ExaQUte

Exascale Quantification of Uncertainties for Technology and Science Simulation

D3.2 Report on parallel in time methods and release of the solvers

Document information table

Contract number:	800898
Project acronym:	ExaQUte
Project Coordinator:	CIMNE
Document Responsible Partner:	UPC
Deliverable Type:	Report, Other
Dissemination Level:	Public
Related WP & Task:	WP3, Task 3.2
Status:	Draft Version

Authoring

Prepared by:				
Authors	Partner	Modified Page/Sections	Version	Comments
Manuel Caicedo	UPC			
Javier Principe	UPC			
Contributors				
Ramon Codina	UPC			

Change Log

Versions	Modified Page/Sections	Comments

Approval

Approved by:				
	Name	Partner	Date	OK
Task leader	Santiago Badia	CIMNE	29/05/2020	Yes
WP leader	Javier Principe	UPC	29/05/2020	Yes
Coordinator	Javier Principe	UPC	29/05/2020	Yes

Table of contents

List of Figures

Nomenclature / Acronym list

Acronym	Meaning
ExaQUte	EXAscale Quantification of Uncertainties for Technology and Science Simulation
CFL	Courant Friedrichs Lewy
BDDC	Balancing Domain Decomposition by Constraints
STBDDC	Space Time Balancing Domain Decomposition by Constraints
HPC	High performance computing
PDE	Partial differential equation
MN-IV	Marenostrum-IV
FE	finite elements
MPI	Message Passing Interface

1 Introduction

In this deliverable we provide the details related to the design, implementation, and scalability analysis of Space Time Balancing Domain Decomposition by Constraints (STBDDC) preconditioners that have been implemented in the `FEMPAR` project [8]. First, we describe the state of the art of space-time methods in Sect. 2 and we then provide some details of our particular implementation in Sect. 3. Next, in Sect. 4, we present a detailed description of the numerical experiments performed during the project showing the excellent scalability results that these algorithms permit to achieve. At the same time, we show the limitations of these algorithms when dealing with nonlinear problems. Finally we draw some concluding remarks in Sect. 5.

2 Parallel in time and space-time methods

In this section we briefly summarize the state of the art in parallel-in-time and space-time methods.

The traditional approach to perform the temporal discretization of time-dependent problems is based on a finite difference approximation in time by time stepping. The simulation starts from the given initial condition and proceeds computing the solution at the current time step using information of the previous ones. High order methods require many previous steps and implicit methods include the current step in the computation of temporal derivatives whereas explicit ones do not. Implicit methods require the solution of a linear system of equations whereas explicit ones only the update of the unknown at the current step. By construction, this procedure is sequential in time and parallelism can only be exploited in space.

There are therefore two problems with this class of methods. On the one hand, when the spatial size of the problem is fixed, i.e. the finite element mesh is already sufficiently refined, opportunities for exploiting parallelism are exhausted: at some point the number of processors cannot be increased because the amount of work to distribute would be smaller than the one required to do the distribution (communications would dominate). At this point the wall-clock time of the simulation will increase with the length of the time domain.

On the other hand, weak scalability if not possible for this type of methods. Indeed, if the mesh size h is reduced, enlarging the size of the spatial discretization, as it is done in a weak scaling experiment, the time step size δt has to be reduced accordingly. This reduction depends on the method used and the differential equation considered. Explicit methods have need to strictly satisfy a Courant Friedrichs Lewy (CFL) type condition which implies that δt needs to be reduced as h (when advection dominates) or h^2 (when diffusion dominates). Although implicit methods do not need to satisfy this type of conditions to be stable, equilibration of errors suggest a similar decay and, when applied to nonlinear problems, experience indicates that a similar reduction is actually needed to achieve convergence. Therefore, even if the spatial solver has perfect weak scalability, it will be applied sequentially an increasing number of times and thus, weak scalability of the full simulation will not be possible.

The research for the development of parallel-in-time and space-time methods has a long history, starting with the first parallel computing architectures [13]. The problem was identified as a critical performance issues in [1] and amount of literature about the subject

has been increasing during the last years, particularly in view of the future transition to exascale.

Since the pioneer work of Nievergelt [20] parallel in time methods have been steadily developed arriving to the celebrated "Pararreal" method in [18]. In this family of methods the time interval is partitioned in strips and a traditional time marching scheme is used on each slab. An external iterative procedure is in charge of the convergence to the actual solution in the whole time window. For the solution of at each time slab a parallelization in space can be used in a nested fashion.

A more general class of methods try to directly find the solution in space-time, looking for parallelization opportunities on the whole problem. A generalisation of the parareal method in this direction was proposed in [15]. Another method of this type is the PFASST [11, 19], which combines a spectral deferred correction time integration with a nonlinear multigrid spatial solver. This method was actually evaluated in the JUQUEEN supercomputer proving weak scalability up to 65k cores [21]. Further, weakly scalable space-time AMG methods can be found in [12, 14, 24].

In this work we follow a different approach, based on the Balancing Domain Decomposition by Constraints (BDDC) method that has been developed at the Universitat Politecnica de Catalunya (UPC) and the International Centre for Numerical Methods in Engineering (CIMNE) over the years [2–6]. This method and its extension to handle space-time problems are described in the next section.

3 The space-time BDDC preconditioner

The BDDC method was originally proposed in [9] and extended to cover a wide range of applications, such as, e.g incompressible flows [17, 22] and electromagnetic problems [10, 25]. Even though the mathematical framework is quite complex the underlying can be explained in simple terms.

When large scale computations are performed based on some mesh-based discretization of the domain, the number of unknowns in the interfaces between subdomains is huge, particularly in $3D$. This interface unknowns located at different processors are related to each other and because they belong to different address spaces these relations are implemented in terms of communications. To obtain the final solution of the global problem we need to iteratively correct these interface values until convergence is achieved within a Krylov procedure. Because the number of iterations is determined by the condition number which scales as h^{-2} for, e.g. the finite element (FE) approximation of the Poisson problem, a preconditioner is required to accelerate the convergence.

As a model problem we consider a (possibly) nonlinear parabolic equation,

$$\partial_t u - \nabla \cdot \big((\nu + \nu^t)\nabla u\big) = f \quad \text{in } \Omega \times (0,T], \tag{1}$$

$$u = 0 \quad \text{on } \partial\Omega \times (0,T], \tag{2}$$

$$u(0,x) = u^0, \quad \text{on } \partial\Omega$$

where ν is a constant viscosity coefficient and the nonlinearity arises in the dependence of the "turbulent" viscosity with the unknown, which we take of a p-Laplacian type

$$\nu^t = c|\nabla u|^{p-2}. \tag{3}$$

We consider both the linear problem, with $c = 0$ (and therefore $\nu^t = 0$) and the nonlinear one, with $c = 1$ and $p = 3$, in the examples of Sect. 4. The p-Laplacian model for the scalar equation is a simple version of the popular Smagorinsky model of turbulent flows [16].

Observe that in Eq. 1 we consider a cartesian product approximation of the space-time domain, which permits to approximate in time using finite differences, as usual, even if the problem is solved in the whole space-time domain. Therefore, we consider the weak-in-space, strong-in-time form of the problem: find $u(t) \in V$ such that

$$(v, \partial_t u) + \big(\nabla v, (\nu + \nu^t)\nabla u\big) = (v, f)$$

for any $v \in V$.

