Electronic Transactions on Numerical Analysis.
Volume 51, pp. 135–150, 2019.
Copyright © 2019, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol51s135

ETNA
Kent State University and
Johann Radon Institute (RICAM)

A NOTE ON PARALLEL PRECONDITIONING FOR ALL-AT-ONCE EVOLUTIONARY PDES*

ANTHONY GODDARD† AND ANDY WATHEN†

Abstract. McDonald, Pestana, and Wathen [SIAM J. Sci. Comput., 40 (2018), pp. A1012–A1033] present a method for preconditioning time-dependent PDEs via an approximation by a nearby time-periodic problem, that is, they employ circulant-related matrices as preconditioners for the non-symmetric block Toeplitz matrices which arise from an all-at-once formulation. They suggest that such an approach might be efficiently implemented in parallel. In this short article, we present parallel numerical results for their preconditioner which exhibit strong scaling. We also extend their preconditioner via a Neumann series approach which also allows for efficient parallel execution. Results are shown for both parabolic and hyperbolic PDEs. Our simple implementation (in C++ and MPI) is available at the Git repository `PARALAAOMPI`.[1]

Key words. parallel-in-time, monolithic method, preconditioning

AMS subject classifications. 65M20, 65F08, 65Y05

1. Introduction.

There have been several suggestions to achieve computationally efficient parallel methods for time-dependent problems. Perhaps the most common approaches are based on the Parareal algorithm [9] and its use together with multilevel ideas [3], although there are several other approaches; see the review by Gander [4]. A recent suggestion by McDonald et al. [11] involves the use of circulant-related preconditioners for the block Toeplitz matrices that arise from the approximation of initial value problems with constant time-steps. Their approach is particularly geared to linear initial value problems for PDEs, although it can be applied in the simpler context of ODE IVPs [10]. For PDEs, regularity of the spatial grid is not required. We mention that Gander et al. [5] present a complementary all-at-once method that imposes the restriction that all the time steps be distinct.

The approach requires the solution of block diagonal systems in different orderings so as to effect the action of the preconditioner as we show below. It also allows for an extension involving a Neumann series which we introduce in this paper. Both the extended and the original preconditioners would appear suitable for effective parallel implementation, although such implementation details have not been explored before. That the original preconditioner leads to a small number of iterations, which is independent of the number of time-steps when employed with the widely used GMRES method [12], is established in [11].

In this short article we make a preliminary exploration of the possibilities for effective parallel implementation of these preconditioners. Our initial results using C++ and MPI show strong scaling with up to 32 cores for the preconditioned GMRES solution of the all-at-once (monolithic) system. This system is derived from a spatial finite element approximation and a simple time-stepping strategy in the usual method of lines approach. We explore all-at-once formulations for the heat equation and for the wave equation together with associated initial and boundary conditions. In particular, literature on hyperbolic problems is currently limited, but see [7].

In Section 2 we describe the original McDonald-Pestana-Wathen preconditioner and our extension of it. In Section 3 we provide an analysis of the spectral properties of the matrices associated with the all-at-once formulation of the wave equation. The details of our parallel

*Received August 15, 2018. Accepted April 16, 2019. Published online on June 15, 2019. Recommended by Ken Hayami. The work of A. G. was supported by the James Pantyfedwen Foundation.
†Mathematical Institute, Oxford University, UK (`wathen@maths.ox.ac.uk`).
[1] `https://github.com/anthonyjamesgoddard/PARALAAOMPI`

ETNA
Kent State University and
Johann Radon Institute (RICAM)

implementation are given in Section 4 and numerical (timing) results in Section 5 are followed by conclusions.

We comment that for structured systems other than block Toeplitz, there has been significant previous work on parallel preconditioning. For the important cases of block bi-diagonal and block-banded systems, see, for example, [1, 8, 13, 14].

2. Description of the preconditioners. Consider the heat equation

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} && \text{on } \Omega \times (0,T], \\ u &= 0 && \text{for } \mathbf{x} \in \partial\Omega, \\ u(0,\mathbf{x}) &= s(\mathbf{x}). \end{aligned}$$

Discretising in space with standard Galerkin finite elements, we obtain

$$M\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -K\mathbf{u}, \tag{2.1}$$

where $M, K \in \mathbb{R}^{n\times n}$ are the mass and stiffness matrices and n is the number of nodes in the spatial discretisation. Now we use the implicit Euler scheme to discretise the temporal domain to obtain

$$(K+\tau M)\mathbf{u}_k = M\mathbf{u}_{k-1}, \qquad k \in [1,\ell], \tag{2.2}$$

with τ being the constant time-step, ℓ the number of time-steps in the temporal discretisation, and $\mathbf{u}_0$ is a projection of the initial data. The idea presented in [11] involves packaging the approximate solutions into a so-called monolithic system. Executing this idea yields

$$\mathcal{A}\mathbf{U} = \begin{bmatrix} A_0 & & & \\ A_1 & A_0 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_\ell \end{bmatrix} = \begin{bmatrix} M\mathbf{u}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \mathbf{b},$$

where $A_0 = K+\tau M$, $A_1 = -M$, and $\mathcal{A} \in \mathbb{R}^{n\ell\times n\ell}$. We note that the monolithic system does not need to be formed explicitly; it is merely used as a conduit for demonstration purposes. The McDonald-Pestana-Wathen preconditioner in its original form is the block circulant matrix

$$\mathcal{P} = \begin{bmatrix} A_0 & & & A_1 \\ A_1 & A_0 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0 \end{bmatrix}. \tag{2.3}$$

We can extend the all-at-once method to non-uniform time-stepping schemes. Consider applying the implicit Euler scheme to (2.1) with variable time-steps $\tau_1, \tau_2, ..., \tau_\ell$. Equation (2.2) then becomes

$$(K+\tau_i M)\mathbf{u}_k = M\mathbf{u}_{k-1}, \qquad k \in [1,\ell].$$

We can package this sequence into the monolithic system

$$\mathcal{B}\mathbf{U} = \begin{bmatrix} A_0^1 & & & \\ A_1 & A_0^2 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0^\ell \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_\ell \end{bmatrix} = \begin{bmatrix} M\mathbf{u}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \mathbf{b},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $A_0^i = K + \tau_i M$ and $A_1 = -M$. While we have lost the block Toeplitz structure of the system, this will not prevent us from applying

$$\mathcal{Q} = \begin{bmatrix} A_0^1 & & & A_1 \\ A_1 & A_0^2 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0^\ell \end{bmatrix}$$

as a preconditioner. The issue is that of the computational cost of the application of the preconditioner. Using the standard Kronecker product, we can write

$$\mathcal{Q} = \mathcal{P} + \sigma \otimes K,$$

where σ is a diagonal matrix with diagonal entries given by $\sigma_i = \tau_i - \tau$, $i \in [0, \ell]$, and $\tau = 1/\ell$. Assuming that $||\sigma \otimes K|| < 1$ we can use a Neumann approximation to calculate the inverse of $\mathcal{Q}$ to obtain

$$\begin{aligned} \mathcal{Q}_i^{-1} = \mathcal{P}^{-1} - \mathcal{P}^{-1}([\sigma \otimes K]\mathcal{P}^{-1}) + \mathcal{P}^{-1}([\sigma \otimes K]\mathcal{P}^{-1})^2 + \dots \\ + (-1)^{i-1}\mathcal{P}^{-1}([\sigma \otimes K]\mathcal{P}^{-1})^{i-1}, \end{aligned} \tag{2.4}$$

where i is some positive integer that we must choose.

