is comparable to our first estimate.

We next present a sharper estimate using the special structure of S and S^{-1}.

LEMMA 2.10. *For any* $\mathbf{f}$*, for any diagonal matrix* Δ*, let* $\mathbf{u} = S\Delta S^{-1}\mathbf{f}$ *and* $\hat{\mathbf{u}}$ *be the computed value of* $\mathbf{u}$*. Then*

$$\|\mathbf{u}-\hat{\mathbf{u}}\|_\infty \lesssim \frac{2^{2(N-1)}}{(N-1)!}\varepsilon^{-(N-1)}\;\underline{u}\;\|\Delta\|_\infty\;\|\mathbf{f}\|_\infty. \tag{2.36}$$

Proof. Standard truncation error estimates as in Theorem 2.8 show that the approximate value of $SDS^{-1}\mathbf{f}$ gives an error which can be bounded by

$$\begin{aligned}&|\overline{S\Delta S^{-1}\mathbf{f}} - S\Delta S^{-1}\mathbf{f}|\\ &\le \|\mathbf{f}\|_\infty\;\|\Delta\|_\infty\;\underline{u}\begin{pmatrix}1\\ 1+|p_1|+|q_1|\\ \vdots\\ |p_{N-1}|+|p_{N-2}|(|q_1|+1)+\cdots+|q_{N-1}|+|q_{N-2}||p_1|+\cdots+|q_1|+1\end{pmatrix}\\ &\quad+\mathcal{O}(\underline{u}^2).\end{aligned}$$

Considering the leading term in ε, suppose that ε and N are such that the largest coefficient p_{N-1} satisfies $|p_{N-1}|\underline{u} \ll 1$, to obtain

$$\|\overline{S\Delta S^{-1}\mathbf{f}} - S\Delta S^{-1}\mathbf{f}\|_\infty \lesssim \|\mathbf{f}\|_\infty\;\|\Delta\|_\infty\;\underline{u}\sum_{n=0}^{N-1}|p_j q_{N-1-j}|.$$

We can estimate the sum using an asymptotic expansion,

$$\sum_{n=0}^{N-1}|p_j q_{N-1-j}| \sim \sum_{n=0}^{N-1}\frac{2^n}{n!}\,\varepsilon^{-n}\;\frac{2^{N-1-n}}{(N-1-n)!}\,\varepsilon^{-(N-1-n)} \sim \frac{2^{2(N-1)}}{(N-1)!}\varepsilon^{-(N-1)},$$

and we obtain

$$\frac{\|\overline{S\Delta S^{-1}\mathbf{f}} - S\Delta S^{-1}\mathbf{f}\|_\infty}{\|\mathbf{f}\|_\infty} \lesssim\; \|\Delta\|_\infty\frac{2^{2(N-1)}}{(N-1)!}\varepsilon^{-(N-1)}\underline{u}, \tag{2.37}$$

which concludes the proof. ∎

We can now prove the following sharper estimate for the roundoff error.

THEOREM 2.11 (sharper asymptotic roundoff error estimate). *For any* $\mathbf{f}$*, let* $\mathbf{u} = S(D+aI)^{-1}S^{-1}\mathbf{f}$ *and* $\hat{\mathbf{u}}$ *be the computed value of* $\mathbf{u}$*. Then*

$$\frac{\|\mathbf{u}-\hat{\mathbf{u}}\|_\infty}{\|U^0\|} \lesssim \psi_3\left(\frac{aT}{2N},N\right)\underline{u}\;\varepsilon^{-(N-1)}, \tag{2.38}$$

where $\psi_3(y,N) := \frac{2^{2N-\frac{1}{2}}N}{(N-1)!}\frac{1}{y^2+1}$.

Proof. We have $\|(D+a^2I)^{-1}\|_\infty = \frac{1}{a^2+\frac{4}{k_N^2}}$, and replacing Δ by $(D+a^2I)^{-1}$ in (2.36), we obtain

$$\|\mathbf{u}-\widehat{\mathbf{u}}\|_\infty \lesssim \frac{2^{2(N-1)}}{(N-1)!}\varepsilon^{-(N-1)}\ \underline{u}\ \frac{1}{a^2+\frac{4}{k_N^2}}\ \|\mathbf{f}\|_\infty. \tag{2.39}$$

We need now an estimate of $\|\mathbf{f}\|_\infty$ with

$$\mathbf{f} = C^{-1}(B_1C^{-1}F_0+G_0)\mathbf{e_1}, \quad F_0 = \frac{1}{k_1}f+\frac{g}{2}, \quad G_0 = \frac{1}{k_1}g-\frac{a^2}{2}f. \tag{2.40}$$

We first explicitly compute the inverse of C and obtain the lower triangular matrix $C^{-1}_{ij} = 2(-1)^{i-j}$, $i \geq j$. We can then evaluate the terms in (2.40) to get

$$(C^{-1}\mathbf{e}_1)_i = 2(-1)^{i-1} \implies (C^{-1}B_1C^{-1}\mathbf{e}_1)_i = 4(-1)^{i-1}\left(\frac{1}{k_1}+2\sum_{j=2}^{i}\frac{1}{k_j}\right),$$

where the sum equals zero for $i=1$. Using that the time steps are geometric, $k_j = q^{j-1}k_1$, we define

$$s_i := 1+2\sum_{j=1}^{i-1}\frac{1}{q^j} \text{ for } i\geq 2, \quad s_1 = 1,$$

and obtain $(C^{-1}B_1C^{-1}\mathbf{e}_1)_i = \frac{4}{k_1}(-1)^{i-1}s_i$, which we introduce into (2.40) to get

$$\mathbf{f}_i = (-1)^{i-1}\left(\frac{4}{k_1}F_0s_i+2G_0\right) = (-1)^{i-1}\left(\frac{4}{k_1}s_i\left(\frac{1}{k_1}f+\frac{g}{2}\right)+2\left(\frac{1}{k_1}g-\frac{a^2}{2}f\right)\right).$$

Taking the modulus and rearranging terms leads to

$$|\mathbf{f}_i| = \left|\frac{4}{k_1^2}f\left(s_i-\frac{a^2k_1^2}{4}\right)+\frac{2}{k_1}g(s_i+1)\right|.$$

Defining $y_1 := \frac{k_1a}{2}$, we obtain

$$|\mathbf{f}_i| = \left|\frac{4}{k_1^2}f(s_i-y_1^2)+\frac{2a}{k_1}\frac{g}{a}(s_i+1)\right| = \frac{4}{k_1^2}\left|f(s_i-y_1^2)+y_1\frac{g}{a}(s_i+1)\right|.$$