We consider a standard FE approximation in space and backward Euler discretization in time, with a uniform partition of the interval into K segments of size δ_t and a spatial mesh of size h, which gives a space-time mesh discretization parameter $\delta = (\delta_t, h)$. The fully discrete version of the problem reads: find $u_\delta = (u_h^1, ..., u_h^K) \in V_\delta$ such that

$$a(u_\delta, v_\delta) = l(v_\delta) \tag{4}$$

for any $v_\delta \in V_\delta$ where

$$a(u_\delta, v_\delta) = \beta \left(v_h^1, u_h^1\right) + \beta \sum_{k=2}^{K} \left(v_h^k, u_h^k - u_h^{k-1}\right) + \left(\nabla v_h^k, (\nu + \nu^t)\nabla u_h^k\right) \tag{5}$$

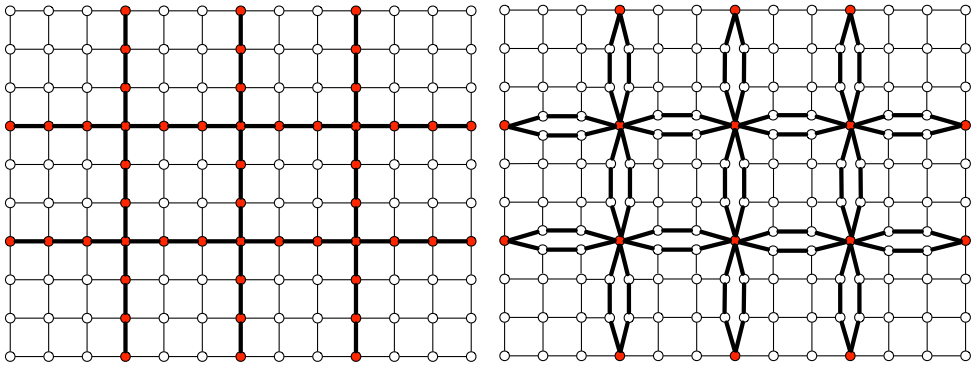
and

$$l(v_\delta) = \sum_{k=1}^{K} \left(v_h^k, f\right) \tag{6}$$

In Eq. 5 $\beta = 1/\delta_t = T/K$ whereas $\beta = 0$ for the steady problem.

The idea of the BDDC preconditioner is to solve this problem in a space of reduced continuity. The FE space of functions V_δ where the solution is sought is build using a standard mesh, shown in Fig. 1a where red points mark degrees of freedom on subdomain boundaries that require communications. The preconditioner is built in the BDDC space $\tilde{V}_\delta$ shown in Fig. 1b. This is an intermediate space between V_δ and the fully discontinuous space V_δ^d, that is $V_\delta \subseteq \tilde{V}_\delta \subseteq V_\delta^d$.

In the following we describe the main ingredients of this construction and which of them has been reworked to accommodate space-time problems. In order to use the solution of the problem in the space $\tilde{V}_\delta$ a transfer operator is required. The injection $E : \tilde{V}_\delta \to V_\delta$ is defined as $E = E_1 \times \ldots \times E_n$ with $E_i = R_i^t W_i$ on each subdomain (n is the number of subdomains), where R_i is the restriction operator (a global to local map) and W_i a weighting matrix. The operator E permits to recover a solution in the continuos space V_δ and it is implemented as weighting, a communication and the final addition of the values at the two (or more) sides of the interface between subdomains, see [3] for the details of the implementation. When applying the BDDC method to spatial problems, e.g. Poisson, the weighting matrix is defined as a diagonal matrix whose entries are the inverse of the number of subdomains that share a given point (for interface points) or one (for interior points). In the extension to space-time, it is better to take the value on the interface as the value on the first subdomain when looking at time interfaces, because this respects



(a) V_δ (b) $\tilde{V}_h$

Figure 1: Differences between subassembled space and BDDC

causality, while keeping the standard definition on spatial interfaces, see [2] for a detailed discussion of this point.

Given a residual r_δ (inside a Krylov iterative procedure), the preconditioner gives $x_\delta = B^{-1}_{BDDC} r_\delta$ from the solution $\tilde{x}_\delta$ of

$$\tilde{a}(\tilde{x}_\delta, \tilde{v}_\delta) = \langle r_\delta, E\tilde{v}_\delta \rangle, \quad \forall \tilde{v}_\delta \in \tilde{V}_\delta. \tag{7}$$

Observe that the solution $\tilde{x}_\delta$ is discontinuous between subdomains (except at corners, the red points in Fig. 1b) and therefore it is postprocessed applying the operator E. Finally, interior values on each subdomain are corrected using a discrete harmonic operator $\mathcal{E}$ to obtain $x_\delta = \mathcal{E} E \tilde{x}_\delta$.

In order to solve problem 7 in parallel, the BDDC space is decomposed as $\tilde{V}_h = \tilde{V}_F \oplus \tilde{V}_C$ so that functions $\tilde{v} = [\tilde{v}_\circ \; \tilde{v}_\bullet]$ (see Fig. 1b) are decomposed as $\tilde{v} = \tilde{v}_F + \tilde{v}_C$ in a unique way. By definition, the fine space $\tilde{V}_F$ is made up of functions of the form $[\tilde{v}_\circ \; 0]$. Because these points are the only ones connecting subdomains, the fine space is built from independent contributions from each subdomain.

A key aspect of the BDDC method is the definition of the coarse space that couples subdomains. Let us consider the spatial BDDC for symmetric problems, that is, $\beta = 0$ in Eq. 5. In this case, $\tilde{V}_C$ is defined as the a-orthogonal complement of $\tilde{V}_F$, that is $\tilde{V}_C \perp_a \tilde{V}_F$. This orthogonality implies

$$\tilde{a}(\tilde{x}_F, \tilde{y}_C) = 0 \tag{8}$$

$\forall \tilde{x}_F \in \tilde{V}_F$ and $\forall \tilde{y}_C \in \tilde{V}_C$. In an implementation, an explicit construction of $\tilde{V}_C$ is required, through the computations of a basis, which can performed solving local problems, see [3]. For symmetric problems Eq. 8 also implies $\tilde{a}(\tilde{y}_C, \tilde{x}_F) = 0$, $\forall \tilde{x}_F \in \tilde{V}_F$ and $\forall \tilde{y}_C \in \tilde{V}_C$. Then Eq. 7 is decoupled into fine and coarse problems which can be solved separately and concurrently, that is, solving the problem: find $\tilde{x}_F \in \tilde{V}_F$ and $\tilde{x}_C \in \tilde{V}_C$ such that

$$\begin{aligned} \tilde{a}(\tilde{x}_F, \tilde{v}_F) &= \langle r, E\tilde{v}_F \rangle \quad \forall \tilde{v}_F \in \tilde{V}_F \\ \tilde{a}(\tilde{x}_C, \tilde{v}_C) &= \langle r, E\tilde{v}_C \rangle \quad \forall \tilde{v}_C \in \tilde{V}_C \end{aligned}$$

and recover $\tilde{x} = \tilde{x}_F + \tilde{x}_C$. This concurrency was exploited in [4] to develop a highly scalable implementation overlapping fine and coarse calculations.