We can also apply the all-at-once method to hyperbolic equations. We will restrict our scope to the wave equation

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \frac{\partial^2 u}{\partial x^2} && \text{on } \Omega \times (0, T], \\ u &= 0 && \text{for } \mathbf{x} \in \partial\Omega, \\ \frac{\partial u}{\partial t}(0, \mathbf{x}) &= 0, \\ u(0, \mathbf{x}) &= s(\mathbf{x}). \end{aligned}$$

Discretising in space we obtain

$$M\frac{\mathrm{d}^2\mathbf{u}}{\mathrm{d}t^2} = -K\mathbf{u}. \tag{2.5}$$

We can choose the central difference formula to approximate the second time derivative, resulting in the sequence

$$M\mathbf{u}_{n-1} + (\tau^2 K - 2M)\mathbf{u}_n + M\mathbf{u}_{n+1} = \mathbf{0}.$$

Casting this sequence into a monolithic system, we obtain

$$\mathcal{C}_{CD}\mathbf{U} = \begin{bmatrix} A_0 & M & & & \\ M & A_0 & M & & \\ & M & A_0 & \ddots & \\ & & \ddots & \ddots & M \\ & & & M & A_0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \mathbf{u}_\ell \end{bmatrix} = \begin{bmatrix} -M\mathbf{u}_0 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \mathbf{b}_{CD},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $A_0 = \tau^2 K - 2M$ for this problem. We can then precondition this system with the block circulant matrix

$$\mathcal{R}_{CD} = \begin{bmatrix} A_0 & M & & & M \\ M & A_0 & M & & \\ & M & A_0 & \ddots & \\ & & \ddots & \ddots & M \\ M & & & M & A_0 \end{bmatrix}.$$

Alternatively, we can approximate the second time derivative as

$$\frac{\mathrm{d}^2 \mathbf{u}_n}{\mathrm{d}t^2} \approx \frac{\mathbf{u}_{n-2} - 2\mathbf{u}_{n-1} + \mathbf{u}_n}{\tau^2}. \tag{2.6}$$

Expression (2.6) will be referred to as the 2-Step Backwards Difference (BD2) formula, which is a first-order approximation. By substituting (2.6) into (2.5) and rearranging, we obtain

$$Mu_{n-2} - 2Mu_{n-1} + (M + \tau^2 K)u_n = \mathbf{0},$$

which can be compiled into a monolithic system

$$\mathcal{C}_{BD2}\mathbf{U} = \begin{bmatrix} A_0 & & & & \\ A_1 & A_0 & & & \\ A_2 & A_1 & A_0 & & \\ & \ddots & \ddots & \ddots & \\ & & A_2 & A_1 & A_0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \mathbf{u}_\ell \end{bmatrix} = \begin{bmatrix} (M + \tau^2 K)\mathbf{u}_0 \\ -M\mathbf{u}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \mathbf{b}_{BD2},$$

where $A_0 = M + \tau^2 K, A_1 = -2M$, and $A_2 = M$ for the problem arising from the wave equation. We will precondition the above system with

$$\mathcal{R}_{BD2} = \begin{bmatrix} A_0 & & & A_2 & A_1 \\ A_1 & A_0 & & & A_2 \\ A_2 & A_1 & A_0 & & \\ & \ddots & \ddots & \ddots & \\ & & A_2 & A_1 & A_0 \end{bmatrix}.$$

We can also use a second-order backwards difference formula

$$\frac{\mathrm{d}^2 \mathbf{u}_n}{\mathrm{d}t^2} \approx \frac{-\mathbf{u}_{n-3} + 4\mathbf{u}_{n-2} - 5\mathbf{u}_{n-1} + 2\mathbf{u}_n}{\tau^2}, \tag{2.7}$$

which is a second-order method and will be referred to as the 4-Step Backwards Difference (BD4) formula. By substituting (2.7) into (2.5), we obtain

$$(2M + \tau^2 K)\mathbf{u}_n - 5M\mathbf{u}_{n-1} + 4M\mathbf{u}_{n-2} - M\mathbf{u}_{n-3} = \mathbf{0}.$$

As a consequence of using a 4-step method we have to approximate u_1 and u_2 using sub-4-step methods. In this case we can use BD2 and the initial conditions. This results in the monolithic

ETNA
Kent State University and
Johann Radon Institute (RICAM)

system

$$\mathcal{C}_{BD4} = \begin{bmatrix} B & & & & & & & & \\ C & B & & & & & & & \\ A_2 & A_1 & A_0 & & & & & & \\ A_3 & A_2 & A_1 & A_0 & & & & & \\ & A_3 & A_2 & A_1 & A_0 & & & & \\ & & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & & A_3 & A_2 & A_1 & A_0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \\ \mathbf{u}_5 \\ \vdots \\ \vdots \\ \mathbf{u}_\ell \end{bmatrix} = \begin{bmatrix} B\mathbf{u}_0 \\ -M\mathbf{u}_0 \\ M\mathbf{u}_0 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \vdots \\ \mathbf{0} \end{bmatrix} = \mathbf{b}_{BD4},$$

where $B = M+\tau^2 K, C = -2M, A_0 = 2M+\tau^2 K, A_1 = -5M, A_2 = 4M$, and $A_3 = -M$. We precondition this formulation of the wave equation system in a slightly different way. Previously, we have preconditioned the Toeplitz system with its corresponding circulant. In this instance, we do not have a Toeplitz structure to begin with. To overcome this, we precondition the system with the circulant matrix that would result if we had a Toeplitz system. That is, we precondition the system with

$$\mathcal{R}_{BD4} = \begin{bmatrix} A_0 & & & & & & A_3 & A_2 & A_1 \\ A_1 & A_0 & & & & & & A_3 & A_2 \\ A_2 & A_1 & A_0 & & & & & & A_3 \\ A_3 & A_2 & A_1 & A_0 & & & & & \\ & A_3 & A_2 & A_1 & A_0 & & & & \\ & & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & & A_3 & A_2 & A_1 & A_0 \end{bmatrix}.$$

We will not pursue non-uniform temporal domains for this application since there is less interest in using non-uniform time-steps for the wave equation.