For sufficiently small k_1, y_1 is smaller than 1, and since $s_i > 1$ and the sequence s_i is increasing, we obtain

$$\|\mathbf{f}\|_\infty \leq \frac{4}{k_1^2}\left(|f|s_N+\frac{|g|}{a}(s_N+1)\right) \leq \frac{4}{k_1^2}(s_N+1)\left(|f|+\frac{|g|}{a}\right) \leq \frac{4\sqrt{2}}{k_1^2}(s_N+1)\|U_0\|.$$

We use now that q is close to 1 to estimate asymptotically

$$s_N+1 = 2(1+\sum_{j=1}^{N-1}\frac{1}{q^j}) = 2\sum_{j=0}^{N-1}\frac{1}{q^j} = 2\frac{1-\frac{1}{q^N}}{1-\frac{1}{q}} \lesssim 2N,$$

and therefore

$$\|\mathbf{f}\|_\infty \lesssim \frac{8N\sqrt{2}}{k_1^2}\|U_0\|.$$

This allows us to also estimate $\|\mathbf{f}\|_\infty$ in (2.39) asymptotically,

$$\|\mathbf{u}-\widehat{\mathbf{u}}\|_\infty \lesssim \frac{2^{2(N-1)}}{(N-1)!}\varepsilon^{-(N-1)}\ \underline{u}\ \frac{1}{a^2+\frac{4}{k_N^2}}\ \frac{8N\sqrt{2}}{k_1^2}\|U_0\|.$$

We now combine the last two terms asymptotically using that

$$\frac{\frac{4}{k_1^2}}{a^2+\frac{4}{k_N^2}} \sim \frac{1}{1+y^2}, \quad y := \frac{aT}{2N},$$

and obtain (2.38). ∎

We show in Figure 2.2 graphically that $\psi_1(\frac{aT}{2N},N)$ and $\psi_2(\frac{aT}{2N},N)$ are comparable, while $\psi_3(\frac{aT}{2N},N)$ is substantially smaller, and thus gives a much sharper estimate.

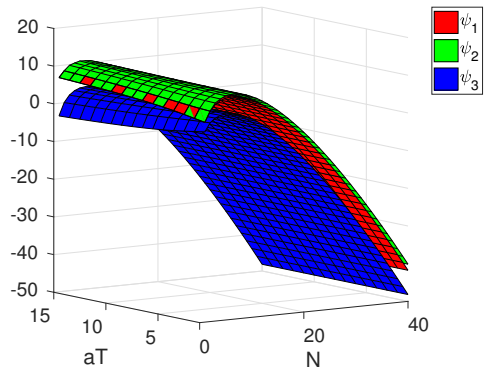

FIG. 2.2. *Comparison of the logarithm of the functions ψ_j, $j=1,2,3$.*

2.3. Optimization of the algorithm parameters. To start, we perform a numerical experiment for the scalar model problem (2.1) with $a=1$ and initial conditions $u(0)=1$, $\dot{u}(0)=0$. We show in Figure 2.3 first as a reference the measured discretization error on an equidistant time partition with N points (called "error 1," and independent of ε). Next we plot the error due to the geometric time partition, which increases when ε is increasing (called "error 2"), and our theoretical estimate $\phi\varepsilon^2$ from Theorem 2.1, which is rather sharp, since it coincides in the figure with the numerically measured values, except for large ε. We finally also plot the roundoff error we measure due to the diagonalization procedure, which decreases when ε is increasing (called "error 3"), and our theoretical bound $\psi_3\varepsilon^{-(N-1)}\underline{u}$ from Theorem 2.11. From these plots, we can see that an optimized choice of ε would balance these two errors, and for the case on the left with $T=5$ and $N=10$ (10 processors) it would be $\varepsilon^* \approx 4.5e-2$. In this case, the additional errors due to the time parallelization would remain much smaller than the actual discretization error one would have on an equidistant time grid. For the case on the right in Figure 2.3 with $T=10$ and $N=20$ (20 processors), the best choice would be $\varepsilon^* \approx 1e-1$, and now the additional errors due to the parallelization would be of the same order as the actual discretization error one would have on an equidistant time grid.

To determine a formula for the best ε^*, we can bound the actual error of the parallel time integrator by adding and subtracting $u^N(\mathcal{T}_1)$ and also $u^N(\mathcal{T}_q)$ with $q=1+\varepsilon$ and using the triangle inequality, we obtain at time T an error estimate between

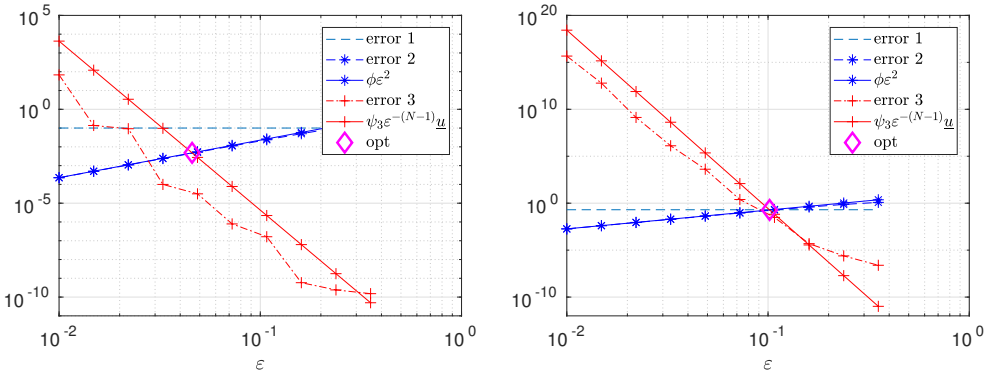

FIG. 2.3. *Discretization and parallelization errors, together with our theoretical bounds. Left:* $T = 5$, $a = 1$, $N = 10$. *Right:* $T = 10$, $a = 1$, $N = 20$.

the exact solution at time $T = t_N$ and the last value $\hat{u}_N$ of the approximate solution computed by diagonalization, namely,

$$\frac{|u(t_N) - \hat{u}_N|}{\|U^0\|} \le \underbrace{\frac{|u(t_N) - u^N(\mathcal{T}_1)|}{\|U^0\|}}_{\le \text{ error 1}} + \underbrace{\frac{|u^N(\mathcal{T}_1) - u^N(\mathcal{T}_q)|}{\|U^0\|}}_{\le \text{ error 2}} + \underbrace{\frac{|u^N(\mathcal{T}_q) - \hat{u}_N|}{\|U^0\|}}_{\le \text{ error 3}}. \tag{2.41}$$

The first term, error 1, is the truncation error of the sequential method using equal time steps; see also Figure 2.3. The second term, error 2, is due to the geometric time partition and was estimated asymptotically in Theorem 2.1 to be $\phi\varepsilon^2$. The last term, error 3, can be estimated using Theorem 2.11. Because the second term is decreasing in ε and the last term is growing in ε (see also Figure 2.3), we equilibrate them asymptotically and obtain the following.

