Efficient preconditioners for solving dynamical optimal transport via interior point methods

Enrico Facca, Gabriele Todeschi, Andrea Natale, Michele Benzi

EFFICIENT PRECONDITIONERS FOR SOLVING DYNAMICAL OPTIMAL TRANSPORT VIA INTERIOR POINT METHODS

Enrico Facca
University of Bergen
Bergen, Norway
enrico.facca@uib.no

Gabriele Todeschi
Univ. Grenoble Alpes, ISTerre
F-38058 Grenoble, France
gabriele.todeschi@univ-grenoble-alpes.fr

Andrea Natale
Univ. Lille, Inria, CNRS, UMR 8524
Laboratoire Paul Painlevé
Lille, France
andrea.natalea@inria.fr

Michele Benzi
Scuola Normale Superiore
Piazza dei Cavalieri, 7, 56126
Pisa, Italy
michele.benzi@sns.it

ABSTRACT

In this paper we address the numerical solution of the quadratic optimal transport problem in its dynamical form, the so-called Benamou-Brenier formulation. When solved using interior point methods, the main computational bottleneck is the solution of large saddle point linear systems arising from the associated Newton-Raphson scheme. The main purpose of this paper is to design efficient preconditioners to solve these linear systems via iterative methods. Among the proposed preconditioners, we introduce one based on the partial commutation of the operators that compose the dual Schur complement of these saddle point linear systems, which we refer as $\boldsymbol{BB}$-preconditioner. A series of numerical tests show that the $\boldsymbol{BB}$-preconditioner is the most efficient among those presented, despite a performance deterioration in the last steps of the interior point method. It is in fact the only one having a CPU-time that scales only slightly worse than linearly with respect to the number of unknowns used to discretize the problem.

***K*eywords** Optimal transport · Benamou-Brenier formulation · Saddle point problem · Algebraic multigrid methods · Preconditioners

1 Introduction

Optimal transport deals with the problem of finding the optimal way to reallocate one nonnegative density into another by minimizing the total cost of displacement in space. In recent years, numerous contributions have been made to the study of this problem, both on the theoretical and computational level. We suggest, for example, the monographs [50, 47, 4, 44] for a detailed presentation of the subject. Due to these advances, optimal transport is nowadays an established tool for many applications including, for example, the analysis of partial differential equations (PDE) [5], physical modeling [47], data science and machine learning [44], economics [28], and inverse problems [36].

When the cost of displacement per unit mass is given by the square of the Euclidean distance, the problem can be recast dynamically, as shown by Benamou and Brenier [8]. Consider a compact and convex domain $\Omega \subset \mathbb{R}^d$ and two nonnegative densities ρ^{in} and ρ^{f} in $L^1(\Omega)$, with $\int_\Omega \rho^{\text{in}}\, dx = \int_\Omega \rho^{\text{f}}\, dx$. To transport the former to the latter, we aim to find a time-dependent density $\rho : [0,1] \times \Omega \to \mathbb{R}_{\geq 0}$ and a velocity field $v : [0,1] \times \Omega \to \mathbb{R}^d$ that solve the following

minimization problem:

$$\min_{\rho,v} \int_0^1 \int_\Omega \frac{\rho|v|^2}{2}\, dt\, dx \;:\quad \begin{cases} \partial_t \rho + \operatorname{div}(\rho v) = 0 & \text{in } [0,1]\times\Omega \\ \rho v \cdot \hat{n} = 0 & \text{on } [0,1]\times\partial\Omega \\ \rho(0,\cdot) = \rho^{\text{in}},\ \rho(1,\cdot) = \rho^{\text{f}} \end{cases}. \tag{1}$$

The total kinetic energy represents the cost of displacement. Thanks to a change of variables $(\rho, v) \to (\rho, m = \rho v)$, (1) can be rewritten as a convex optimization problem. From the optimality conditions, one can deduce that the optimal velocity field is the gradient of a potential $\phi : [0,1]\times\Omega \to \mathbb{R}$. The potential ϕ and the density ρ are given as the solution of the following system of PDEs:

$$-\partial_t \rho - \operatorname{div}(\rho \nabla \phi) = 0, \tag{2a}$$

$$\partial_t \phi + \frac{\|\nabla \phi\|^2}{2} + s = 0, \tag{2b}$$

$$\rho \geq 0,\ s \geq 0,\ \rho s = 0, \tag{2c}$$

with boundary conditions $\rho(0,\cdot) = \rho^{\text{in}}$, $\rho(1,\cdot) = \rho^{\text{f}}$, $\rho \nabla \phi \cdot \hat{n} = 0$ on $[0,1]\times\partial\Omega$. The auxiliary variable $s : [0,1]\times\Omega \to \mathbb{R}_{\geq 0}$ is related to the positivity constraint on ρ.