3. Preliminary analysis. We begin this section with a theorem that partially describes the spectrum of the preconditioned system associated with the backwards difference formulation of the wave equation, then conclude with a discussion of the ramifications of such a theorem. Related results for the heat equation are given by [11].

Define $E_i = \mathbf{e}_i \otimes \mathbb{I}_n$, where $\mathbf{e}_i$ is a vector of zeros except in the i-th position, which is equal to 1 and $\mathbb{I}_n$ is the $n \times n$ identity matrix.

THEOREM 3.1. *The preconditioned system $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}$ has at least $(\ell-2)n$ eigenvalues equal to 1.*

Proof. We can write $\mathcal{R}_{BD2} = \mathcal{C}_{BD2} + \mathcal{U}\mathcal{W}\mathcal{V}$, where

$$\mathcal{U} = \begin{bmatrix} E_1 & E_2 & E_1 \end{bmatrix}, \quad \mathcal{W} = \begin{bmatrix} A_2 & & \\ & A_2 & \\ & & A_1 \end{bmatrix}, \quad \mathcal{V} = \begin{bmatrix} E_{\ell-1}^T \\ E_\ell^T \\ E_\ell^T \end{bmatrix}.$$

Applying the Sherman-Morrison-Woodbury formula to this expression for $\mathcal{R}_{BD2}$, we obtain

$$\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2} = \mathbb{I}_{nl} - \mathcal{C}_{BD2}^{-1}\mathcal{U}(\mathcal{W}^{-1} + \mathcal{V}\mathcal{C}_{BD2}^{-1}\mathcal{U})^{-1}\mathcal{V}.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

We can write $X = \mathcal{W}^{-1} + \mathcal{V}\mathcal{C}_{BD2}^{-1}\mathcal{U}$ and denote the entries of the $3n$-by-$3n$ matrix X^{-1} by $X_{i,j}^{-1}, i, j \in [1, 3]$. Therefore, we have

(3.1)
$$\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2} = \mathbb{I}_{nl} - \mathcal{C}_{BD2}^{-1} \begin{bmatrix} & X_{1,1}^{-1} + X_{3,1}^{-1} & X_{1,2}^{-1} + X_{1,3}^{-1} + X_{3,2}^{-1} + X_{3,3}^{-1} \\ & X_{2,1}^{-1} & X_{2,2}^{-1} + X_{2,3}^{-1} \\ & & \\ & & \end{bmatrix}.$$

As $\mathcal{C}_{BD2}$ is block Toeplitz and block lower triangular, so is $\mathcal{C}_{BD2}^{-1}$. Therefore we can write the inverse of $\mathcal{C}_{BD2}$ in the form

$$\mathcal{C}_{BD2}^{-1} = \begin{bmatrix} (\mathcal{C}_{BD2}^{-1})_1 & & & \\ (\mathcal{C}_{BD2}^{-1})_2 & (\mathcal{C}_{BD2}^{-1})_1 & & \\ \vdots & & \ddots & \\ (\mathcal{C}_{BD2}^{-1})_\ell & & & (\mathcal{C}_{BD2}^{-1})_1 \end{bmatrix}. \tag{3.2}$$

If we substitute (3.2) into (3.1), then we obtain a matrix of the form

$$\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2} = \begin{bmatrix} \mathbb{I}_n & & D_1 & F_1 \\ & \ddots & \vdots & \vdots \\ & \mathbb{I}_n & D_{\ell-2} & F_{\ell-2} \\ & & \mathbb{I}_n + D_{\ell-1} & F_{\ell-1} \\ & & D_\ell & \mathbb{I}_n + F_\ell \end{bmatrix}, \tag{3.3}$$

where the matrices D_i, F_i are linear combinations of $X_{i,j}^{-1}$ and $(\mathcal{C}_{CD}^{-1})_i$. Using the formula

$$\det \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \det(\mathbf{A}) \det(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbb{I}_n(1-\lambda) & & \\ & \ddots & \\ & & \mathbb{I}_n(1-\lambda) \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} D_1 & F_1 \\ \vdots & \vdots \\ D_{\ell-2} & F_{\ell-2} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}^T, \quad \mathbf{D} = \begin{bmatrix} \mathbb{I}_n(1-\lambda) + D_{\ell-1} & F_{\ell-1} \\ D_\ell & \mathbb{I}_n(1-\lambda) + F_\ell \end{bmatrix},$$

we see that at least $(\ell - 2)n$ eigenvalues of the matrix (3.3) are equal to 1. □

The above theorem shows us that most of the eigenvalues of the preconditioned system associated with the BD2 formulation of the wave equation are equal to 1. Figure 3.1 is a plot of the eigenvalues that are not described by the above theorem. In the case displayed in Figure 3.1, the majority of the remaining eigenvalues are clustered around 1 or are indeed equal to 1.

We can not guarantee that this eigenvalue spectrum will affect the performance of GMRES. It is discussed in [2] that even if a preconditioner exhibits a favourable spectrum (e.g., most

ETNA
Kent State University and
Johann Radon Institute (RICAM)

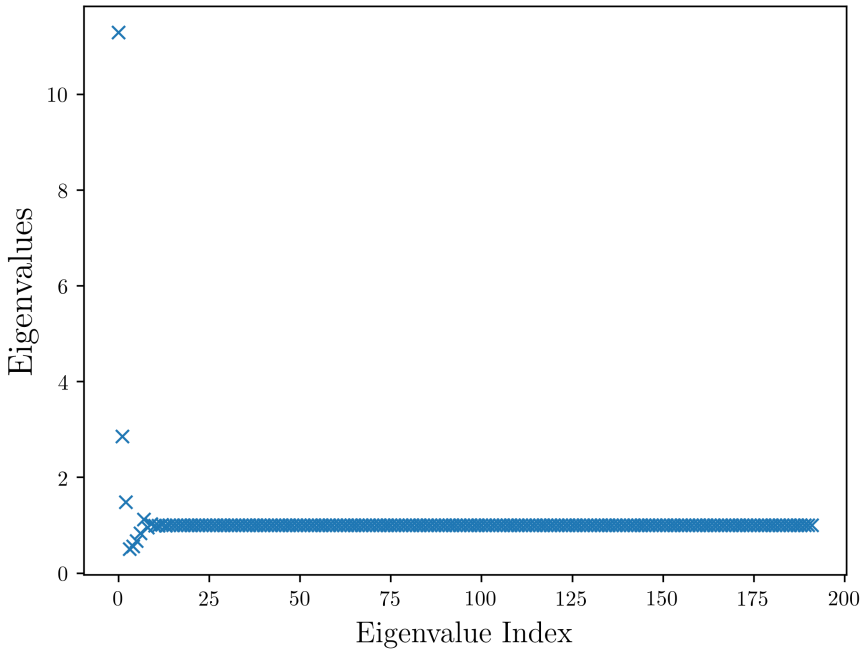

FIG. 3.1. *The first* $2n = 192$ *eigenvalues of* $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}$, $n, m = 96$. *The remaining eigenvalues are equal to 1.*

eigenvalues are clustered around 1) it could be the case that it will perform worse than a preconditioner with a less favourable spectrum. More precise statements of such results can be found, for example, in [6].