THEOREM 2.12 (optimized geometric time partition). *Suppose the time steps are geometric,* $k_n = q^{n-1}k_1$, *and* $q = 1 + \varepsilon$ *with* ε *small. Let* $\underline{u}$ *be the machine precision. Fix* a, T, *and* N. *For* $\varepsilon = \varepsilon^*(aT, N)$ *with*

$$\varepsilon^*(aT, N) = \left(\frac{3\ 2^{2N}}{(N^2-1)(N-1)!} \frac{1+y^2}{y^3} \underline{u} \right)^{\frac{1}{N+1}} \quad \text{with } y = \frac{aT}{2N}, \tag{2.42}$$

the error due to time parallelization is asymptotically comparable to the one produced by the geometric time partition.

Proof. Equilibration of the error produced by the geometric mesh and the diagonalization means imposing $\phi(y, N)\varepsilon^2 = \psi_3(y, N)\,\underline{u}\,\varepsilon^{-(N-1)}$, and thus $\varepsilon^* = (\frac{\psi_3}{\phi}(y, N)\,\underline{u})^{\frac{1}{N+1}}$, where ψ_3 is defined in Theorem 2.11 and ϕ is defined in Theorem 2.1. Introducing these quantities and simplifying leads to (2.42). ∎

We see in Figure 2.3 that the theoretically predicted optimized ε^* marked by a rhombus is a good estimate of the numerical best parameter. We next show in Figure 2.4 on the left the optimized value $\varepsilon^*(aT, N)$ from Theorem 2.12 as a function of the two arguments aT and N. Choosing $\varepsilon = \varepsilon^*(aT, N)$, the ratio between the additional errors due to parallelization to the truncation error of the fixed time step method (error 2/error 1) is shown in Figure 2.4 on the right, together with the relative truncation error for a fixed time step method in red (error 1). We see that one can

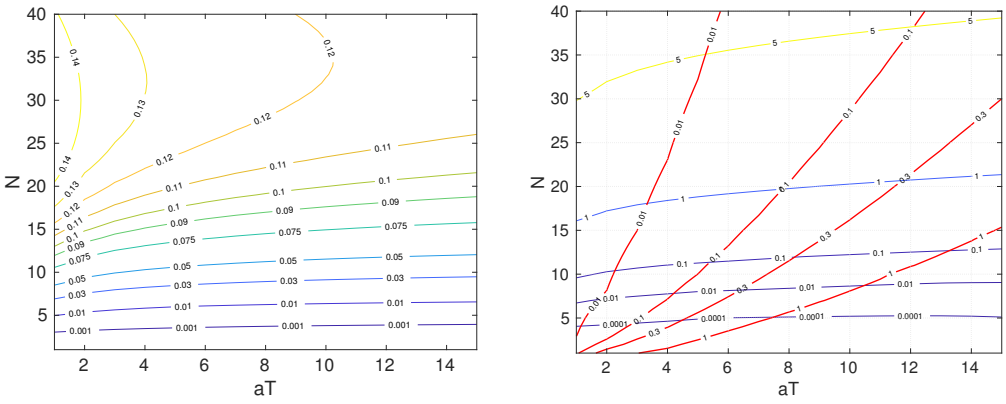

FIG. 2.4. *Left: optimized choice $\varepsilon^*(aT, N)$ from Theorem 2.12. Right: ratio of the additional errors due to parallelization to the truncation error of the fixed step method with this choice of $\varepsilon^*(aT, N)$ (error 2/error 1), and relative truncation error for a fixed time step method in red (error 1).*

use approximately 20 processors ($N = 20$) for a reasonably large range of aT by only increasing the error with the same order one would have had using a sequential integrator on a fixed time mesh, and when using 10 processors, the additional errors due to time parallelization are negligible (about 10% of the errors on an equidistant time grid). When using 40 processors however ($N = 40$), a five-fold error has to be expected, due to the parallelization, compared to the sequential time stepping on a fixed time mesh, even with the best possible value $\varepsilon = \varepsilon^*(aT, N)$. This shows that one can not parallelize with the diagonalization technique in time using an arbitrarily large number of processors.

Remark 2.13. To obtain an estimate for the best ε to chose in the wave equation case, we recall the Fourier transform in space with Fourier variable ξ which corresponds to our parameter a. We can thus apply our results from the ODE analysis with $a^2 = |\xi|^2$, where ξ is the dominant frequency in the solution we are trying to calculate. If we have no information about the dominant frequency in the solution, we can also use the upper bound $\xi_{\max}^2 := \pi^2/h^2$ in one dimension (1D) (h the mesh size in space), and $|\xi_{\max}|^2 := \pi^2/h_1^2 + \pi^2/h_2^2$ (h_1 and h_2 the two mesh sizes in space in two dimensions (2D)), since $\phi\varepsilon^*(aT, N)^2$ is growing in aT; see Figure 2.4 on the right.[2] Note that the growth seems to slow down for large values of aT, so that one can even estimate the value of $\varepsilon^*(aT, N)$ outside the plotted range of aT.

3. Numerical experiments. We now test our theoretical results on the wave equation (1.1) in one and two spatial dimensions, and also on an industrial test case using the equations of elasticity, where we use time windowing in order to integrate over a larger number of time steps than authorized by the diagonalization technique.

3.1. The wave equation in one dimension. We solve the wave equation in one dimension on $[0, 1]$, with homogeneous Dirichlet boundary conditions and as initial conditions $u_0(x) = \sin(\pi x)$, $u_1(x) = 0$. The spatial mesh is $h = \frac{1}{10}$, and we use $N = 10$ time steps on the time interval $(0, 1)$. We first show in Figure 3.1 three

[2] In fact we need only spectral information of the spatial operator to determine the relevant range of a, and thus can also handle variable coefficients; see the example in subsection 3.3.