The Benamou-Brenier formulation offers several advantages. In addition to the total displacement cost, it also directly provides how to continuously reallocate the mass. This also constitutes a natural way to define interpolations between densities. Furthermore, it draws a clear link between optimal transport and continuum mechanics, and it is naturally suited for Eulerian discretizations. Finally, it can be easily generalized to other problems by penalizing/constraining the evolution ρ (such as variational mean field games and planning problems [2], or unbalanced optimal transport [16, 6]). On the other hand, the numerical solution of (1), or its system of optimality conditions (2), poses significant challenges. Although it is a convex optimization problem, it is nonlinear and, in general, non-smooth for vanishing densities. Moreover, it is a time-space boundary value problem and there is a positivity constraint on ρ to take into account.

While different strategies have been proposed to discretize the Benamou-Brenier formulation, most of these rely on staggered time discretization and a space discretization which may be based on finite differences [8, 41], finite volumes [29, 38, 34, 32] or finite elements [35, 39, 34]. Here, we focus on the framework considered in [38], where one uses finite volumes in space with a two-level discretization of the domain Ω, in order to discretize the density ρ and the potential ϕ separately, which alleviates some checkerboard instabilities that may appear when the same grid is used to discretize both variables (a strategy first used in [26, 27] when dealing with optimal transport with unitary displacement cost given by the Euclidean distance). This choice is not restrictive since such a framework constitutes a generalization of the finite volume scheme studied in [29, 34] (in which a single grid is used for density and potential), and it also contains as special cases the finite difference scheme in [41] (when a single Cartesian grid is used in space), or the discrete transport models on networks studied in [23] (by an appropriate reinterpretation of the space tessellation).

From a computational point of view, the numerical solution of the resulting discrete optimization problem is generally tackled by iterative first-order primal-dual optimization schemes (see, e.g., [8, 41]). The computational cost per iteration of these methods is typically low but the total number of iterations strongly depends on the data and the tuning of the parameters, generally increasing with the problem size. In this paper we will consider instead a second-order approach, an Interior Point (IP) method proposed in [38], where the optimization problem eq. (1) is relaxed by adding a logarithmic barrier function (we refer the reader to [51, 14, 30] for a broad introduction of IP methods). This perturbation provides smoothness by enforcing the strict positivity of the density and uniqueness of the solution. The problem can then be effectively solved by solving the relaxed optimality conditions via the Newton method, and the original unperturbed solution is retrieved by repeating this procedure while reducing the relaxation term to zero. Numerical experiments in [38] show that the total number of Newton iterations remains practically constant, independently of the number of unknowns.

The most demanding task in IP methods is the solution of a sequence of saddle point linear systems in the following form

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T \\ \boldsymbol{\mathcal{B}} & -\boldsymbol{\mathcal{C}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix},$$

generated by the Newton method. In our problem all matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{C}}$ change at each Newton iteration but they are sparse, hence iterative solvers are the natural candidate for solving these linear systems. Iterative methods were not considered in [38], and thus the goal of this paper is to determine whether one can devise a preconditioning strategy such that, ideally, the total CPU time scales linearly with the number of unknowns.

We consider three preconditioners: a preconditioner based on the approximation of the inverse of the primal Schur complement $\boldsymbol{\mathcal{S}}_p = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{B}}^T$, the SIMPLE preconditioner [42], and one preconditioner inspired by the ideas in [20], where we approximate the inverse of the dual Schur complement $\boldsymbol{\mathcal{S}} = -\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^{-1}\boldsymbol{\mathcal{B}}^T$ looking at the differential nature of the operators that compose $\boldsymbol{\mathcal{S}}$ (note that, formally, we cannot write $\boldsymbol{\mathcal{A}}^{-1}$ since in our problem the matrix $\boldsymbol{\mathcal{A}}$ will be singular). A series of numerical experiments suggest that the latter, named $\boldsymbol{BB}$-preconditioner, is the most efficient among those presented in this paper, despite some loss of robustness in the last IP iterations.

Let us stress that, while we conducted our experiments using the scheme described in [38], we do expect that the $\boldsymbol{BB}$-preconditioner will provide good results when adopting different discretization schemes even outside the framework considered here, e.g. when finite elements are used. This is because it is entirely derived from the differential operators involved in the continuous problem.

1.1 Related works

The literature on solving eq. (2) using second-order and/or Newton-based methods is relatively scarce. The most closely related works are [3, 32]. In the first paper, the authors study linear algebra approaches involved in the solution of mean-field games systems via the Newton method. Problem (1) may in fact be seen as the vanishing viscosity limit of a particular mean-field game. However, their linear algebra approaches do not fit into the staggered temporal grids, which is generally required to discretize the Benamou-Brenier formulation. Moreover, they lose efficiency with vanishing viscosity. In the second paper, the authors solved precisely the Benamou-Brenier problem using an inexact Newton method combined with finite volumes on Cartesian grids. However, the discretization scheme is not designed to handle densities ρ^{in} and ρ^{f} with compact support. The solution of the sequence of saddle-point linear systems associated to the Newton method was performed using a preconditioning strategy that resembles the approach based on the approximation of the primal Schur complement, dropping some of the mixed time-space terms. However, this approach did not give satisfactory results in our experiments (more details in section 4.2).