When moving to space-time problems, the bilinear form in Eq. 5 is not symmetric anymore. The extension to non-symmetric problems [2, 23] requires two different coarse spaces, $\tilde{V}_C$ and $\tilde{V}_C^*$, to satisfy the orthogonality properties

$$\tilde{a}(\tilde{x}_F, \tilde{y}_C^*) = 0 \tag{9}$$
$$\tilde{a}(\tilde{y}_C, \tilde{z}_F) = 0 \tag{10}$$

$\forall \tilde{x}_F, \tilde{z}_F \in \tilde{V}_F$, $\forall \tilde{y}_C \in \tilde{V}_C$ and $\forall \tilde{y}_C^* \in \tilde{V}_C^*$. After the basis of both $\tilde{V}_C$ and $\tilde{V}_C^*$ are built, the coarse grid correction is computed as: find $\tilde{x}_C \in \tilde{V}_C$

$$\tilde{a}(\tilde{x}_C, \tilde{v}_C^*) = \langle r, E\tilde{v}_C^* \rangle \quad \forall \tilde{v}_C^* \in \tilde{V}_C^*$$

4 Numerical experiments

In this section we present the numerical experiments carried out during the project. We first describe the setup of these experiments, including their design, details of the algorithmic parameters and some information of the High Performance Computing (HPC) platform in Sect. 4.1. Then we proceed to present the results obtained for the linear problem in Sect 4.2 and finally those obtained for the nonlinear problem in Sect. 4.3.

4.1 Experimental setup

The goal of the experiments is to analyze the weak scalability of the algorithms presented in Sect. 3. This type of study is typically made for the discretization of the (spatial) Poisson problem. However, when dealing with problems obtained from the discretization of partial differential operators with several terms, the weak scalability only makes sense when the relation between all terms of the discrete formulation remain constant along the analysis. Otherwise, the effect of changing the discretization would be blurred by the change in the nature of the problem. Therefore, a key requirement in the weak scalability analysis is to keep constant the relative weight of the discrete differential operators, leading to the concept of CFL-constant weak scalability [12**?**]. To fulfil this requirement, for the weak scalability analysis in space, we proceed as follows: for a fixed physical problem (i. e. for a fixed value of physical properties, initial and boundary conditions), and keeping fixed the FE and subdomain characteristic sizes (h and H respectively), the physical domain will be scaled by a factor that depends on the number of processors per dimension[1]. A similar approach is used in the weak scalability analysis in time, keeping constant the FE and subdomain characteristic sizes (δ_t and Δ_t respectively). In space-time scaling, both, the spatial domain and time integration interval are properly scaled in order to keep these characteristic sizes constant. As a consequence, the number of FE and subdomains will increase, but the values of $h, H, \Delta t$ and δt will remain constant along the test. This allows us keep a constant diffusive CFL number, defined as $\text{CFL} = \nu \frac{\delta t}{h^2}$. We proceed as follows:

- In space, the FE and subdomain characteristic sizes (h and H) are considered fixed. The physical domain will be scaled in order to increase the number of tasks involved in the test, i.e. considering $\Omega' = \alpha\Omega = [0, \alpha]^d$, $\alpha \in \mathbb{N}^+$, the global conforming partition will involve αn_h FE cells and αn_H subdomains per spatial dimension.

[1] There are no restrictions on the space-time domain scaling, both, isotropic and anisotropic scalings are supported.

- With respect to time, a similar procedure is carried out. The time integration is distributed into P_t subdomains. In order to keep the characteristic sizes (Δt and δt) constant through all tests, the initial time interval is properly scaled with the number of subdomains P_t. As consequence, the time interval of the problem to be solved will be P_t times larger (in terms of time) than the initial one.

Regarding the iterative solvers used in the experiments, we consider the following set up:

- We use the standard Newton-Rahpson method as nonlinear iterative solver. Defining a maximum of 10 nonlinear iterations, the scheme is automatically stopped whenever the ratio between the Euclidean norms of the solution and its increment, for a given nonlinear iterate, is lower than a predefined tolerance, i. e. $\frac{\|\Delta \mathbf{u}\|_2}{\|\mathbf{u}^*\|_2} \leq \texttt{nls\_rtol}$. We considered $u_0 = 0$ as initial guess for the solution, and $\texttt{nls\_rtol} = 10^{-6}$.

- The STBDDC iterative solver used to solve the system of equations at each nonlinear solver iteration is automatically stopped once the residual, for a given linear iterate r^k, satisfies the following condition: $\|r^k\|_2 \leq \texttt{ils\_rtol} * \|r^0\|_2 + \texttt{ils\_atol}$, with r^0 being the initial residual, and `ils_rtol` and `ils_atol` the relative and absolute predefined tolerances, respectively. As initial guess we choose $x_0 = 0$, and $\texttt{ils\_rtol} = 10^{-5}$ and $\texttt{ils\_atol} = 10^{-14}$.

All the numerical experiments presented herein have been performed at Marenostrum-IV (MN-IV), a supercomputer hosted by the Barcelona Supercomputing Center. This petascale machine is equipped with 3,456 compute nodes interconnected together with the Intel OPA HPC network. Each node has 2x Intel Xeon Platinum 8160 multi-core CPUs, with 24 cores each (i.e. 48 cores per node) and 96 GBytes of RAM.

The approach presented in this report has been implemented and tested in `FEMPAR` [7], an open-source project with an object oriented design pattern coded in Fortran 2008, compiled with Intel v18.0.5 compilers using system recommended optimization flags. `FEMPAR` was linked against the Intel Message Passing Interface (MPI) Library (v2018.4.274) for message-passing and the BLAS/LAPACK and PARDISO available on the Intel MKL library for optimized dense linear algebra kernels and sparse direct solvers, resp. All floating-point operations were performed in IEEE double precision.

4.2 Parabolic linear equation

In these experiments, we consider a 2D parabolic linear equation, that is, we take a constant viscosity $\nu = 1.0$ and $\nu^t = 0$. The source term and the corresponding boundary conditions are chosen such that the solution $u = \sin(\pi x)\sin(\pi y)\sin(\pi t)$ is satisfied. In order to obtain its discrete counterpart, we consider a global conforming uniform partition to discretize both, space and time dimensions. In space, the physical domain $\Omega := [1, 0]^2$, is uniformly partitioned into hexahedra FE ($Q_1(K)$) with the same characteristic size $h := \frac{1}{n_h}$ in both spatial dimensions, with n_h, being the total number of cells per spatial dimension. Similarly, on time, the problem is uniformly partitioned into a discrete mesh with characteristic size $\delta_t := \frac{1}{n_t}$, with n_t as the total number of time steps.

In order to provide a *local load* to each subdomain, this global mesh is also partitioned uniformly into a *coarse grid*, with $P_x \times P_y \times P_t$ subdomains. Each subdomain is provided with a *local load* of $\frac{H}{h} \times \frac{H}{h} \times \frac{\Delta t}{\delta t}$ FE cells, with H and Δt being the spatial and temporal

subdomain characteristic sizes, respectively. In consequence, the size of the global mesh can be computed as $P_x \frac{H}{h} \times P_y \frac{H}{h} \times P_t \frac{\Delta t}{\delta t}$ hexahedra FEs.