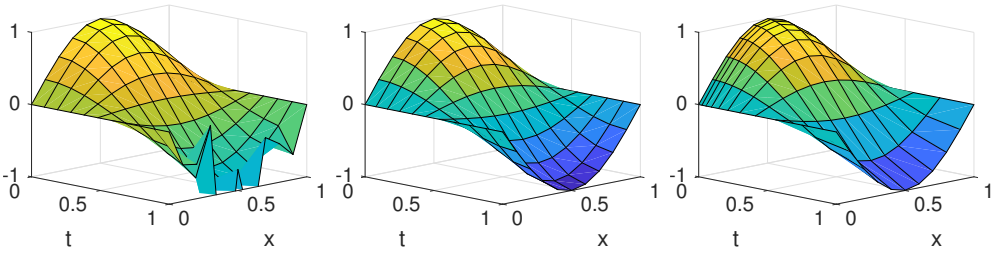

FIG. 3.1. *Approximate solutions obtained by the time parallel algorithm using diagonalization. Left:* $\varepsilon = 0.015$. *Middle:* $\varepsilon = \varepsilon^* = 0.05$. *Right:* $\varepsilon = 0.3$.

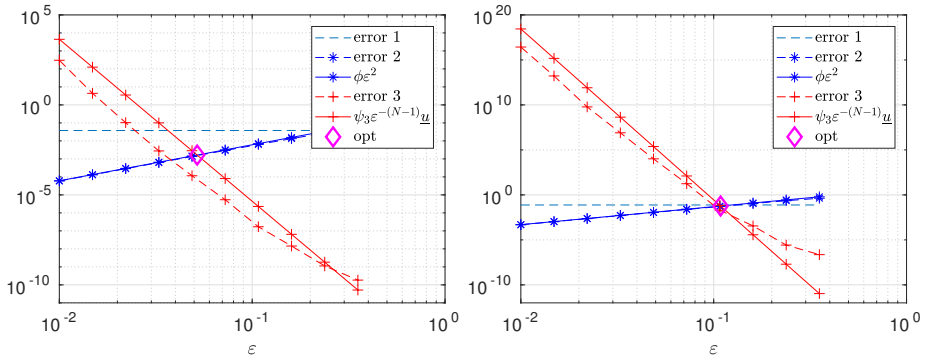

FIG. 3.2. *Discretization and parallelization errors in* 1*D, together with our theoretical bounds for the PDE. Left:* $T = 1$, $N = 10$. *Right:* $T = 2$, $N = 20$.

numerical solutions obtained for three different geometric time partitions using the diagonalization approach. We clearly see on the left that the ε chosen is too small; the solution becomes inaccurate and presents strong oscillations due to the roundoff error introduced by the diagonalization. On the right ε is too big, and the solution is less accurate for the second part of the time interval. In the middle, we chose the optimal ε^* and obtain an accurate solution, comparable to the one computed sequentially on a fixed step size partition. In Figure 3.2 we show the corresponding measured discretization and diagonalization errors compared to our theoretical bounds. We see again on the left that for $T = 1$ using $N = 10$ (10 time steps corresponding to computing with 10 processors), using the optimized choice of ε leads to a very small error increase compared to the already existing truncation error, as in the ODE case in Figure 2.3. For $N = 20$ however (corresponding to using 20 processors) on a twice as long time interval $T = 2$, the errors due to parallelization are now comparable to the existing truncation error on an equally spaced time grid, and as in the ODE case, one can not use more than about 20 processors for the time parallelization of the wave equation without introducing substantial additional errors due to the parallelization method based on diagonalization.

3.2. The wave equation in two dimensions. We now solve the wave equation on the unit square, with homogeneous Dirichlet boundary conditions and as initial conditions $u_0(x, y) = \sin(\pi x)\sin(\pi y)$, $u_1(x, y) = 0$, using $h = \frac{1}{10}$ in space. In Figure 3.3 we show again the corresponding measured discretization and diagonalization errors compared to our theoretical bounds, and we see that the results are very similar to the one dimensional case.

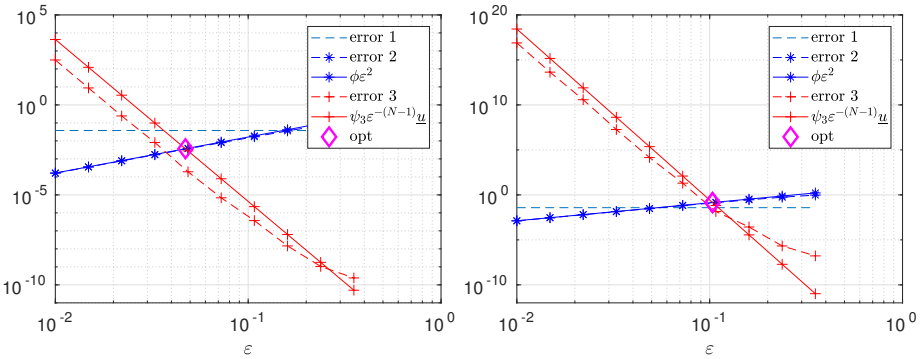

FIG. 3.3. *Discretization and parallelization errors in 2D, together with our theoretical bounds for the PDE. Left:* $T = 1$, $N = 10$. *Right:* $T = 2$, $N = 20$.

We also implemented this algorithm on a computer cluster adding to the initial algorithm a time windowing process in order to compare costs and accuracy for a larger number of time steps, and to truly evaluate parallel performance. Time windowing means that the initial global time interval $(0, \tilde{T})$ is first partitioned into so-called time windows of length T, and then the diagonalization technique is applied sequentially to the time windows, starting with the first one, then continuing with the second one, and so on. We denote by NWin the number of time windows, where each time window consists of N time steps to apply the diagonalization technique in each time window. The algorithm now performs 5 steps for each time window, namely,

$$
\begin{array}{lrcl}
(1) & F & = & (\frac{1}{k_1}u_h^0 + \frac{1}{2}\dot{u}_h^0, \ldots, 0), \\
 & G & = & (\frac{1}{k_1}\dot{u}_h^0 + \frac{1}{2}\Delta_h u_h^0, 0, \ldots, 0), \\
(a) & \mathbf{g} & = & (S_1 \otimes I_x)F + (S_2 \otimes I_x)G, \\
(b) & (\frac{1}{k_n^2} - \Delta_h)\mathbf{w}^n & = & \mathbf{g}^n, \quad 1 \le n \le N, \\
(c) & \mathbf{u} & = & (S \otimes I_x)\mathbf{w}, \\
(2) & \dot{\mathbf{u}} & = & (B_1^{-1}C \otimes I_x)\Delta_h \mathbf{u} + (B_1^{-1} \otimes I_x)G,
\end{array}
\tag{3.1}
$$

where $S_1 = S^{-1}C^{-1}B_1$ and $S_2 = S^{-1}C^{-1}$. Compared to the three basic steps (a), (b), and (c) of the initial algorithm (1.7), the additional step (1) serves to set up the proper initial values of the current time window, and the additional step (2) has to be computed in order to initialize F and G for the next time window.