1.2 Paper structure

In section 2 we summarize the discretization method proposed in [38], the IP approach used to solve the optimization problem, and the nonlinear problems associated with it. Then, in section 3, we present the linear system studied in this paper. Finally, in section 4 we present the preconditioners described in this paper, together with some numerical experiments where we solve a specific test case for different time and space refinements. In particular, in section 4.5 we compare the CPU time required by the preconditioners in solving different test cases.

2 Discrete problem and Interior Point method

In this section we present the discrete counterpart of system eq. (2) considered in [38]. This combines the use of finite volumes with staggered temporal grids. We further present the IP strategy adopted for solving the problem.

2.1 Spatial discretization

In [38], the authors considered two different discretizations of the domain Ω, which is assumed to be polygonal, both admissible for Two Point Flux Approximation (TPFA) finite volumes, according to [24, Definition 9.1]. At each time $t \in [0, 1]$, the variable $\rho(t)$ is discretized on a Delaunay triangulation, with the further hypothesis that only acute angles appear. The variable $\phi(t)$ is discretized on a finer grid obtained from the previous one by dividing each triangular cell into three quadrilateral cells, joining the edges midpoints to the triangle's circumcenter [38, Figure 1]. Both meshes consist of two sets $(\mathcal{T}, \mathcal{E})$, the set of cells c and edges e, respectively. To distinguish the coarser mesh from the finer one, we denote the former by $(\mathcal{T}', \mathcal{E}')$, with the same notation for all its elements. Due to the no-flux boundary condition, boundary edges are not relevant to the discrete model. We will then consider, by convention, the sets $\mathcal{E}$ and $\mathcal{E}'$ without boundary edges. Let us denote by $N_{\mathcal{T}'}$ and $N_{\mathcal{E}'}$ the total number of cells and (internal) edges of the coarser mesh. The total number of cells and edges of the finer mesh are then $N_{\mathcal{T}} = 3N_{\mathcal{T}'}$ and $N_{\mathcal{E}} = 2N_{\mathcal{E}'} + 3N_{\mathcal{T}'}$.

We define

$$\boldsymbol{M} := \text{Diag}\left(|\boldsymbol{c}|\right) \ , \ |\boldsymbol{c}| := (|c_i|)_{i=1}^{N_{\mathcal{T}}},$$
$$\boldsymbol{M'} := \text{Diag}\left(|\boldsymbol{c'}|\right) \ , \ |\boldsymbol{c'}| := (|c_i{}'|)_{i=1}^{N_{\mathcal{T}'}},$$

where $|c_i|$ denotes the area of the triangle c_i. Since the discrete model involves two different spatial discretizations, we also introduce the injection operator $\boldsymbol{J} \in \mathbb{R}^{N_{\mathcal{T}}, N_{\mathcal{T}'}}$,

$$\boldsymbol{J}[i,j] = \begin{cases} 1 & \text{if } c_i \subset {c'}_j\,, \\ 0 & \text{else}\,. \end{cases} \tag{3}$$

The operator $\boldsymbol{J}$ injects piecewise constant functions from the coarse grid to the fine one. The operator $(\boldsymbol{M'})^{-1}\boldsymbol{J}^T\boldsymbol{M}$ is an averaging operator that acts in the opposite direction. Both operators are the identity when the same grid is used for ϕ and ρ.

In order to define the discrete differential operators associated to the finite volume discretization, we fix an arbitrary orientation on the set of edges $\mathcal{E}$ and define the matrix $\boldsymbol{E} \in \mathbb{R}^{N_{\mathcal{T}}, N_{\mathcal{E}}}$ given by

$$\boldsymbol{E}[i,k] = \begin{cases} 1 & \text{if cell } c_i \text{ is the left side of edge } e_k\,, \\ -1 & \text{if cell } c_i \text{ is the right side of edge } e_k\,. \end{cases}$$

The discrete gradient and divergence, $\boldsymbol{\nabla} \in \mathbb{R}^{N_{\mathcal{E}}, N_{\mathcal{T}}}$ and $\mathbf{div} \in \mathbb{R}^{N_{\mathcal{T}}, N_{\mathcal{E}}}$, are given by

$$\begin{aligned} \boldsymbol{\nabla} &= \boldsymbol{\nabla}_{\mathcal{E},\mathcal{T}} = \text{Diag}\left(|\boldsymbol{w}|\right)^{-1} \boldsymbol{E}^T, \\ \mathbf{div} &= \mathbf{div}_{\mathcal{T},\mathcal{E}} = -\boldsymbol{\nabla}^T \text{Diag}\left(|\boldsymbol{w}| \odot |\boldsymbol{e}|\right) = -\boldsymbol{E}\text{Diag}\left(|\boldsymbol{e}|\right), \end{aligned}$$

where $|\boldsymbol{w}| \in \mathbb{R}^{N_{\mathcal{E}}}$ and $|\boldsymbol{e}| \in \mathbb{R}^{N_{\mathcal{E}}}$ are the vectors of distances between the circumcenters of adjacent cells and the lengths of the edges, respectively (we used the symbol $\odot$ to denote the pointwise multiplication of vectors).