4.2.1 Weak scalability in time

To perform the the weak scalability analysis in time, we consider a constant domain size ratio $\left(\frac{H}{h}\right)^2 = 30^2$, taking different values of $K_n = \left(\frac{\Delta t}{\delta t}\right) = \{10, 30, 60\}$. In order to study the sensitivity of the linear iterative solver with respect to the CFL number, the domain was scaled in the temporal dimension, Fig. (2) reports the weak scalability analysis with $T = T_0/2$, and Fig. (3) reports the same study but taking $T' = T_0/60$, with T_0 being the reference time interval $T_0 = (0, 1]$.

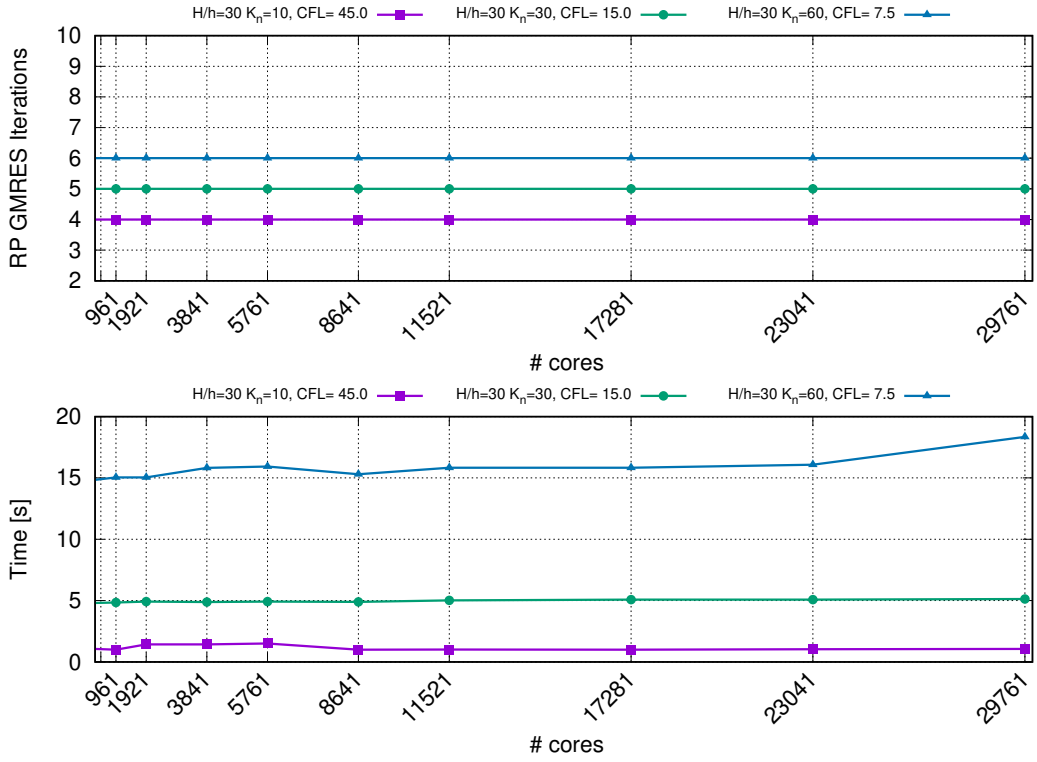


Figure 2: Linear solver iteration counter and total elapsed time with $T' = T_0/2$

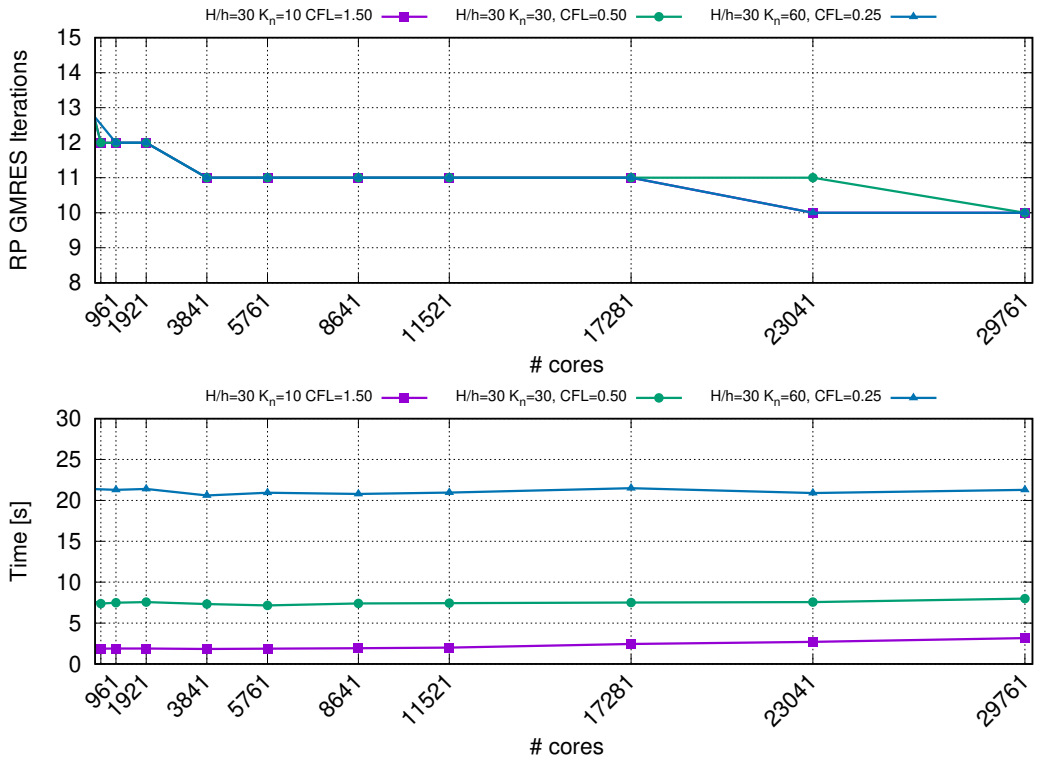


Figure 3: Linear solver iteration counter and total elapsed time with $T' = T_0/60$

As consequence of this study, some conclusions can be obtained. The first one is that, as it is well known, the number of iterations and therefore the wall clock time depend on the local load K_n. Second, the influence of the CFL number on the total number of linear solver iterations, for a fixed number of degrees of freedom, a reduction in the time step size δt will increase the total number of linear solver iterations, however this change is not proportional, if the time step has been reduced by a factor of 60, the total number of linear iterations is increased twice.

However, despite the sensitivity with respect to the CFL number, the algorithm is weak scalable, i.e. for a fixed CFL number, and keeping the same local problem size, the number of linear solver iterations and the elapsed time remain slightly constant as the number of processors is increased.