All the time matrices are lower triangular matrices and all Kronecker products $Y = (A \otimes I_x)X$ where $X = (X_1, X_2, \ldots, X_N)$ and $Y = (Y_1, Y_2, \ldots, Y_N)$ can be written as

$$
\mathbf{Y} = (A \otimes I_x)\mathbf{X} \iff \left\{ \begin{array}{rcl}
Y_1 &=& A(1,1)X_1, \\
Y_2 &=& A(2,1)X_1 + A(2,2)X_2, \\
 & \vdots & \\
Y_i &=& A(i,1)X_1 + A(i,2)X_2 + \ldots + A(i,i)X_i, \\
 & \vdots & \\
Y_N &=& A(N,1)X_1 + A(N,2)X_2 + \ldots + A(N,N)X_N.
\end{array} \right.
\tag{3.2}
$$

In our implementation, we overlap computation and communication when working in a given time window: once the data from step (1) in (3.1) is available to perform the

products of the form (3.2) in step (a), processor 1 sends its data X_1 to processors 2 to N before computing Y_1. Processor 1 can then start with its task for step (b). Similarly, processor 2 sends its data X_2 to processors 3 to N, then computes $A(2,2)X_2$ and as soon as it receives X_1 also $A(2,1)X_1$ to form the sum Y_2 and then starts its step (b). Processors 3 to N proceed similarly. Steps (c) and (2) which contain also products of the form (3.2) are performed in the same way overlapping computation and communication. Once the last processor N has completed its step (2), it then becomes processor 1 for the next time window to save communication. Note also that the processors never need to store the entire space time solution, they only need to compute locally the sums in (3.2) using storage of the size of the vectors X_i to compute Y_i.

In order to measure the performance, we used in this implementation a $\mathbb{P}_1$ finite element discretization for step (b) and solved the linear system in space using the conjugate gradient (CG) method with and without preconditioner, and also the parallel sparse direct solver (PARDISO) [41]. Computations were run on a set of 16 X5670 [Westmere-EP] @ 2.93 GHz processors with a refined mesh, $h = \frac{1}{100}$, and we simulate the process up to $\tilde{T} = 2$ using NWin time windows of length $T = \tilde{T}/\text{NWin}$. The error is measured in the norm defined in the theoretical analysis; see Theorem 2.1. We show in Table 3.1 the optimized parameter ε^* from our analysis given by formula (2.42), and the error due to the time parallelization by diagonalization of the algorithm, which was the same in all experiments, because we used the very small tolerance 10^{-16} for the iterative solvers. We can see in the last column that for 16 and more processors, the error due to parallelization will deteriorate, as predicted by our analysis.

TABLE 3.1
Optimal ε^* and common error to all solvers.

N	**NWin**	ε^*	Error
1	128		4.95×10^{-3}
2	64	0.0098	4.98×10^{-3}
4	32	0.0474	4.98×10^{-3}
8	16	0.135	5.88×10^{-3}
16	8	0.224	1.79×10^{-2}

In Table 3.2, we show the results we obtain using unpreconditioned CG. The first row corresponds to the sequential scheme with a fixed time step $\tilde{k} = \frac{\tilde{T}}{128}$. We then use **NWin** time windows depending on N to get a result over the entire time interval $(0, \tilde{T})$. The column **NIt** gives the range of the number of iterations CG needed to converge. The column **Time RHS** corresponds to steps (1) and (a) in algorithm (3.2), the column **Time CG** to step (b), and the column **Time U, Up** to steps (c) and (2). The efficiency Eff $:= \frac{Time_1}{N \times Time_N}$, where $Time_1$ is the sequential computing time on one processor, and $Time_N$ is the parallel computing time using diagonalization with N processors. Two efficiencies are given in the last two columns: **Solv. Eff** concerns step (b) and **Tot. Eff** concerns the whole time sequence. We see from column **NIt** that as local problems become less well conditioned, the number of iterations increases: we need between 15–33 iterations using one processor, and between 5–149 when using 16 processors, which also affects load balancing. In Table 3.3, we use CG preconditioned with an incomplete LU factorization (ILU0). This improves convergence, but now the cost of an iteration is about twice the cost compared to

the nonpreconditioned case; nevertheless the overall computation time is faster. In Table 3.4, we used a direct solver based on the efficient Intel Math Kernel Library PARDISO [41] that renumbers mesh nodes so that the factorized symmetric matrix is as sparse as possible. We see that PARDISO provides very good local efficiency and is faster than the iterative methods we used in this 2D setting. This good local efficiency however leads to a little less total efficiency than the variant using CG.

TABLE 3.2
2D wave equation using CG without preconditioning, initialization time 0.0007*s.*

N	**NWin**	**NIt**	**Time RHS**	**Time CG**	**Time U, Up**	**Solv. Eff**	**Tot. Eff**
1	128	15-33	0.001 s	0.780 s	0.057 s		
2	64	15-38	0.025 s	0.460 s	0.041 s	84.78 %	79.69 %
4	32	14-47	0.037 s	0.269 s	0.059 s	72.49 %	57.40 %
8	16	10-63	0.045 s	0.160 s	0.139 s	60.94 %	30.45 %
16	8	5-149	0.064 s	0.101 s	0.180 s	48.27 %	15.18 %

TABLE 3.3
*2D wave equation using CG and ILU*0 *preconditioner, initialization time* 0.0039 *s, including ILU*0 *factorization.*