Despite this, it is generally observed that, when GMRES is applied to systems with eigenvalue profiles like that of the system associated with the BD2 formulation of the wave equation, GMRES converges within a relatively small number of iterations.

4. Parallel implementation. Throughout our implementation we keep all matrices on the master process and broadcast them when necessary. Vectors, on the other hand, are defined on all processes. To understand the reasoning for this, consider the fact that our dense block U requires $\mathcal{O}(\ell^2)$ memory and our sparse blocks (linear combinations of M and K) each require $\mathcal{O}(n)$ memory. Since there are ℓ sparse blocks, the total memory cost is $\mathcal{O}(\ell^2 + n\ell)$. Further, since a vector requires $\mathcal{O}(n\ell)$ memory, if each one of p processes has a copy of the vector, then we are using $O(n\ell p)$ memory. In the complexity analysis below, we consider the situation where ℓ processes are available. Using ℓ processes would significantly increase the memory requirements of our implementation with this memory management scheme. In our case, $p << \ell$, and so the vector storage cost almost matches that of the matrix storage cost. Even in the case where $p \sim \ell$, the reduction in communication cost that results from this type of memory management is likely to be significant.

In order to see how (2.3) can be applied in parallel we follow [11] in writing it in the form

$$\mathcal{P} = \mathbb{I}_\ell \otimes A_0 + \Sigma \otimes A_1, \tag{4.1}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where

$$\Sigma = \begin{bmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The key property of circulant matrices is that they can be unitarily diagonalised by a Fourier basis. That is, we can write $\Sigma = U \Lambda U^*$, where $U_{k,j} = e^{((k-1)(j-1)\pi i)/n}/\sqrt{n}$ and the diagonal entries of Λ are the roots of unity for the "downshift" matrix Σ. Hence, again following [11], (4.1) can be written as

$$\mathcal{P} = (U \otimes \mathbb{I}_n)[\mathbb{I}_\ell \otimes A_0 + \Lambda \otimes A_1](U^* \otimes \mathbb{I}_n).$$

Inverting $\mathcal{P}$, we obtain

$$\mathcal{P}^{-1} = (U \otimes \mathbb{I}_n)[\mathbb{I}_\ell \otimes A_0 + \Lambda \otimes A_1]^{-1}(U^* \otimes \mathbb{I}_n).$$

The application of $(U \otimes \mathbb{I}_n)$ to a vector can be carried out in parallel. To see this, consider the explicit representation of $(U \otimes \mathbb{I}_n)$ given by

$$(U \otimes \mathbb{I}_n) = \begin{bmatrix} U_{11}\mathbb{I}_n & \dots & U_{1\ell}\mathbb{I}_n \\ \vdots & \ddots & \vdots \\ U_{\ell 1}\mathbb{I}_n & \dots & U_{\ell\ell}\mathbb{I}_n \end{bmatrix}.$$

First, we broadcast each row of U to a process. Then we can evaluate

$$\mathbf{y}_i = \begin{bmatrix} U_{i1}\mathbb{I}_n & \dots & U_{i\ell}\mathbb{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_\ell \end{bmatrix}$$

on each process, where $\mathbf{z}_i \in \mathbb{R}^n$ is a chunk of the $n\ell$-vector $\mathbf{z}$ and i is an integer such that $i \in [1, \ell]$. This can be done in $\mathcal{O}(n\ell)$. The local resultants of each of the calculations carried out on each process $\mathbf{y}_i$ can then be reduced to yield the final resultant of the calculation $(U \otimes \mathbb{I}_n)\mathbf{z}$.

The only other implementation-specific issue that we need to be concerned with is the inversion of the block-tridiagonal matrix $\mathbb{I}_\ell \otimes A_0 + \Lambda \otimes A_1$. This can easily be carried out in parallel by assigning each tridiagonal block to a process and then applying the Thomas algorithm to each block, which would incur a cost of $\mathcal{O}(n)$ over ℓ processes. Therefore, the total complexity of this implementation is $\mathcal{O}(n\ell)$ over ℓ processes. As in [11], multilevel iterations can be applied as approximate solvers for the spatial operators when there is more than one spatial dimension.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

An alternative method involves the use of the Fast Fourier Transform (FFT). In order to see how the FFT can be applied in this case, we can write

$$\begin{aligned}(U \otimes \mathbb{I}_n)\mathbf{z} &= \begin{bmatrix} U_{11}\mathbb{I}_n & \dots & U_{1\ell}\mathbb{I}_n \\ \vdots & \ddots & \vdots \\ U_{\ell 1}\mathbb{I}_n & \dots & U_{\ell\ell}\mathbb{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_\ell \end{bmatrix} \\ &= \begin{bmatrix} U & & \\ & \ddots & \\ & & U \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \\ &= (\mathbb{I}_n \otimes U)\mathbf{x},\end{aligned}$$

where $(\mathbf{x}_i)_k = (\mathbf{z}_k)_i$, $\mathbf{x}_i \in \mathbb{R}^\ell$, for $i \in [1, n]$, are the chunks of $\mathbf{x}$, and $k \in [1, \ell]$. This transformation is called a vector transpose and can be carried out in $\mathcal{O}(n)$ over ℓ processes.[2] Therefore, we can form

$$\mathbf{y}_i = U\mathbf{x}_i,$$

where i is an integer such that $i \in [1, n]$. According to [15], the cost of applying U to a vector using the FFT is $\mathcal{O}(\ell \log \ell)$. Before we can progress with the calculation, we will have to apply the vector transpose again to $[\mathbf{y}_1, ..., \mathbf{y}_n]^T$. Consequently, the complexity of this implementation now becomes $\mathcal{O}(\ell \log \ell + n^2)$ over $\max(n, \ell)$ processes.