The TPFA finite volume discretization and the two-level grids require the introduction of two additional operators to match the dimension between the different spaces. The first is a reconstruction operator $\boldsymbol{R}_{\mathcal{E}} : \mathbb{R}^{N_{\mathcal{T}'}} \to \mathbb{R}^{N_{\mathcal{E}}}$, mapping a positive density $\boldsymbol{\rho}$ defined on the cells of the coarse grid to a variable defined on the edges of the finer one. In [38] this reconstruction first lifts the density $\boldsymbol{\rho}$ into the finer space via the operator $\boldsymbol{J}$ and then averages the values of adjacent cells. The authors considered two types of averages, a weighted arithmetic mean and a weighted harmonic mean (the latter is not considered in this paper to avoid complicating the exposition). Using the former, the linear reconstruction operator explicitly writes

$$(\boldsymbol{R}_{\mathcal{E}}\boldsymbol{\rho})_e := \lambda_e(\boldsymbol{J}\boldsymbol{\rho})_{c_i} + (1-\lambda_e)(\boldsymbol{J}\boldsymbol{\rho})_{c_j}$$

for the two cells c_i, c_j sharing the edge e, where λ_e is a weight that depends on the mesh geometry. Moreover, they introduced a further operator $\boldsymbol{R}_{\mathcal{T}} : \mathbb{R}^{N_{\mathcal{E}}} \to \mathbb{R}^{N_{\mathcal{T}'}}$ that maps the variables defined at the edges of the finer grid to the variables defined at the cells of the coarser one. To preserve the variational structure of the discrete problem, it is defined as

$$\boldsymbol{R}_{\mathcal{T}} := \left(\boldsymbol{R}_{\mathcal{E}}\right)^T \text{Diag}\left(|\boldsymbol{w}| \odot |\boldsymbol{e}|\right).$$

2.2 Temporal discretization

In [38], the temporal discretization is based on staggered temporal grids. The time interval $[0,1]$ is divided into $K+1$ sub-intervals $[t^k, t^{k+1}]$ of equal length $\Delta t = 1/(K+1)$, for $k = 0, \ldots, K \geq 1$, with $t^0 = 0$ and $t^{K+1} = 1$. The variables ρ and s are defined at time t^k for $k = 1, \ldots, K$, while ϕ is discretized at each instant $(t^k + t^{k+1})/2$ for $k = 0, \ldots, K$.

Combining this temporal and spatial discretization, the discrete counterpart of the potential ϕ, the density ρ, and the slack variable s in equation eq. (2) are the vectors $\boldsymbol{\phi} \in \mathbb{R}^n, \boldsymbol{\rho} \in \mathbb{R}^m_{\geq 0}$, and $\boldsymbol{s} \in \mathbb{R}^m_{\geq 0}$ with $n = N_{\mathcal{T}}(K+1)$ and $m = N_{\mathcal{T}'}K$ given by

$$\begin{aligned} \boldsymbol{\phi} &= \left(\boldsymbol{\phi}^1; \ldots; \boldsymbol{\phi}^{K+1}\right),\ \boldsymbol{\phi}^k \in \mathbb{R}^{N_{\mathcal{T}}}, \\ \boldsymbol{\rho} &= \left(\boldsymbol{\rho}^1; \ldots; \boldsymbol{\rho}^K\right), \qquad \boldsymbol{\rho}^k \in \mathbb{R}^{N_{\mathcal{T}'}}_{\geq 0}, \\ \boldsymbol{s} &= \left(\boldsymbol{s}^1; \ldots; \boldsymbol{s}^K\right), \qquad \boldsymbol{s}^k \in \mathbb{R}^{N_{\mathcal{T}'}}_{\geq 0}, \end{aligned}$$

where we use the symbol ; to denote the concatenation of vectors. Moreover, we will denote by $\boldsymbol{x}^k$ the k-slice of a concatenated vector $\boldsymbol{x}$, which corresponds to its k-th time portion. The discrete counterparts of the initial and final densities are the two vectors $\boldsymbol{\rho}^0$ and $\boldsymbol{\rho}^{K+1}$ in $\mathbb{R}^{N_{\mathcal{T}'}}$ given by

$$\rho_i^0 = \frac{1}{|{c_i}'|}\int_{{c_i}'} \rho^{\text{in}}\,dx\,,\ \rho_i^{K+1} = \frac{1}{|{c_i}'|}\int_{{c_i}'} \rho^{\text{f}}\,dx, \quad i = 1, \ldots, N_{\mathcal{T}'}\,.$$