N	**NWin**	**NIt**	**Time RHS**	**Time-CG-ILU**	**Time U, Up**	**Solv. Eff**	**Tot. Eff**
1	128	11-11	0.001 s	0.665 s	0.062 s		
2	64	11-11	0.026 s	0.383 s	0.041 s	86.81 %	80.89 %
4	32	10-11	0.038 s	0.166 s	0.071 s	100.15 %	58.39 %
8	16	7-15	0.051 s	0.088 s	0.119 s	94.46 %	33.09 %
16	8	2-33	0.059 s	0.074 s	0.129 s	56.17 %	17.37 %

TABLE 3.4
2D wave equation using PARDISO, initialization time 0.102 *s including the factorization with PARDISO.*

N	**NWin**	**NIt**	**Time RHS**	**Time Direct Sol.**	**Time U, Up**	**Solv. Eff**	**Tot. Eff**
1	128	1	0.001 s	0.256 s	0.033 s		
2	64	1	0.024 s	0.134 s	0.034 s	95.52 %	64.36 %
4	32	1	0.038 s	0.066 s	0.067 s	96.97 %	37.76 %
8	16	1	0.047 s	0.035 s	0.130 s	91.43 %	17.10 %
16	8	1	0.061 s	0.016 s	0.146 s	99.99 %	8.13 %

These results show that time parallelization by diagonalization is very efficient as long as the local problem is sufficiently computation intensive so as not to be overtaken by communications cost necessary to compute the right hand side (RHS) and the final solution (U) and its first time derivative (Up).

3.3. Application to a large 3D industrial problem. We now apply this technique to an industrial problem: we want to compute the response of a carbon/epoxy laminated composite panel subjected to an impact-like loading. This class of material is modeled in its linear elastic domain with a transverse isotropic Hooke law. When the fiber direction is aligned with the $\mathbf{e}_1$ direction, and using the Voigt notation,[3] the Hooke law is given by

[3]Voigt notation uses a matrix representation for fourth order tensors. It also exploits the symmetries of the second order strain and stress tensors (hence the 6 components).

$$(3.3)\qquad \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{pmatrix} = \begin{pmatrix} \frac{1}{E_L} & -\frac{\nu_{LT}}{E_L} & -\frac{\nu_{LT}}{E_L} & 0 & 0 & 0 \\ -\frac{\nu_{LT}}{E_L} & \frac{1}{E_T} & -\frac{\nu_{TT}}{E_T} & 0 & 0 & 0 \\ -\frac{\nu_{LT}}{E_L} & -\frac{\nu_{TT}}{E_T} & \frac{1}{E_T} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\frac{1+\nu_{TT}}{E_T} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{LT}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{LT}} \end{pmatrix} \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{pmatrix}.$$

Here, E_L is the longitudinal elastic modulus (in the carbon fiber direction), E_T is the transverse elastic modulus, G_{LT} is the elastic shear modulus, and ν_{LT} and ν_{TT} are the so-called Poisson ratios, and we simulate the elasticity equations,

$$\rho\ddot{\mathbf{u}} = \operatorname{div}(\sigma) + \mathbf{f}, \quad \varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

The selected material is a T700GC/M21 carbon/expoxy system used in the aeronautical industry. The corresponding elastic moduli are given in Table 3.5. The geometry of the problem is taken from the experimental work described in [46]. The tested plate is made of 12 plies with a symmetric stacking sequence. The mesh, illustrated in Figure 3.4, consists of 152607 degrees of freedom. Each ply is modeled with one element in the thickness direction. The stacking sequence (i.e., the angle between the fiber orientations and the global $\mathbf{e}_x$ direction for each ply) is taken as [0 / 45 / 90 / -45 / 0 / 0 / 0 / 0 / -45 / 90 / 45 / 0] which implies that the material is heterogeneous at the structure scale. Dirichlet boundary conditions (represented in blue) are applied on areas representative of the experiment. A homogeneous pressure field is applied at the center of the plate (red circle) and is representative of a small impact. The time is discretized with 2000 time steps over the $\tilde{T}$:=10ms simulation range.

TABLE 3.5
Elastic moduli of the T700GC/M21 carbon/epoxy system.

E_L	E_T	ν_{LT}	ν_{TT}	G_{LT}
130.0 GPa	7.7 GPa	0.33	0.4	4750 MPa

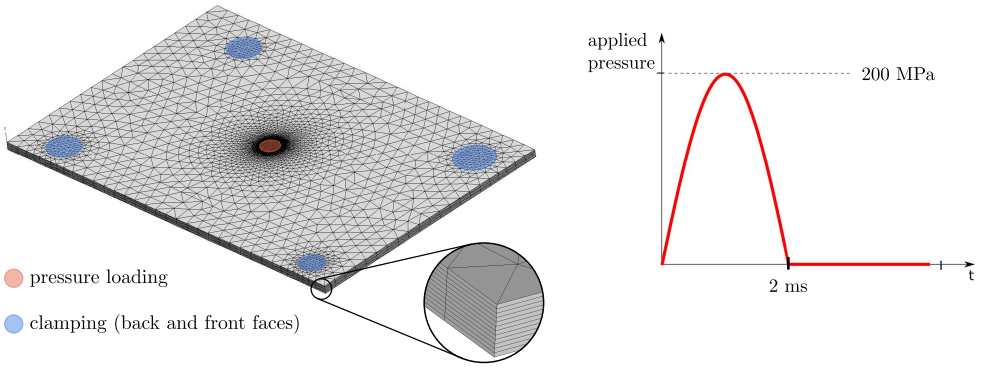

FIG. 3.4. *Mesh configuration and loading for the elasticity problem.*

The finite element code Zset [48] is used to handle the specific finite element features—mesh generation, material orientation, computation of mass and stiffness matrices, boundary conditions, etc. Zset comes with a Python interface which permits to handle the global operators (stiffness and mass matrices) and vectors in a

very convenient way through numpy arrays. All the algebraic operations described in (1.2)–(1.7) are thus performed with the numpy and scipy Python packages. Parallelism is achieved with the mpi4py package, the Python interface to message passing interface (MPI). The script is launched on a 2.4 GHz, Xeon E5-2680 processor with 24 cores, but we use only up to $N = 16$ for our experiments. To reach the 2000 time steps, we thus have to use $2000/N$ time windows. There is an evident gain here in using a direct solver, since once the N sparse operators (see (1.7.b)) have been factorized during the first block, their decomposition can be reused for the solutions in the $2000/N - 1$ remaining time windows. In this implementation, we use the multifrontal massively parallel sparse direct solver [39] and its LU decomposition capability. Because of the multiple time windows, we also have to properly account for the nonzero initial conditions at the beginning of each block. These operations require additional algebraic operations that we refer to as windows initializations (steps (a), (c) and (2) in (3.1)).