4.2.2 Weak scalability in space-time

To perform the weak scalability analysis in space-time, we consider three different values of the spatial subdomain size $\left(\frac{H}{h}\right)^2 = 10^2, 30^2, 40^2$. With respect to the time interval discretisation, we consider three different values of $K_n = \frac{\Delta t}{\delta t} = \{10, 30, 40\}$. To study the sensitivity of the linear iterative solver with respect to space-time scaling, two different anisotropic domain scalings have been studied, Fig. (4) and (5) report the results using subdomain grid distributions with $P_x = 2k, P_y = 2k, P_t = 12k$ and $P_x = 6k, P_y = 8k, P_t = k$, respectively, both leading to a $P = P_x \times P_y \times P_t$ subdomain partition of the scaled domain $P_d\Omega_d \times P_t(0, T']$, with $d = \{x, y\}$, P being the total number of subdomains, and T' the scaled time interval. Finally, in order to obtain the full test, k takes the following values: $k = \{2, 3, 4, 5, 6, 7, 8\}$, equivalent to $961, 1927, 3841, 5761, 8641, 11521, 17281$, and 23041 processors. In all cases, the local load per processor can be computed as $\left(\frac{H}{h}\right)^d \times \frac{\Delta t}{\delta t}$.

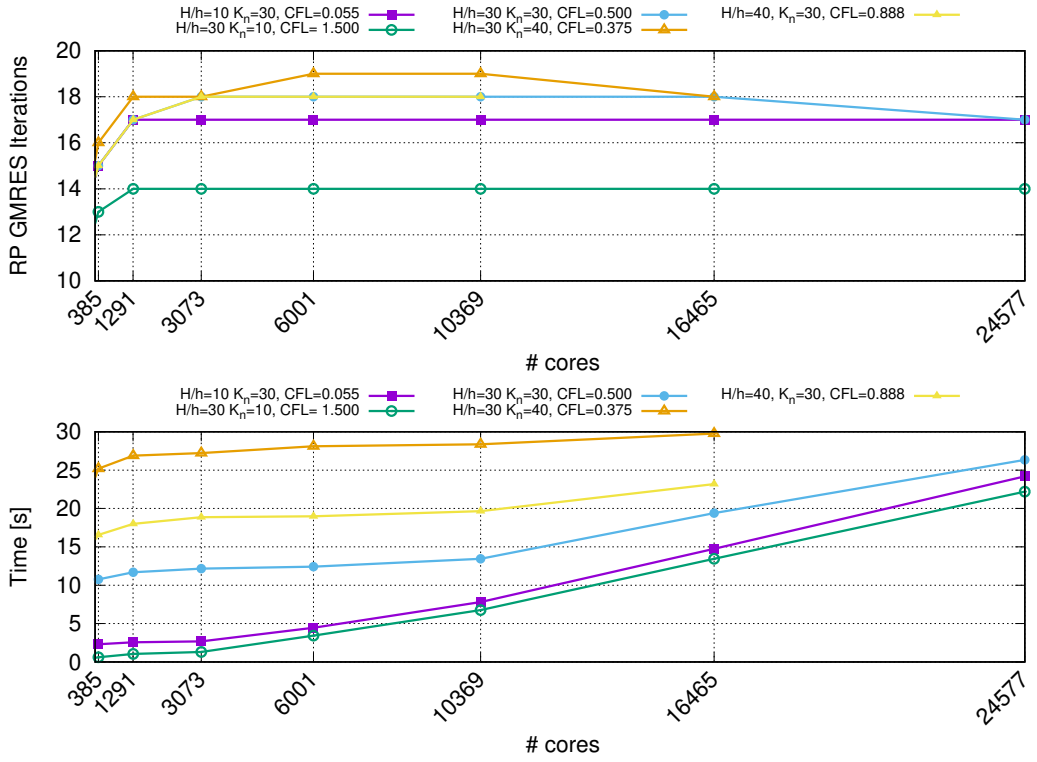


Figure 4: Linear solver iteration counter and elapsing time with $P_x = 2k, P_y = 2k, P_t = 12k$

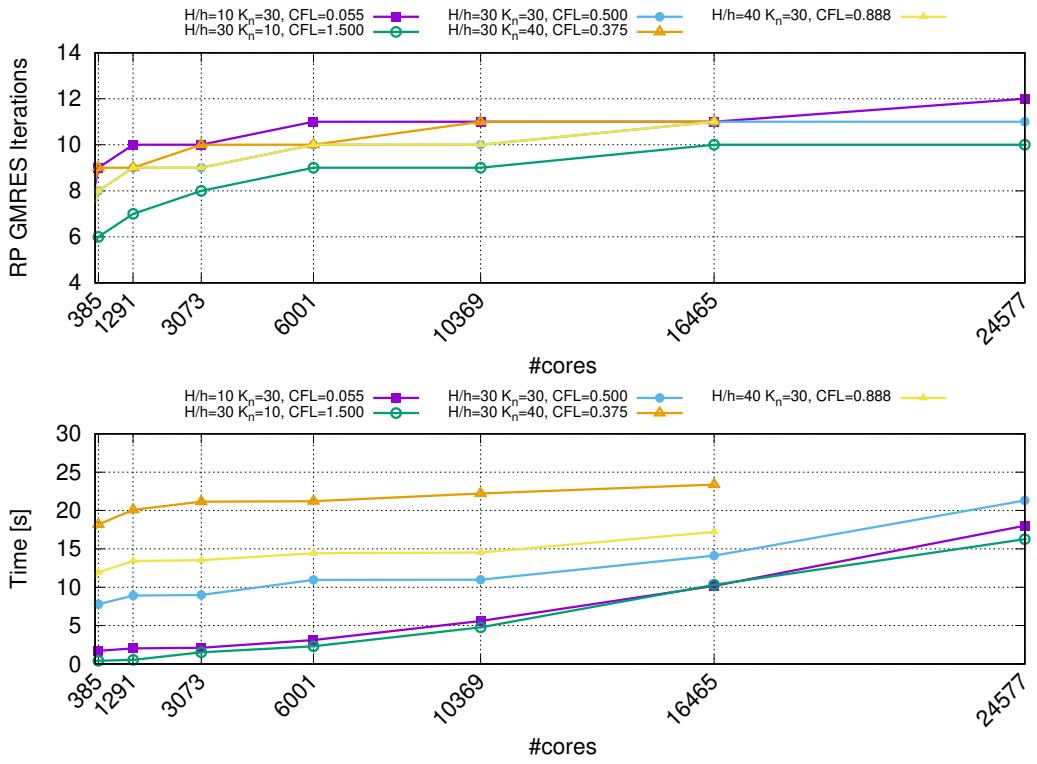


Figure 5: Linear solver iteration counter and total elapsed time with $P_x = 6k, P_y = 8k, P_t = k$

Similar trends can be observed in this case for both spatial partitions. Algorithmic weak scalability (i.e. the number of iterations) is always observed but actual weak scalaibility (i.e. wall-clock times) depends on the local load. When fine scale processors have enough load, the coarse problem is masked and scalability is maintained. For lower loads some degradation is observed when more than 10k cores are used. It is worth mentioning that some points are missing in these figures due to exhausted computing hours available.