2.3 Discrete nonlinear system of equations

With the aforementioned discretization, the discrete counterpart of the nonlinear system eq. (2) is given by

$$\begin{aligned} F_{\boldsymbol{\phi}}(\boldsymbol{\phi},\boldsymbol{\rho}) &:= \left(F_{\boldsymbol{\phi}}^1;\dots;F_{\boldsymbol{\phi}}^{K+1}\right) &&= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}}(K+1)}\,, \\ F_{\boldsymbol{\rho}}(\boldsymbol{\phi},\boldsymbol{\rho},\boldsymbol{s}) &:= \left(F_{\boldsymbol{\rho}}^1;\dots;F_{\boldsymbol{\rho}}^{K}\right) &&= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}'}K}\,, \\ F_{\boldsymbol{s}}(\boldsymbol{\rho},\boldsymbol{s}) &:= \left(F_{\boldsymbol{s}}^1;\dots;F_{\boldsymbol{s}}^{K}\right) &&= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}'}K}\,, \end{aligned} \tag{4}$$

where for $k = 1,\dots,K+1$ the functions $F_{\boldsymbol{\phi}}^k$ are given by

$$F_{\boldsymbol{\phi}}^k(\boldsymbol{\phi},\boldsymbol{\rho}) = -\boldsymbol{M}\boldsymbol{J}\left(\tfrac{\boldsymbol{\rho}^k-\boldsymbol{\rho}^{k-1}}{\Delta t}\right) - \mathbf{div}\left(\boldsymbol{R}_{\mathcal{E}}\left(\tfrac{\boldsymbol{\rho}^k+\boldsymbol{\rho}^{k-1}}{2}\right)\odot\boldsymbol{\nabla}\boldsymbol{\phi}^k\right), \tag{5}$$

while for $k = 1,\dots,K$ the functions $F_{\boldsymbol{\rho}}^k$ and $F_{\boldsymbol{s}}^k$ are given by

$$F_{\boldsymbol{\rho}}^k(\boldsymbol{\phi},\boldsymbol{\rho},\boldsymbol{s}) = \boldsymbol{M}'\boldsymbol{J}^T\left(\tfrac{\boldsymbol{\phi}^{k+1}-\boldsymbol{\phi}^k}{\Delta t}\right) + \tfrac{1}{4}\boldsymbol{R}_{\mathcal{T}}\left(\left(\boldsymbol{\nabla}\boldsymbol{\phi}^k\right)^2 + \left(\boldsymbol{\nabla}\boldsymbol{\phi}^{k+1}\right)^2\right) + \boldsymbol{M}'\boldsymbol{s}^k, \tag{6}$$

$$F_{\boldsymbol{s}}^k(\boldsymbol{\rho},\boldsymbol{s}) = \boldsymbol{\rho}^k\odot\boldsymbol{s}^k, \tag{7}$$

with the additional (component-wise) requirement $\boldsymbol{\rho},\boldsymbol{s}\geq\mathbf{0}$. System eq. (4) is the system of optimality conditions for the discrete counterpart of the problem in eq. (1).

Remark 1. *Thanks to the finite volume discretization, the conservative structure of the continuity equation is preserved. Then, due to the no-flux boundary conditions, which are encoded explicitly in the divergence operator, the equation $F_{\boldsymbol{\phi}}^k(\boldsymbol{\phi},\boldsymbol{\rho}) = \mathbf{0}$ implies*

$$\mathbf{1}^T F_{\boldsymbol{\phi}}^k = |\boldsymbol{c}|^T\boldsymbol{J}\left(\boldsymbol{\rho}^k-\boldsymbol{\rho}^{k-1}\right) = |\boldsymbol{c}'|^T\left(\boldsymbol{\rho}^k-\boldsymbol{\rho}^{k-1}\right) = 0 \quad \forall k = 1,\dots,K\,.$$

Hence, at each intermediate time step, the discrete mass $|\boldsymbol{c}'|^T\boldsymbol{\rho}^k$ is preserved and is equal to the mass of the discrete initial and final densities.