We show in Figure 3.5 the measured CPU times when parallelizing with different values of N. The times for the LU factorizations, the LU linear solves by forward and backward substitution, the tensorial operations, the windows initializations, the preprocessing, and the MPI communications are shown separately, and we see that the LU linear solves are fully parallelized in time with this diagonalization technique. Figure 3.5 indeed shows that the corresponding times scale nicely up to $N = 8$ processors, and one obtains substantial speed-up using this method. The overhead due to the different algebraic operations (mainly factorization and tensorial operations) remains about the same from $N = 2$ to $N = 16$ but it becomes more significant in the last case (steps (a), (c) and (2) in (3.1)). From a practical point of view, the 650s gain from a sequential to an $N = 8$ parallel solution seems to be narrow. However, this kind of simulation is typically used to optimize the mechanical behavior according to the stacking sequence. Depending on the optimizer, this process may not be fully parallelizable. When tens or hundreds of runs are required, the gain of using time parallelization then becomes significant. Figure 3.6 shows the displacement of the central node on the back face of the plate for the sequential solution and the parallel solution for $N = 16$. No deviation is noticeable between the two responses, showing that no significant error has been introduced by the parallelization with these parameters.

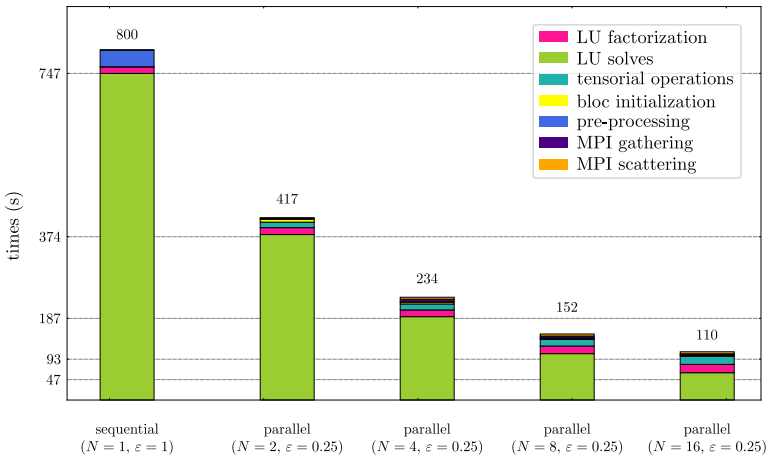

FIG. 3.5. *Computing times for the industrial elasticity problem.*

This type of time parallel algorithm would be well suited to a hybrid MPI/open multiprocessing (OpenMP) approach: MPI for space domain partitioning which can by far exceed shared memory systems and OpenMP for the time parallelisation (less than 20 threads). In terms of memory, there could be a gain (space variables would not need to be replicated) but probably not so in terms of CPU time which is already quite good as shown in the histogram in Figure 3.5 (see MPI gathering and MPI scattering).

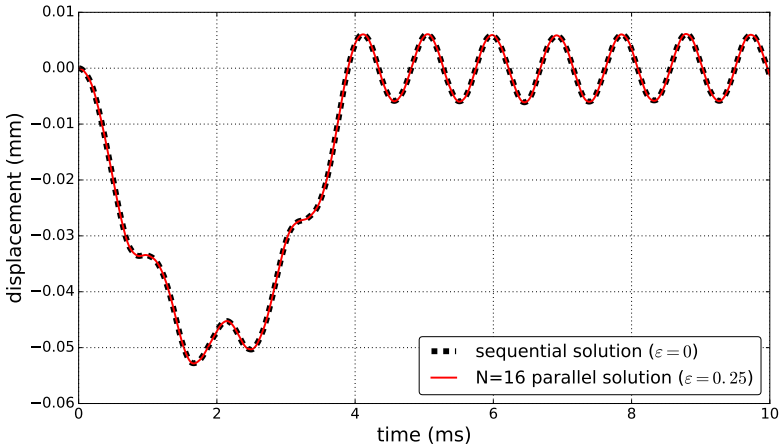

FIG. 3.6. *Deflection of the central node on the back face of the plate for the sequential and the parallel solution with* $N = 16$.

4. Conclusions. We presented and analyzed the time parallelization method by diagonalization for the wave equation. We derived an optimal choice for the geometric stretching of the time grid, balancing carefully truncation and roundoff error. We tested the method numerically for the model wave equation problems in different spatial dimensions and also on an industrial test case using the elasticity equations. These results show that our theoretical parameters predict well the best choice for the geometric time grid stretching, and substantial speedup is possible when solving wave propagation problems using this technique, and this also in an industrial setting.

REFERENCES

[1] D. BENNEQUIN, M. J. GANDER, AND L. HALPERN, *A homographic best approximation problem with application to optimized Schwarz waveform relaxation*, Math. Comp., 78 (2009), pp. 185–223.

[2] T. A. BICKART, *An efficient solution process for implicit Runge-Kutta methods*, SIAM J. Numer. Anal., 14 (1977), pp. 1022–1027.

[3] J. C. BUTCHER, *On the implementation of implicit Runge-Kutta methods*, BIT, 16 (1976), pp. 237–240.

[4] A. J. CHRISTLIEB, C. B. MACDONALD, AND B. W. ONG, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835.

[5] M. EMMETT AND M. L. MINION, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.

[6] R. D. FALGOUT, S. FRIEDHOFF, T. KOLEV, S. P. MACLACHLAN, AND J. B. SCHRODER, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661.

[7] C. FARHAT, J. CORTIAL, C. DASTILLUNG, AND H. BAVESTRELLO, *Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses*, Int. J. Numer. Methods Eng., 67 (2006), pp. 697–724.

[8] S. Friedhoff, R. D. Falgout, T. Kolev, S. P. MacLachlan, and J. B. Schroder, *A multigrid-in-time algorithm for solving evolution equations in parallel*, in Proceedings of the Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, 2013.

[9] M. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, in Domain Decomposition Methods in Science and Engineering XVII, O. B. Widlund and D. E. Keyes, eds., Vol. 60, Springer, New York, 2008, pp. 45–56.

[10] M. Gander, L. Halpern, J. Ryan, and T. Tran, *A direct solver for time parallelization*, in Lecture Notes in Computational Science and Engineering, T. Dickopf, M. Gander, L. Halpern, R. Krause, and L. Pavarino, eds., Vol. XXII, Springer, New York, 2016, pp. 491–499.