The timing results in the next section were obtained using the first method of evaluating $(U \otimes \mathbb{I}_n)$, not the FFT method. The reason for this is that the extra communication required to perform the transpose operator multiple times significantly reduced the performance of the all-at-once implementation. However, the functionality to implement both routines is provided in the GitHub repository.

From an inspection of the operations we can see that the most expensive component of a GMRES iteration is the application of the preconditioner. If we consider applying the above ideology to $\mathcal{Q}_i^{-1}\mathcal{B}\mathbf{U} = \mathcal{Q}_i^{-1}\mathbf{b}$, then for us to increase i from 1 to 2 we have to take into account the cost of one extra preconditioner application as well as communication. That is, for us to consider the Neumann method effective, we must expect the number of GMRES iterations that are needed to solve the problem to reduce by more than half.

5. Numerical results. All parallel results were generated on the nightcrawler workstation at Oxford University. This machine is equipped with 2×18 core Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz processors, 768GB RAM, and 4600GB in scratch disk capacity. Care was taken to access the machine when the workload was low so as to maintain consistent results.

In [11] we see that the idea of preconditioning a block Toeplitz system with a block circulant matrix yields very good theoretical results. Table 1 of [11] shows that few iterations are required when computing the solution of such preconditioned Toeplitz systems. It was suggested in [11] that the all-at-once method can be executed in parallel. Timing results for the heat equation on a uniform temporal domain with initial condition

$$u_0^{s1}(x) = x(1 - x)$$

are given in Table 5.1.

[2] See `VectorTranspose` of `ParallelRoutines.cpp` in the Git repository.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.1
Timed results (in seconds) for solving the system $\mathcal{P}^{-1}\mathcal{A}\mathbf{U} = \mathcal{P}^{-1}\mathbf{b}$ using GMRES with tolerance set to 10^{-5}. The iteration count remained at a constant value of 2 for all values of n and ℓ tested. p is the number of processes used in the calculations.

		$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$
$\ell = 768$	$n = 320$	77.72	29.26	15.32	8.95	5.11	3.34
	$n = 512$	152.64	57.54	32.71	17.52	11.54	6.69
	$n = 768$	245.47	97.77	50.81	30.71	16.66	9.65
$\ell = 1024$	$n = 320$	146.67	54.68	28.40	17.059	10.35	6.07
	$n = 512$	265.22	107.07	60.86	34.13	20.40	11.75
	$n = 768$	459.12	198.94	101.23	55.85	28.55	16.12
$\ell = 1440$	$n = 320$	325.14	124.67	63.64	39.78	22.74	13.06
	$n = 512$	646.81	239.65	123.44	72.44	40.95	22.50
	$n = 768$	979.85	432.46	215.77	114.99	59.80	32.41
$\ell = 1440$	$n = 1568$	2119.91	815.93	431.13	218.24	118.62	63.30

Referring to Table 5.1, we can see that increasing the number of processes from 1 to 32 results in a significant reduction in the time taken to solve the preconditioned system associated with the heat equation. The most significant speed-up is achieved when $\ell = 1440$ and $n = 1568$: the time taken to solve the system reduces from ~ 35 minutes to ~ 1 minute. Observe that, for all values of n and ℓ, the time taken to solve the problem reduces by more than half as a result of increasing p from 1 to 2. We suspect that this is because half of the problem better fits in local memory than the entire problem does.

While the data presented in Table 5.1 is very useful for highlighting the speed-up achieved by distributing the calculation across multiple processes, it would be desirable to quantify how efficiently the processes are being used. To see this, we consider the parallel efficiency of p processes defined by

$$P_{eff}^{p} = \frac{\text{Time Taken on 1 Process}}{p \times \text{Time Taken on } p \text{ Processes}}.$$

The parallel efficiency results are shown in Figure 5.1. The suspected reason for the jump in going from P_{eff}^{1} to P_{eff}^{2} is that, as noted above, the problem better fits in local memory over two processes. The parallel efficiency falls as we increase the number of processes used in the calculation, which is to be expected. This is a consequence of the number of communications taking place between processes. We note that, as we increase the number of degrees of freedom, P_{eff}^{32} also increases, particularly for the values $n = 1440$ and $\ell = 1568$, $P_{eff}^{32} > 1$. That is, our metric for measuring parallel efficiency implies that the all-at-once implementation is more efficient on 32 processes than it is on a single process.

In Section 2 we have introduced an extension to the all-at-once method that enables us to consider problems that are non-uniformly discretised in time. The key behind the extension is the Neumann approximation of the preconditioner $\mathcal{Q}^{-1}$ given by (2.4). The non-uniform temporal discretisation that was used to obtain the numerical results associated with the system $\mathcal{Q}_i^{-1}\mathcal{B}\mathbf{U} = \mathcal{Q}_i^{-1}\mathbf{b}$ is given by

$$t_j = \begin{cases} 0, & j = 0, \\ \frac{1}{n}(j + \delta(\texttt{Rand(0,1,j)} - 0.5)), & j \in [1, \ell - 1], \\ 1, & j = \ell, \end{cases}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

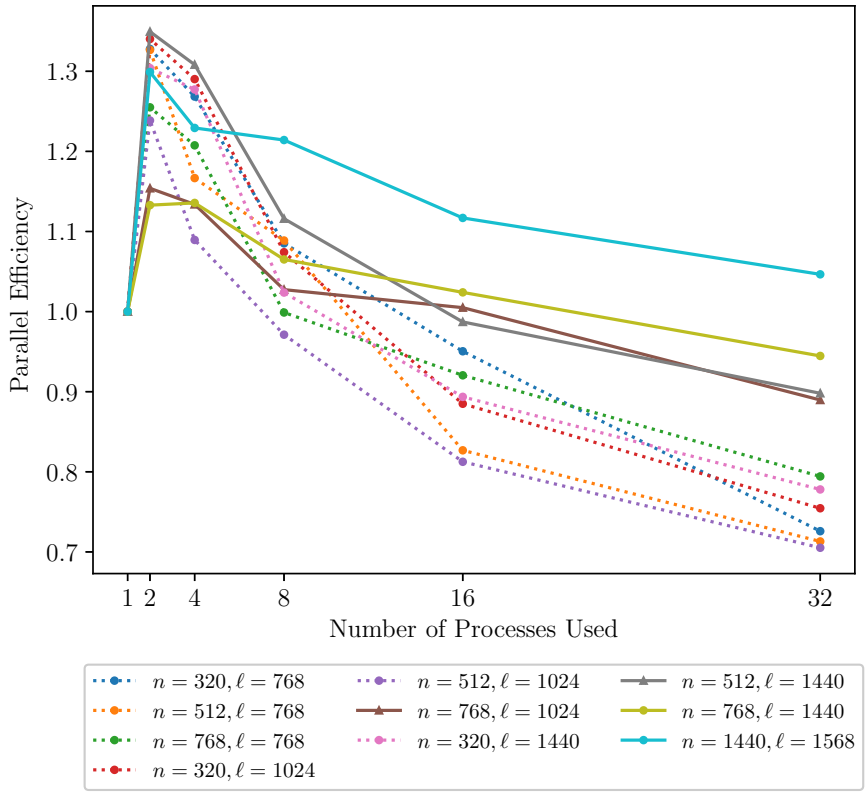

FIG. 5.1. *The parallel efficiency of our implementation of GMRES used to solve the all-at-once formulation of the preconditioned heat equation system* $\mathcal{P}^{-1}\mathcal{A}\mathbf{U} = \mathcal{P}^{-1}\mathbf{b}$.