4.3 Nonlinear p-Laplacian problem

In section we consider the 2D nonlinear p-Laplacian equation with two different viscosities, namely $\nu = 1$ and $\nu = 10$, to which we add the turbulent viscosity given by Eq. 3 with $c = 1$ and $p = 3$. Observe that the bigger the value of ν the closer the problem is to the linear one. As in the previous linear case, the source term and the corresponding boundary conditions are chosen such that the solution is $u = \sin(\pi x)\sin(\pi y)\sin(\pi t)$. The domain is $\Omega = [0, 1]^d \times (0, T]$ and we consider homogeneous boundary and initial conditions. To discretize the problem, let us consider a global conforming uniform partition. In space, the physical domain Ω, is uniformly partitioned into hexahedra FEs ($Q_1(K)$), with the same characteristic size $h := \frac{L}{n_h}$ in both spatial dimensions, being n_h the total number of cells per spatial dimension. The time interval is also uniformly partitioned into a discrete mesh with characteristic size $\delta_t := \frac{T'}{n_t}$, with n_t being the total number of time steps and T' the scaled time interval. In this case, we will consider the same criteria to scale the domain, and the iterative solvers setup presented in Sec. 4.1.

4.3.1 Weak scalability in time

To perform the the weak scalability analysis in time, we consider two different values for the subdomain size $\left(\frac{H}{h}\right)^2 = \{10^2, 30^2\}$, and two different values of $K_n = \left(\frac{\Delta t}{\delta t}\right) = \{10, 30\}$. In this case, the time interval T' is properly scaled in order to keep a constant value of K_n while the number of processors P_t is increased. The spatial scaling parameter $\alpha = 1.0$, is considered, i. e. no scaling in the spatial dimensions is used. The subdomain grid distribution considered in this case is $P_x = 1, P_y = 1, P_t = 48k$, with $k = \{8, 27, 80, 120, 240, 360, 480\}$, corresponding to $961, 1297, 3841, 5761, 8641, 11521, 17281$, and 23041 processors.

In order to study the influence of the constant viscosity ν in the behavior of both iterative solvers, two different values has been considered $\nu = \{1.0, 10.0\}$. In the following, Fig. (6) and (7) report the weak scalability analysis with $\nu = 1.0$ and $\nu = 10.0$, respectively.

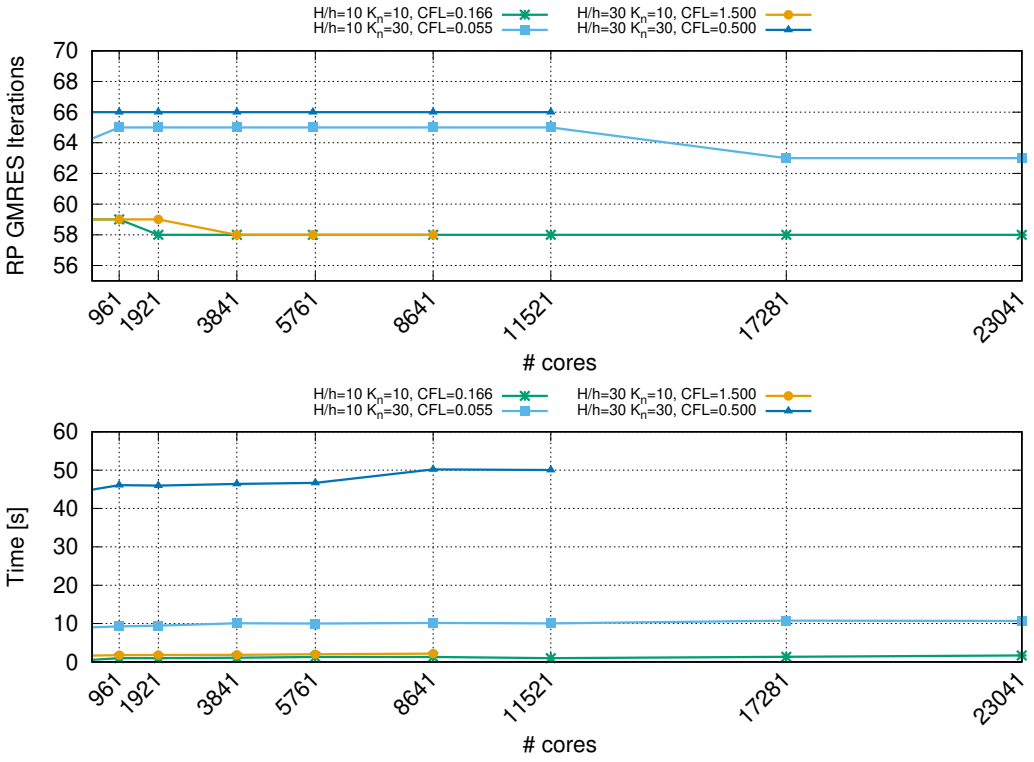


Figure 6: Linear solver iteration counter and total elapsed time with $\nu = 1.0$

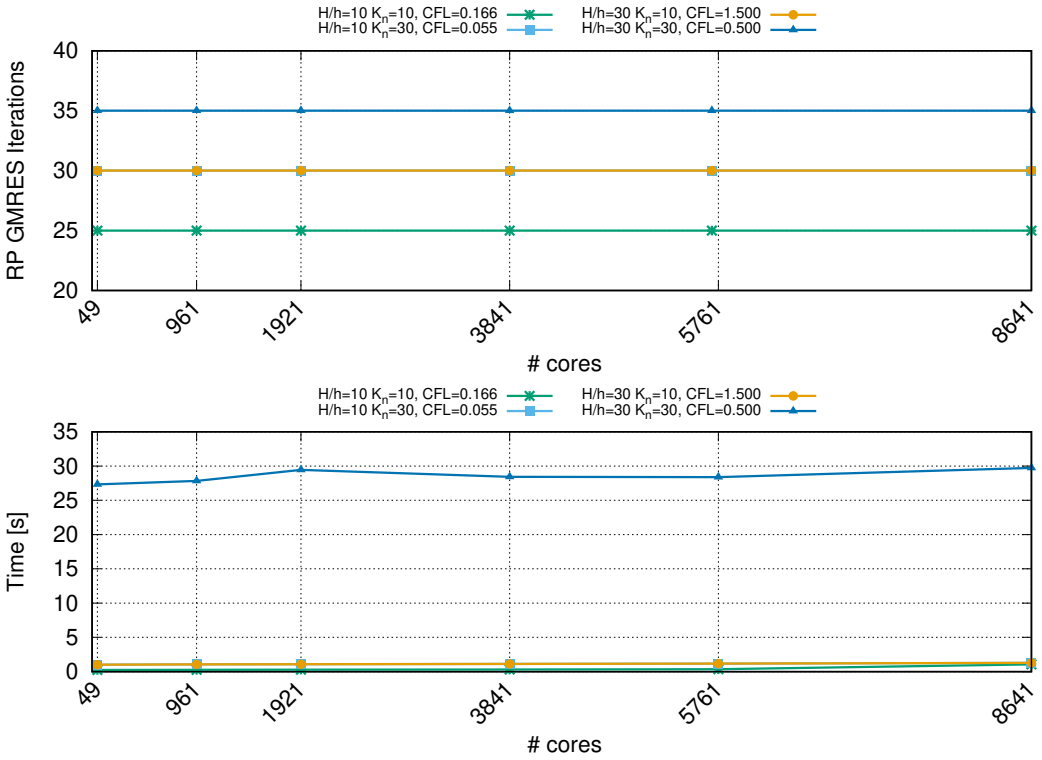


Figure 7: Linear solver iteration counter and total elapsed time with $\nu = 10.0$

Out of these plots, we can state some conclusions. Regarding the influence of the viscosity ν on the total number of linear solver iterations: a reduction in the viscosity ν

results in more linear solver iterations. In most of cases, once the viscosity is reduced by a factor of 10, the total number of linear iterations is increased around twice.