[11] M. Gander and C. Rohde, *Overlapping Schwarz waveform relaxation for convection-dominated nonlinear conservation laws*, SIAM J. Sci. Comput., 27 (2005), pp. 415–439.

[12] M. J. Gander, *Analysis of the parareal algorithm applied to hyperbolic problems using characteristics*, SeMA J., 42 (2008), pp. 21–35.

[13] M. J. Gander, *50 years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Springer, New York, 2015, pp. 69–113.

[14] M. J. Gander and S. Güttel, *Paraexp: A parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142.

[15] M. J. Gander and L. Halpern, *Absorbing boundary conditions for the wave equation and parallel computing*, Math. Comp., 74 (2004), pp. 153–176.

[16] M. J. Gander and L. Halpern, *Optimized Schwarz waveform relaxation methods for advection reaction diffusion problems*, SIAM J. Numer. Anal., 45 (2007), pp. 666–697.

[17] M. J. Gander, L. Halpern, and F. Nataf, *Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation*, in Proceedings of the Eleventh international Conference of Domain Decomposition Methods, C.-H. Lai, P. Bjørstad, M. Cross, and O. Widlund, eds., 1999, pp. 27–36.

[18] M. J. Gander, L. Halpern, and F. Nataf, *Optimal Schwarz waveform relaxation for the one dimensional wave equation*, SIAM J. Numer. Anal., 41 (2003), pp. 1643–1681.

[19] M. J. Gander, F. Kwok, and B. Mandal, *Dirichlet-Neumann and Neumann-Neumann waveform relaxation algorithms for parabolic problems*, ETNA, 45 (2016), pp. 424–456.

[20] M. J. Gander, F. Kwok, and B. Mandal, *Dirichlet-Neumann and Neumann-Neumann waveform relaxation for the wave equation*, in Lecture Notes in Computational Science and Engineering, T. Dickopf, M. Gander, L. Halpern, R. Krause, and L. Pavarino, eds., Vol. XXII, Springer, New York, 2016, pp. 501–509.

[21] M. J. Gander, F. Kwok, and H. Zhang, *Multigrid interpretations of the parareal algorithm leading to an overlapping variant and MGRIT*, 2017, to appear.

[22] M. J. Gander and M. Neumüller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM J. Sci. Comp., 38 (2016), pp. A2173–A2208.

[23] M. J. Gander and M. Petcu, *Analysis of a modified parareal algorithm for second-order ordinary differential equations*, in AIP Conference Proceedings, Vol. 936, AIP, 2007, pp. 233–236.

[24] M. J. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, in ESAIM: Proceedings, Vol. 25, EDP Sciences, 2008, pp. 114–129.

[25] M. J. Gander and A. M. Stuart, *Space time continuous analysis of waveform relaxation for the heat equation*, SIAM J. Sci. Comput., 19 (1998), pp. 2014–2031.

[26] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578.

[27] G. Gasper and M. Rahman, *Basic hypergeometric series*, Vol. 96, Cambridge University Press, Cambridge, UK, 2004.

[28] E. Giladi and H. B. Keller, *Space time domain decomposition for parabolic problems*, Numer. Math., 93 (2002), pp. 279–313.

[29] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2013.

[30] J. Gopalakrishnan, J. Schöberl, and C. Wintersteiger, *Mapped tent pitching schemes for hyperbolic systems*, preprint, arXiv:1064.01081 [math.NA], 2016, https://arxiv.org/abs/1064.01081.

[31] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.

[32] F. Kwok, *Neumann–Neumann waveform relaxation for the time-dependent heat equation*, in Domain Decomposition Methods in Science and Engineering XXI, Springer, New York, 2014, pp. 189–198.

[33] J. L. Lions, Y. Maday, and G. Turinici, *A parareal in time discretization of PDE's*, C.R. Acad. Sci. Paris, 332 (2001), pp. 661–668.
[34] C. Lubich and A. Ostermann, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.
[35] Y. Maday and E. M. Rønquist, *Fast tensor product solvers. Part* II: *Spectral discretization in space and time*, Tech. Report 7-9, Laboratoire Jacques-Louis Lions, 2007.
[36] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space-time solvers*, C. R. Math. Acad. Sci. Paris, 346 (2008), pp. 113–118.
[37] B. Mandal, *A time-dependent Dirichlet-Neumann method for the heat equation*, in Lecture Notes in Computational Science and Engineering, J. Erhel, M. Gander, L. Halpern, G. Pichot, T. Sassi, and O. Widlund, eds., Vol. XXI, Springer, New York, 2014, pp. 467–475.
[38] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Commun. Appl. Math. Comput. Sci., 5 (2010), pp. 265–301.
[39] *MUMPS: Multifrontal Massively Parallel Sparse Direct Solver*, MUMPS 5.0.0, 2015, http://mumps-solver.org.
[40] N. M. Newmark, *A method of computation for structural dynamics*, J. Engrg. Mech. Div., 85 (1959), pp. 67–94.
[41] V. Kumar and F. Gennady, *Intel MKL PARDISO*, July 30, 2017, https://software.intel.com/en-us/articles/intel-mkl-pardiso.
[42] A. Qaddouri, L. Laayouni, S. Loisel, J. Côté, and M. Gander, *Optimized Schwarz methods with an overset grid for the shallow-water equations: Preliminary results*, Appl. Numer. Math., 58 (2008), pp. 459–471.
[43] J. Rannou and J. Ryan, *PRF transition statique-dynamique dans les structures composites*, Tech. Report RT 4/18450 DMSM, Onera, Palaiseau, France, 2012.
[44] J. Rannou and J. Ryan, *PRF transition statique-dynamique dans les structures composites*, Tech. Report RT 9/18450 DMSM, Onera, Palaiseau, France, 2013.
[45] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Comput. & Fluids, 59 (2012), pp. 72–83.
[46] E. Trousset, J. Rannou, J.-F. Maire, and L. Guillaumat, *Prediction of low-velocity impact damage on composite plates*, in Proceedings of the 19th DYMAT Technical Meeting-STRASBOURG, 2011.
[47] S. Vandewalle and E. Van de Velde, *Space-time concurrent multigrid waveform relaxation*, Ann. Numer. Math, 1 (1994), pp. 347–363.
[48] *Z-set*, http://www.zset-software.com, ZSET 8.7.1, 2018.