2.4 Interior Point method

Due to the IP strategy, the discrete optimization problem is perturbed by adding a logarithmic barrier function,

$$-\mu\sum_{k=1}^{K}\Delta t\sum_{i=1}^{N_{\mathcal{T}'}}\log(\rho_i^k)|c_i|$$

tuned by a parameter $\mu>0$. This turns the system of optimality conditions eq. (4) into

$$\begin{aligned} F_{\boldsymbol{\phi}}(\boldsymbol{\phi},\boldsymbol{\rho}) &= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}}(K+1)}\,, \\ F_{\boldsymbol{\rho}}(\boldsymbol{\phi},\boldsymbol{\rho},\boldsymbol{s}) &= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}'}K}\,, \\ F_{\boldsymbol{s}}(\boldsymbol{\rho},\boldsymbol{s}) - \mu\mathbf{1} &= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}'}K}\,, \end{aligned} \tag{8}$$

where the complementarity constraint between ρ and s in eq. (7) is now relaxed. In this way, the two variables are forced to be strictly positive, making the problem easier to solve by using Newton methods.

The solution of the original problem is recovered in the limit $\mu\to 0$ and the parameter μ directly provides a bound on the sub-optimality of the discrete solution of eq. (8) for the original unperturbed problem [38, Section 4]. From the practical point of view, it is computed via a continuation method: for each value of μ^l of a sequence of relaxation parameters $(\mu^l)_{l\geq 0}$, the problem is solved with a Newton method initialized with the previously computed solution. We refer to [38] for the precise set up of the algorithm. We stress only that for each value of μ, problem eq. (8) is solved up to the relatively low tolerance $1e-6$ in order to obtain a more robust implementation. Furthermore, within each Newton cycle, we ensure that any Newton update does not introduce negative entries in the vectors $\boldsymbol{\rho}$ and $\boldsymbol{s}$ with a simple line-search procedure.

3 The linear algebra problem and the KKT system

The nonlinear system of equations eq. (8) is solved using an inexact Newton method. Each Newton iteration requires the solution of a linear system (referred to as KKT system) in the form

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T & \\ \boldsymbol{\mathcal{B}} & & \boldsymbol{\mathcal{M}}' \\ & \mathrm{Diag}\,(\boldsymbol{s}) & \mathrm{Diag}\,(\boldsymbol{\rho}) \end{pmatrix}\begin{pmatrix}\boldsymbol{\delta\phi}\\ \boldsymbol{\delta\rho}\\ \boldsymbol{\delta s}\end{pmatrix} = \begin{pmatrix}\boldsymbol{f}\\ \boldsymbol{g}\\ \boldsymbol{h}\end{pmatrix} = -\begin{pmatrix}\boldsymbol{F_\phi}\\ \boldsymbol{F_\rho}\\ \boldsymbol{F_s}-\mu\mathbf{1}\end{pmatrix}, \tag{9}$$

where the block matrix in equation eq. (9) is the Jacobian matrix of $(\boldsymbol{F_\phi}; \boldsymbol{F_\rho}; \boldsymbol{F_s} - \mu\mathbf{1})$. We denote by $\boldsymbol{\mathcal{M}'}$ and $\boldsymbol{\mathcal{M}}$ the matrices given by

$$\boldsymbol{\mathcal{M}'} = \text{Block Diag}\left((\boldsymbol{M}')_{k=1}^{K}\right), \quad \boldsymbol{\mathcal{M}} = \text{Block Diag}\left((\boldsymbol{M})_{k=1}^{K+1}\right).$$

The matrices $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{B}}^T$ in eq. (9) are the finite-dimensional version of the following differential operators (up to a multiplication by a mass matrix)

$$\boldsymbol{\mathcal{A}} \approx -\operatorname{div}(\rho\nabla), \quad \boldsymbol{\mathcal{B}} \approx \partial_t + \nabla\phi\cdot\nabla, \quad \boldsymbol{\mathcal{B}}^T \approx -\partial_t - \operatorname{div}(\cdot\,\nabla\phi).$$

According to the discretization described in section 2, matrix $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n,n}$ is a symmetric block diagonal matrix with $K+1$ blocks. Each block is a weighted Laplacian matrix given by

$$\boldsymbol{A}^k = -\mathbf{div}\text{Diag}\left(\tilde{\boldsymbol{\rho}}^k\right)\boldsymbol{\nabla} \in \mathbb{R}^{N_\mathcal{T}, N_\mathcal{T}}, \quad \tilde{\boldsymbol{\rho}}^k := \boldsymbol{R_\mathcal{E}}\left(\frac{\boldsymbol{\rho}^k + \boldsymbol{\rho}^{k-1}}{2}\right),$$

which we can write equivalently as $\boldsymbol{\mathcal{A}} = -\boldsymbol{\mathcal{D}iv_x}\text{Diag}\left((\tilde{\boldsymbol{\rho}}^1; \ldots; \tilde{\boldsymbol{\rho}}^k; \ldots; \tilde{\boldsymbol{\rho}}^{K+1})\right)\boldsymbol{\mathcal{D}_x}$ where

$$\boldsymbol{\mathcal{D}iv_x} = \text{Block Diag}\left((\mathbf{div})_{k=1}^{K+1}\right), \quad \boldsymbol{\mathcal{D}_x} = \text{Block Diag}\left((\boldsymbol{\nabla})_{k=1}^{K+1}\right).$$