where j is an integer, δ is a real number such that $\delta \in (0, 1)$, and `Rand(0,1,j)` is a random real number between 0 and 1. Larger values of δ clearly tend to give more irregular time-steps $\tau_j = t_j - t_{j-1}$. Table 5.2 displays the GMRES iteration counts required to solve $\mathcal{Q}_i^{-1}\mathcal{B}\mathbf{U} = \mathcal{Q}_i^{-1}\mathbf{b}$ for increasing values of i. We also display the difference between the time taken to solve the problem with $i = 2$ and the time taken to solve the problem with $i = 1$, $\Delta_{2,1}$. That is, a positive value of $\Delta_{2,1}$ indicates that it took longer to solve the problem with $i = 2$ than it did with $i = 1$.

Firstly, $\Delta_{2,1} > 0$ for all values of n, ℓ, and δ tested. This somewhat validates the claim we made in the previous section according to which the iteration count would need to reduce by more than half for any improvements to be observed in the timed results. For $n = 768, \ell = 768$, and $\delta = 0.7$, the iteration count reduced by a half as a result of increasing i from 1 to 2; in this case $\Delta_{2,1} = 21.68$, which is quite significant. On a positive note, Table 5.2 highlights the robustness of the all-at-once formulation in the presence of temporal perturbations. For instance, increasing δ from 0.1 to 0.9 increases the iteration count by 2 for all values of n and ℓ tested. Increasing i beyond 2 does halve the iteration count in some cases, but since the cost incurred by communication is going to be increased further, there will be no advantage in doing so.

In Section 2 we have introduced an all-at-once formulation of the wave equation. Three formulas were considered as candidates to approximate the time derivative. The central

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.2
GMRES iteration count for $\mathcal{Q}_i^{-1}\mathcal{B}\mathbf{U} = \mathcal{Q}_i^{-1}\mathbf{b}$. The tolerance was set to 10^{-5}. The values of n and ℓ for each table are given as (a) $n = 320, \ell = 768$, (b) $n = 512, \ell = 768$, (c) $n = 768, \ell = 768$, (d) $n = 512, \ell = 1024$, (e) $n = 768, \ell = 1024$, (f) $n = 1024, \ell = 1024$. $\Delta_{2,1}$ is the difference between the time taken to solve the problem with $i = 2$ and the time taken to solve the problem using $i = 1$. In this particular experiment, 16 processes were utilised.

(a)	$i=1$	$i=2$	$i=3$	$\Delta_{2,1}$	(b)	$i=1$	$i=2$	$i=3$	$\Delta_{2,1}$
$\delta = 0.9$	6	4	3	9.32	$\delta = 0.9$	6	5	3	25.28
$\delta = 0.8$	6	4	2	9.36	$\delta = 0.8$	6	5	2	29.73
$\delta = 0.7$	4	4	2	12.83	$\delta = 0.7$	4	4	2	20.83
$\delta = 0.6$	4	4	2	12.38	$\delta = 0.6$	4	3	2	21.38
$\delta = 0.5$	4	4	2	11.75	$\delta = 0.5$	4	3	2	16.75
$\delta = 0.4$	4	4	2	10.33	$\delta = 0.4$	4	3	2	15.33
$\delta = 0.3$	4	3	2	7.90	$\delta = 0.3$	4	3	2	14.90
$\delta = 0.2$	4	3	2	5.61	$\delta = 0.2$	4	3	2	15.61
$\delta = 0.1$	4	3	2	4.32	$\delta = 0.1$	4	3	2	15.32

(c)	$i=1$	$i=2$	$i=3$	$\Delta_{2,1}$	(d)	$i=1$	$i=2$	$i=3$	$\Delta_{2,1}$
$\delta = 0.9$	6	4	3	34.13	$\delta = 0.9$	6	5	3	62.65
$\delta = 0.8$	6	4	2	33.75	$\delta = 0.8$	6	4	3	51.10
$\delta = 0.7$	6	3	2	21.68	$\delta = 0.7$	4	4	3	59.12
$\delta = 0.6$	4	3	2	31.87	$\delta = 0.6$	4	4	3	58.01
$\delta = 0.5$	4	3	2	33.43	$\delta = 0.5$	4	3	2	43.71
$\delta = 0.4$	4	3	2	32.42	$\delta = 0.4$	4	3	2	42.33
$\delta = 0.3$	4	3	2	31.31	$\delta = 0.3$	4	3	2	43.51
$\delta = 0.2$	4	3	2	31.91	$\delta = 0.2$	4	3	2	42.51
$\delta = 0.1$	4	3	2	32.36	$\delta = 0.1$	4	3	2	44.32

(e)	$i=1$	$i=2$	$i=3$	$\Delta_{2,1}$	(f)	$i=1$	$i=2$	$i=3$	$\Delta_{2,1}$
$\delta = 0.9$	6	4	3	59.11	$\delta = 0.9$	6	4	3	82.11
$\delta = 0.8$	6	4	3	58.12	$\delta = 0.8$	4	4	3	103.12
$\delta = 0.7$	4	3	3	63.22	$\delta = 0.7$	4	4	2	102.18
$\delta = 0.6$	4	3	2	65.26	$\delta = 0.6$	4	3	2	80.31
$\delta = 0.5$	4	3	2	63.31	$\delta = 0.5$	4	3	2	75.42
$\delta = 0.4$	4	3	2	64.23	$\delta = 0.4$	4	3	2	74.15
$\delta = 0.3$	4	3	2	67.53	$\delta = 0.3$	4	3	2	78.13
$\delta = 0.2$	4	3	2	64.21	$\delta = 0.2$	4	3	2	74.21
$\delta = 0.1$	4	3	2	65.26	$\delta = 0.1$	4	3	2	75.26

differences formulation performed poorly by selecting the smooth initial condition

$$u_0^{s2} = \sin(2\pi x),$$

and GMRES failed to converge to a solution by selecting the non-smooth initial condition

$$u_0^{ns}(x) = \begin{cases} \cos^2 4\pi\big(x - \frac{1}{2}\big), & x \in \big(\frac{3}{8}, \frac{5}{8}\big), \\ 0, & x \in [0,1]\backslash\big(\frac{3}{8}, \frac{5}{8}\big), \end{cases}$$

when applied to the system $\mathcal{R}_{CD}^{-1}\mathcal{C}_{CD}\mathbf{U} = \mathcal{R}_{CD}^{-1}\mathbf{b}_{CD}$. Central differences performed poorly on u_0^{s2} in the sense that the solution displays aggressive numerical dissipation regardless of how large n and ℓ are chosen to be. However, the number of GMRES iterations required to