However, despite the sensitivity with respect to the viscosity ν, the algorithm is weak scalable, i.e. for a fixed CFL, the number of linear solver iterations and its total elapsed time remain constant as the number of processors is increased.

4.3.2 Comparison against sequential approach

The goal of this section is to compare of the STBDDC with a classical sequential approach. We consider a fixed CFL=1.50. This number is obtained by considering $\left(\frac{H}{h}\right)^2 = 30^2$ and $K_n = \frac{\Delta t}{\delta t} = 10$. The subdomain grid distribution is taken as $P_x = 3, P_y = 4, P_t = 4k$ leading to $P = P_x \times P_y \times P_t = 48k$ subdomains. The full test is obtained by taking values of $k = \{1, 2, 3, 4, 8, 16\}$, equivalent to $P_t = \{4, 8, 12, 16, 32, 64\}$ time subdomains.

With respect to the sequential approach, only spatial parallelism is exploded, a subdomain grid with $P_s = P_x \times P_y = 3 \times 4 = 12$ processors is considered. In Fig. (8) and (10), we plot the evolution of the number of local solves computed as the accumulated number of linear iterations through all nonlinear iterations for two different values of viscosity $\nu = \{1, 10\}$, while Fig. (9) and (11) plot the results of the elapsed time for the same set of viscosities $\nu = 1, 10.0$. As it can be observed in these figures, at some point the STBDDC outerperforms the sequential approach both in terms of computational time and number linear solves. Of course, it has to be kept in mind that in the sequential approach (black curves in Figs- 8-11 the number of processors do not grow with the total time, in contrast to the space-time method in which more and more computational resources are used.

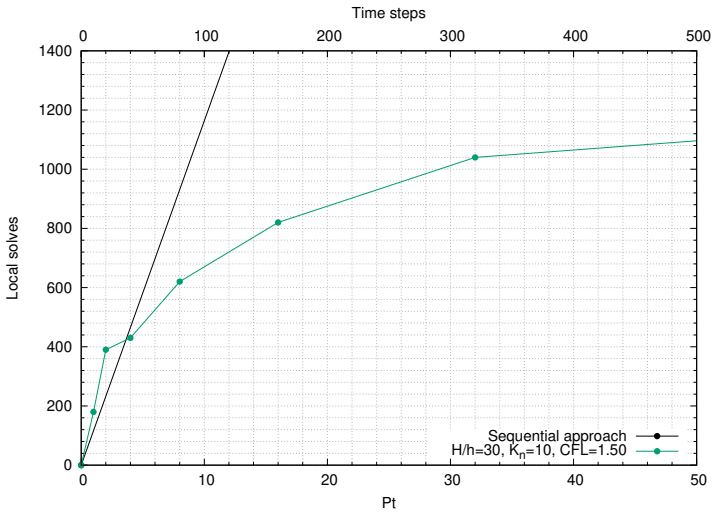


Figure 8: Local solves as function of the number of temporal subdomains P_t with $\nu = 1.0$

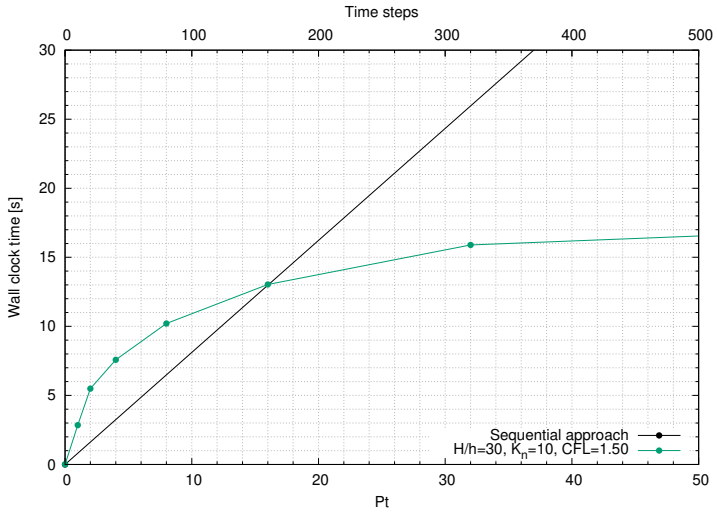


Figure 9: Wall clock time with $\nu = 1.0$

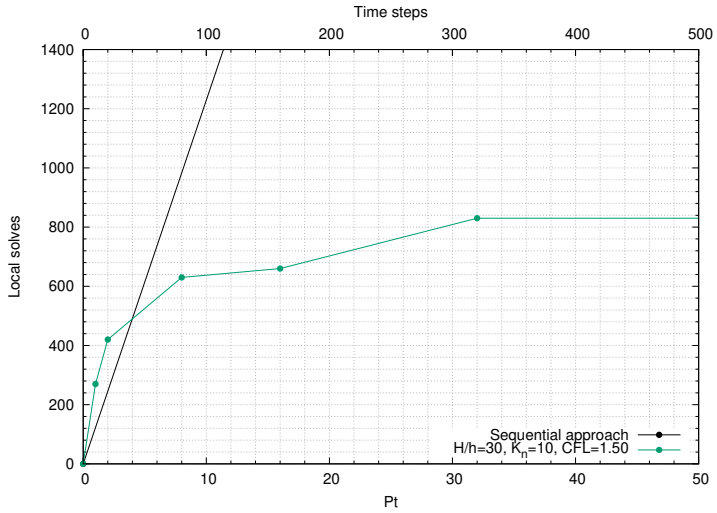


Figure 10: Local solves as function of the number of temporal subdomains P_t with $\nu = 10.0$

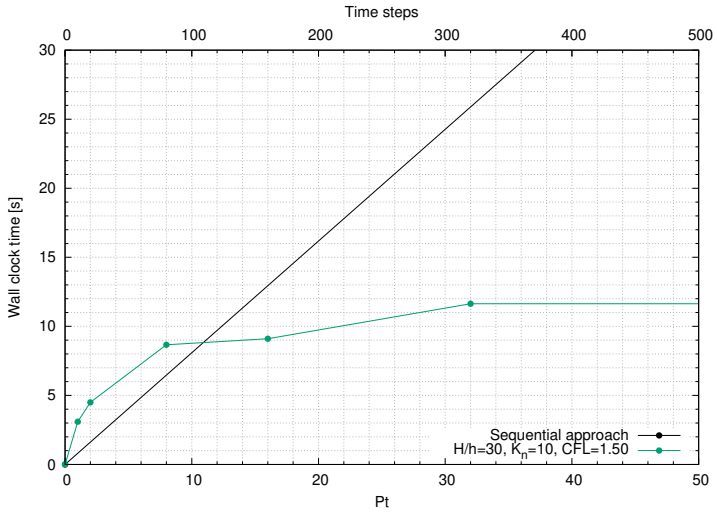


Figure 11: Wall clock time with $\nu = 10.0$

5 Conclusions

In this report we have detailed the implementation of space-time methods in `FEMPAR` and we have presented detailed numerical experiments carried out during the project. These results show that the STBDDC actually permits to extract parallelism when spatial parallelism is exhausted at the price of using more computational resources, both for linear and nonlinear problems. Very good weak scalability upto 30k processors has been observed in MN-IV.