Matrix $\boldsymbol{\mathcal{B}}$ is a block bi-diagonal matrix in $\mathbb{R}^{m,n}$ given by

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{J}}^T\boldsymbol{\mathcal{D}_t}\boldsymbol{\mathcal{M}} + \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{D}_x},$$

where the matrices $\boldsymbol{\mathcal{D}_t} \in \mathbb{R}^{KN_\mathcal{T}, (K+1)N_\mathcal{T}}$, $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{KN_{\mathcal{T}'}, (K+1)N_{\mathcal{T}'}}$, $\boldsymbol{\mathcal{J}} \in \mathbb{R}^{(K+1)N_\mathcal{T}, (K+1)N_{\mathcal{T}'}}$, and $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{(K+1)N_{\mathcal{T}'}, (K+1)N_{\mathcal{E}'}}$ are

$$\boldsymbol{\mathcal{D}_t} = \frac{1}{\Delta t}\begin{pmatrix} -\boldsymbol{I}_{N_\mathcal{T}} & \boldsymbol{I}_{N_\mathcal{T}} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{I}_{N_\mathcal{T}} & \boldsymbol{I}_{N_\mathcal{T}} \end{pmatrix}, \quad \boldsymbol{\mathcal{J}} = \text{Block Diag}\left((\boldsymbol{J})_{k=1}^{K+1}\right), \tag{10}$$

$$\boldsymbol{\mathcal{H}} = \frac{1}{2}\begin{pmatrix} \boldsymbol{I}_{N_{\mathcal{T}'}} & \boldsymbol{I}_{N_{\mathcal{T}'}} & & \\ & \ddots & \ddots & \\ & & \boldsymbol{I}_{N_{\mathcal{T}'}} & \boldsymbol{I}_{N_{\mathcal{T}'}} \end{pmatrix}, \quad \boldsymbol{\mathcal{G}} = \text{Block Diag}\left((\boldsymbol{G}^k)_{k=1}^{K+1}\right), \tag{11}$$

with $(\boldsymbol{G}^k)_{k=1,\ldots,K+1} \in \mathbb{R}^{N_{\mathcal{T}'}, N_\mathcal{E}}$ given by

$$\boldsymbol{G}^k := \boldsymbol{R_\mathcal{T}}\text{Diag}\left(\boldsymbol{\nabla}\boldsymbol{\phi}^k\right).$$

Remark 2. *The kernel of the block diagonal matrix $\boldsymbol{\mathcal{A}}$ has dimension $K+1$, since each block $\boldsymbol{A}^k$ has the constant vectors as kernel. On the other hand, Ker($\boldsymbol{\mathcal{A}}$) is included in Ker($\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{D}_x}$) and therefore*

$$\textit{Ker}(\boldsymbol{\mathcal{A}}) \cap \textit{Ker}(\boldsymbol{\mathcal{B}}) = \textit{Ker}(\boldsymbol{\mathcal{A}}) \cap \textit{Ker}(\boldsymbol{\mathcal{J}}^T\boldsymbol{\mathcal{D}_t}\boldsymbol{\mathcal{M}}) = \langle\mathbf{1}\rangle \in \mathbb{R}^n,$$

thus the Jacobian matrix is singular. This singularity can be removed grounding one entry of the solution $\boldsymbol{\delta\phi}$ or via regularization techniques [13]. However, we prefer avoiding these approaches in this paper, since iterative methods work also when dealing with singular matrices [49, 22].

3.1 Reduction to a saddle point linear system

In order to solve the linear system in eq. (9) it is convenient to eliminate the variable $\boldsymbol{\delta s}$ writing

$$\boldsymbol{\delta s} = (\text{Diag}\,(\boldsymbol{\rho}))^{-1}(\boldsymbol{h} - \text{Diag}\,(\boldsymbol{s})\,\boldsymbol{\delta\rho})$$

and reduce it into the following saddle point system

$$\mathcal{J}\begin{pmatrix}\boldsymbol{\delta\phi}\\ \boldsymbol{\delta\rho}\end{pmatrix} = \begin{pmatrix}\boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T\\ \boldsymbol{\mathcal{B}} & -\boldsymbol{\mathcal{C}}\end{pmatrix}\begin{pmatrix}\boldsymbol{\delta\phi}\\ \boldsymbol{\delta\rho}\end{pmatrix} = \begin{pmatrix}\boldsymbol{f}\\ \tilde{\boldsymbol{g}}\end{pmatrix}, \tag{12}$$

where the matrix $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{m,m}$ and the vector $\tilde{\boldsymbol{g}} \in \mathbb{R}^m$ are given by

$$\boldsymbol{\mathcal{C}} := \boldsymbol{\mathcal{M}'}\text{Diag}\,(\boldsymbol{\rho})^{-1}\,\text{Diag}\,(\boldsymbol{s}), \quad \tilde{\boldsymbol{g}} := \boldsymbol{g} - \boldsymbol{\mathcal{M}'}\text{Diag}\,(\boldsymbol{\rho})^{-1}\,\boldsymbol{h}.$$

The matrix $\mathcal{J}$ in eq. (12) has typically higher conditioning number than the Jacobian matrix in eq. (9), in particular when approaching the optimal solution, i.e. as $\boldsymbol{s} \odot \boldsymbol{\rho} \approx \mu \mathbf{1} \to \mathbf{0}$. However, designing an efficient preconditioner for the fully coupled system in eq. (9) can be harder than for the standard saddle point linear system in eq. (12), for which different preconditioning approaches are described in [9, 12, 37].