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.3
GMRES iteration counts k required to solve the wave equation system. v is the wave speed of the approximate solution. The wave speed of the exact solution is 1. Tolerance was set to 10^{-5}. (a) $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}\mathbf{U} = \mathcal{R}_{BD2}^{-1}\mathbf{b}_{BD2}$, (b) $\mathcal{R}_{BD4}^{-1}\mathcal{C}_{BD4}\mathbf{U} = \mathcal{R}_{BD4}^{-1}\mathbf{b}_{BD4}$.

(a)		k	v	(b)		k	v
	$n = 32$	5	1.03		$n = 32$	60	1.03
$\ell = 32$	$n = 64$	5	1.01	$\ell = 32$	$n = 64$	100	1.01
	$n = 96$	5	1.01		$n = 96$	180	0.67
	$n = 32$	6	2.06		$n = 32$	80	1.03
$\ell = 64$	$n = 64$	6	1.01	$\ell = 64$	$n = 64$	120	1.01
	$n = 96$	6	1.34		$n = 96$	200	0.67
	$n = 32$	6	3.09		$n = 32$	140	3.10
$\ell = 96$	$n = 64$	8	1.52	$\ell = 96$	$n = 64$	180	1.52
	$n = 96$	6	1.01		$n = 96$	9216	1.01

TABLE 5.4
GMRES iteration counts k required to solve $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}\mathbf{U} = \mathcal{R}_{BD2}^{-1}\mathbf{b}_{BD2}$ for larger values of n and ℓ. Tolerance was set to 10^{-5}.

		k
	$n = 320$	8
$\ell = 768$	$n = 512$	8
	$n = 768$	8
	$n = 320$	8
$\ell = 1024$	$n = 512$	8
	$n = 768$	8
	$n = 320$	8
$\ell = 1440$	$n = 512$	8
	$n = 768$	8

solve the system $\mathcal{R}_{CD}^{-1}\mathcal{C}_{CD}\mathbf{U} = \mathcal{R}_{CD}^{-1}\mathbf{b}_{CD}$ with the smooth initial condition remained at 2 for all values of n and ℓ tested. This sensitivity to initial conditions may be related to the spectrum of the matrix associated with the CD formulation of the wave equation, which was discussed in Section 3. For these reasons, the central differences formulation will not be considered further.

Iteration counts for BD2 and BD4 are provided in Table 5.3 and Table 5.4. Plots of the solution of the wave equation, obtained using the corresponding all-at-once formulations, are displayed in Figure 5.2. All of these results were obtained with the non-smooth initial condition u_0^{ns}. Iteration counts for BD2 are low and the wave speeds are largely conserved. The drawback of BD2 is the introduction of numerical dissipation in the solution, as seen in Figure 5.2 (a), although the dissipation is very subtle compared to that observed in the solution of the central differences formulation. BD4 rectifies this drawback, but iteration counts are much higher. Similarly to the central difference approximation, when the smooth initial condition was used, the number of GMRES iterations required to solve the problem remained fixed at 2 for both BD2 and BD4 formulations. Our observations indicate that the all-at-once method formulation of the wave equation is sensitive to the choice of initial conditions.

Timed results for the solution of the wave equation using the all-at-once formulation are provided in Table 5.5 and parallel efficiency results are displayed in Figure 5.3. The system associated with the wave equation resulting from using BD2 is less sparse than the one

ETNA
Kent State University and
Johann Radon Institute (RICAM)

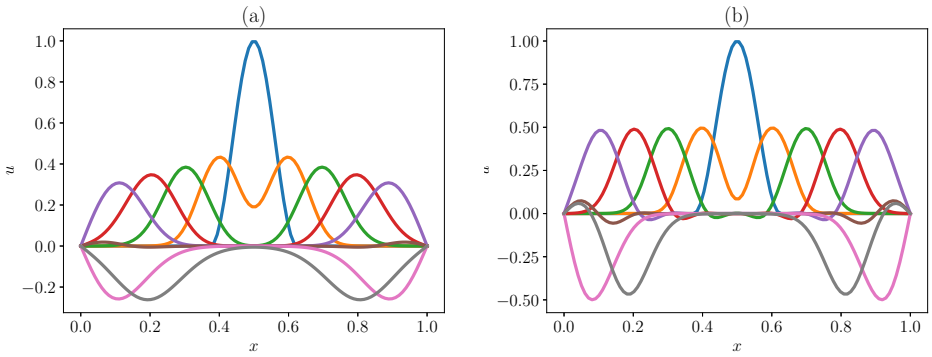

FIG. 5.2. *Solution profiles for the wave equation. The profiles were taken at equally spaced time intervals and the non-smooth initial condition u_0^{ns} was used. $n = \ell = 128$. (a) $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}\mathbf{U} = \mathcal{R}_{BD2}^{-1}\mathbf{b}_{BD2}$ (b) $\mathcal{R}_{BD4}^{-1}\mathcal{C}_{BD4}\mathbf{U} = \mathcal{R}_{BD4}^{-1}\mathbf{b}_{BD4}$.*

TABLE 5.5
The timed results (in seconds) for solving the system $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}\mathbf{U} = \mathcal{R}_{BD2}^{-1}\mathbf{b}_{BD2}$ using GMRES with tolerance set to 10^{-5}. The iteration count remained at a constant value of 2 for all values of n and ℓ tested. p is the number of processes used in the calculations. The smooth initial condition u_0^{s2} was used.