The only limitation that deserves to be mentioned is that the number of processors at which the space-time method starts to make sense depends on the strength of the nonlinearity. As it can be observed in the results of Sect.4.3.2, the STBDDC starts outperforming the sequential approach with 10 for $\nu = 10$ processors compared to 16 processors for $\nu = 1$. In any case, this is not a big difference but the nonlinearity considered herein is very soft. Further research is required to actually quantify the severity of this problem.

References

[1] Report on the Workshop on Extreme-Scale Solvers: Transition to Future Architectures. Technical report, U.S. Department of Energy, 2012. URL `https://science.osti.gov/-/media/ascr/pdf/program-documents/docs/reportExtremeScaleSolvers2012.pdf`.

[2] S. Badia and M. Olm. Space-Time Balancing Domain Decomposition. *SIAM Journal on Scientific Computing*, 39(2):C194—-C213, jan 2017. ISSN 1064-8275. doi:10.1137/16M1074266.

[3] S. Badia, A. F. Martín, and J. Principe. Implementation and scalability analysis of balancing domain decomposition methods. *Archives of Computational Methods in Engineering*, 20(3):239–262, 2013. doi:10.1007/s11831-013-9086-4.

[4] S. Badia, A. F. Martín, and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM Journal on Scientific Computing*, 36(2):C190–C218, 2014. doi:10.1137/130931989.

[5] S. Badia, A. F. Martín, and J. Principe. On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections. *Parallel Computing*, 50:1–24, dec 2015. doi:10.1016/j.parco.2015.09.004.

[6] S. Badia, A. F. Martín, and J. Principe. Multilevel Balancing Domain Decomposition at Extreme Scales. *SIAM Journal on Scientific Computing*, 38(1):C22—-C52, 2016. ISSN 1064-8275. doi:10.1137/15M1013511.

[7] S. Badia, A. F. Martín, and J. Principe. FEMPAR: An Object-Oriented Parallel Finite Element Framework. *Archives of Computational Methods in Engineering*, pages 1–77, 2017.

[8] S. Badia, A. Martıín, and J. Principe. FEMPAR, 2018. URL `http://www.fempar.org`.

[9] C. R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing*, 25(1):246–258, 2003. ISSN 10648275. doi:10.1137/S1064827502412887. URL `http://www.siam.org/journals/ojsa.php`.

[10] C. R. Dohrmann and O. B. Widlund. Some Recent Tools and a BDDC Algorithm for 3D Problems in H(curl). In *Domain Decomposition Methods in Science and Engineering XX*, volume 91 of *Lecture Notes in Computational Science and Engineering*, pages 15–25. Springer Berlin Heidelberg, may 2013. ISBN 978-3-642-35274-4.

[11] M. Emmett and M. Minion. Toward an efficient parallel in time method for partial differential equations. *Communications in Applied Mathematics and Computational Science*, 7(1):105–132, mar 2012. ISSN 2157-5452, 1559-3940. doi:10.2140/camcos.2012.7.105. URL `http://msp.org/camcos/2012/7-1/p04.xhtml`.

[12] R. Falgout, S. Friedhoff, T. Kolev, S. MacLachlan, and J. Schroder. Parallel Time Integration with Multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635—-C661, jan 2014. ISSN 1064-8275. doi:10.1137/130944230.

[13] M. J. Gander. 50 Years of Time Parallel Time Integration. pages 69–113. Springer, Cham, 2015. doi:10.1007/978-3-319-23321-5_3.

[14] M. J. Gander and M. Neumüller. Analysis of a New Space-Time Parallel Multigrid Algorithm for Parabolic Problems. *arXiv:1411.0519 [math]*, nov 2014. URL `http://arxiv.org/abs/1411.0519`.

[15] M. J. Gander, R.-J. Li, Y.-L. Jiang, and R.-J. Li. Parareal Schwarz Waveform Relaxation Methods. In *Domain Decomposition Methods in Science and Engineering XX*, volume Part II of *Lecture Notes in Computational Science and Engineering*, pages 451–458. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-35275-1. URL `https://pdfs.semanticscholar.org/d4dd/c502337890a534ae31db5443920a202bfa0e.pdf`.

[16] J. L. Guermond, J. T. Oden, and S. Prudhomme. Mathematical Perspectives on Large Eddy Simulation Models for Turbulent Flows. *Journal of Mathematical Fluid Mechanics*, 6(2):194–248, 2004. URL http://www.springerlink.com/index/10.1007/s00021-003-0091-5.

[17] J. Li and O. Widlund. BDDC Algorithms for Incompressible Stokes Equations. *SIAM Journal on Numerical Analysis*, 44(6):2432–2455, jan 2006. ISSN 0036-1429, 1095-7170. doi:10.1137/050628556. URL http://epubs.siam.org/action/showAbstract?page=2432&volume=44&issue=6&journalCode=sjnaam.

[18] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(7):661–668, apr 2001. ISSN 0764-4442. doi:10.1016/S0764-4442(00)01793-6.

[19] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht. Interweaving PFASST and Parallel Multigrid. *SIAM Journal on Scientific Computing*, 37(5): S244—-S263, jan 2015. ISSN 1064-8275, 1095-7197. doi:10.1137/14097536X. URL http://epubs.siam.org/doi/10.1137/14097536X.

[20] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7(12):731–733, dec 1964. ISSN 00010782. doi:10.1145/355588.365137. URL http://portal.acm.org/citation.cfm?doid=355588.365137.

[21] R. Speck, D. Ruprecht, M. Emmett, M. Bolten, and R. Krause. *A space-time parallel solver for the three-dimensional heat equation*, volume 25. jul 2013. doi:10.3233/978-1-61499-381-0-263.

[22] X. Tu. A three-level BDDC algorithm for a saddle point problem. *Numerische Mathematik*, 119(1):189–217, sep 2011. ISSN 0029-599X, 0945-3245. doi:10.1007/s00211-011-0375-2. URL http://link.springer.com/article/10.1007/s00211-011-0375-2.

[23] X. Tu and J. Li. A balancing domain decomposition method by constraints for advection-diffusion problems. *Communications in Applied Mathematics and Computational Science*, 3(1):25–60, jul 2008. ISSN 2157-5452, 1559-3940. doi:10.2140/camcos.2008.3.25. URL http://escholarship.org/uc/item/0r28q60w.pdfhttp://msp.org/camcos/2008/3-1/p02.xhtml.

[24] T. Weinzierl. A geometric space-time multigrid algorithm for the heat equation. *Numerical Mathematics: Theory, Methods and Applications*, 5(01):110–130, 2012. URL http://journals.cambridge.org/abstract_S1004897900000258.

[25] S. Zampini, O. B. Widlund, and D. E. Keyes. Scalable and Robust BDDC Preconditioners for Reservoir and Electromagnetics Modeling. sep 2015. doi:10.3997/2214-4609.201414030.