		$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$
$\ell = 768$	$n = 320$	79.07	31.29	16.20	9.53	6.11	4.36
	$n = 512$	163.68	61.33	34.33	19.76	11.54	7.09
	$n = 768$	251.37	100.99	53.14	27.31	14.64	11.09
$\ell = 1024$	$n = 320$	153.37	53.39	30.47	20.99	12.38	7.39
	$n = 512$	287.23	119.72	65.84	39.81	23.23	12.08
	$n = 768$	497.12	222.93	115.24	60.84	32.46	18.01
$\ell = 1440$	$n = 320$	328.17	125.71	65.64	41.70	23.32	14.21
	$n = 512$	680.15	243.53	124.65	73.92	41.42	24.51
	$n = 768$	960.33	434.46	211.01	115.95	60.54	35.12
$\ell = 1440$	$n = 1568$	2211.97	820.21	444.31	230.42	122.63	68.10

associated with the heat equation, and so we would expect the computations to take longer. Also, note that we have a reduction in parallel efficiency for all values of n and ℓ tested. However, the parallel efficiency does seem to increase with the number of degrees of freedom. This trend was observed in the parallel efficiency results for the heat equation.

6. Conclusions. We have provided a parallel implementation of the all-at-once method of [11]. This was achieved using MPI and C++ and is a proof-of-concept software that supports the claims made in [11], namely, that the all-at-once method with the McDonald et al. preconditioner is parallelisable. We have also provided new applications for the all-at-once method, namely, applications to non-uniform temporal discretisation and to hyperbolic equations. Problems in one spatial dimension were considered throughout. To apply the all-at-once method to higher-dimensional problems, some alterations to the implementation provided in this paper are required. An alternative suggested in [11] would be to apply Algebraic Multigrid in parallel instead of the parallel Thomas algorithm for the blocks representing spatial approximation.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

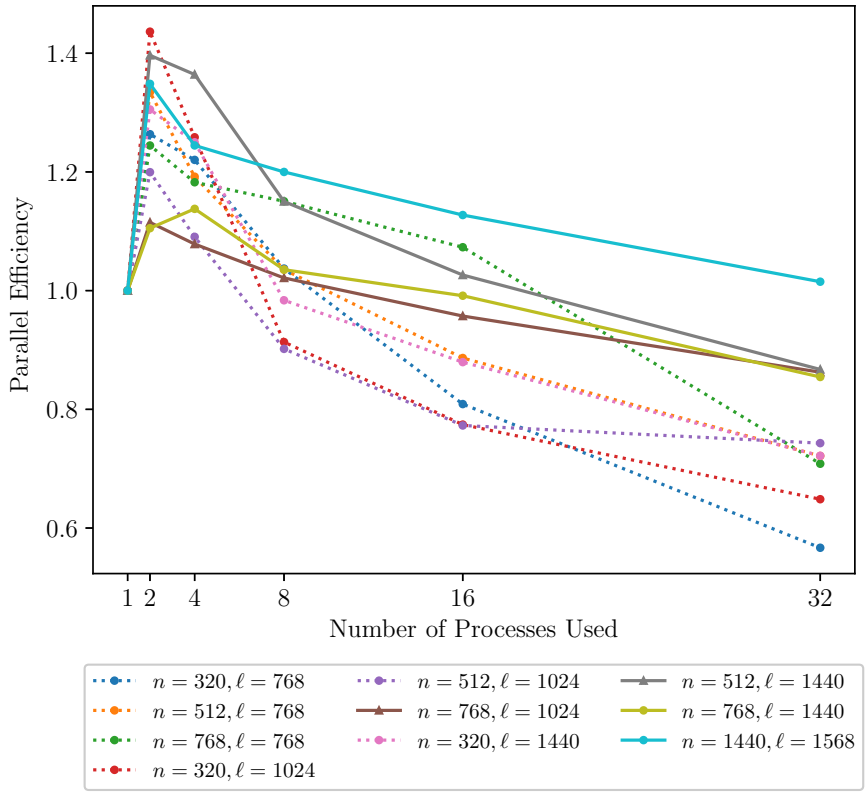

FIG. 5.3. *The parallel efficiency of our implementation of GMRES used to solve the all-at-once formulation of the preconditioned wave equation system* $\mathcal{R}_{BD2}^{-1}\mathcal{C}_{BD2}\mathbf{U} = \mathcal{R}_{BD2}^{-1}\mathbf{b}_{BD2}$. *The smooth initial condition* u_0^{s2} *was used.*

REFERENCES

[1] O. AXELSSON AND J. KARÁTSON, *Preconditioning of block tridiagonal matrices*, Oberwolfach Preprints 2008-05, Oberwolfach, 2008. DOI: 10.14760/OWP-2008-05.

[2] I. DUFF AND H. VAN DER VORST, *Preconditioning and parallel Preconditioning*, Tech. Report TR/PA/98/23, CERFACS, Toulouse, 1998.

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

[4] 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. Rannachev, eds., vol. 9 of Contrib. Math. Comput. Sci., Springer, Cham, 2015, pp. 69–113.

[5] M. J. GANDER, L. HALPERN, J. RYAN, AND T. T. B. TRAN, *A direct solver for time parallelization*, in Domain Decomposition Methods in Science and Engineering XXII, T. Dickopf, M. J. Gander, L. Halpern, R. Krause and L. F. Pavarino, eds., vol. 104 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2016, pp. 491–499.

[6] A. GREENBAUM, V. PTÁK, AND Z. STRAKOŠ, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[7] A. J. HOWSE, H. DE STERCK, R. D. FALGOUT, S. MACLACHLAN, AND J. SCHRODER, *Parallel-in-time multigrid with adaptive spatial coarsening for the linear advection and inviscid Burgers equations*, SIAM J. Sci. Comput., 41 (2019), pp. A538–A565.

[8] M. H. KOULAEI AND F. TOUTOUNIAN, *On computing of block ILU preconditioner for block tridiagonal systems*, J. Comput. Appl. Math., 202 (2007), pp. 248–257.

[9] J.-L. LIONS, Y. MADAY, AND G. TURINICI, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[10] E. MCDONALD, S. HON, J. PESTANA, AND A. WATHEN, *Preconditioning for nonsymmetry and time-dependence* in Domain Decomposition Methods in Science and Engineering XXIII, C.-O. Lee, X.-C. Cai, D. E. Keyes, H. H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, eds., Lect. Notes Comput. Sci. Eng. 116, Springer, Cham, 2017, pp. 81–91.

[11] E. MCDONALD, J. PESTANA, AND A. WATHEN, *Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations*, SIAM J. Sci. Comput., 40 (2018), pp. A1012–A1033.

[12] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[13] D. K. SALKUYEH, *On the preconditioning of the block tridiagonal linear system of equations*, J. Appl. Math. Comput., 28 (2008), pp. 133–146.

[14] H. A. VAN DER VORST, *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45–54.

[15] C. VAN LOAN, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.