We adopt an Inexact Newton approach, which means that we seek for a solution $(\boldsymbol{\delta\phi}, \boldsymbol{\delta\rho})$ such that

$$\|\mathcal{J}(\boldsymbol{\delta\phi}; \boldsymbol{\delta\rho}) - (\boldsymbol{f}; \tilde{\boldsymbol{g}})\| \leq \varepsilon_{\text{out}} \|(\boldsymbol{f}; \boldsymbol{g}; \boldsymbol{h})\| \,.$$

The linear system residual is scaled by the right-hand side of the original system to avoid over-solving issues we faced using $\|(\boldsymbol{f}; \tilde{\boldsymbol{g}})\|$. In our experiments we use a fixed tolerance $\varepsilon_{\text{out}} = 1e-5$. This value may be quite small compared to the literature for an inexact Newton approach (for example [7, 31]), but we want to avoid more aggressive inexact strategies which could undermine the effectiveness of the nonlinear solver and focus on the linear algebra only.

Remark 3. *From remark 1, we know the solution $\boldsymbol{\rho}$ to equation eq.* (8) *must satisfy $|\boldsymbol{c'}|^T \boldsymbol{\rho}^k = |\boldsymbol{c'}|^T \boldsymbol{\rho}^0$ for all $k = 1, \dots K$. If this condition is satisfied for the starting point of the Newton scheme, then the increment $\boldsymbol{\delta\rho}$ solution to equations* (9) *satisfies*

$$|\boldsymbol{c'}|^T \boldsymbol{\delta\rho}^k = 0 \quad \forall k = 1, \dots K \,,$$

at each Newton iteration. To see this, sum both sides of the first block of equations of system (12)*, for each time step k, and use the no-flux boundary conditions in the divergence operator. Due to our inexact Newton steps, this condition is not verified exactly, but only up to the tolerance chosen. We impose it by renormalizing $\boldsymbol{\rho}$ after each Newton update.*

4 Preconditioning approaches and numerical results

In this section we present three preconditioners to solve the linear system in eq. (12) and a series of numerical experiments used to measure their performance. All of these preconditioners involve the usage of inexact inner solvers, therefore, they may not be linear and must be applied as right preconditioners within a FGMRES cycle [46]. In all cases, the inner solver we adopted is the AGgregation-based Algebraic MultiGrid (AGMG) method described in [40].

We adopt three measures to evaluate the preconditioner performance for each Newton cycle associated with each IP iteration:

1. **Outer/Lin.sys.**: the average number of outer FGMRES iterations. It is calculated as the number of outer iterations divided by the number of linear systems solved for each Newton cycle.
2. **CPU/Lin.sys.**: the average CPU time (measured in seconds) required for the solution of the linear systems. It is computed as the total CPU time required to solve all linear systems within a Newton cycle divided by the number of linear systems solved (all experiments were conducted single core, on a machine equipped with an Intel® Xeon® 2.8 GHz CPU and 128 GiB of RAM memory).
3. **Inner/Outer**: the average number of inner iterations per preconditioner application. It is computed by dividing the cumulative number of AGMG iterations required to solve the linear systems involved in the preconditioner application (for each preconditioner, we will specify the linear systems to which we will refer) by the total number of outer iterations for each Newton cycle.

4.1 Test cases

The IP algorithm and the linear solver strategies described in the next sections are tested on four numerical experiments adapted from [38, Section 5.2]. The first three are set in $\Omega = [0,1]^2$ whereas the fourth one, three dimensional, in $\Omega = [0,1]^3$. They are:

1. Gaussian densities ρ^{in} and ρ^{f} centered at $(0.3, 0.3)$ and $(0.7, 0.7)$ respectively with variance 0.1. This ensures that ρ^{in} and ρ^{f} are lower bounded by $1e-3$.
2. Translation of a compactly supported smooth sinusoidal density ρ^{in}.
3. Compression of a compactly supported smooth sinusoidal density ρ^{in}. This test case has been shown experimentally in [38] to exhibit severe instabilities in the discrete solution, and this, in turn, has a negative effect on the discrete solver.
4. Same as test 1 but in three dimensions. The Gaussian densities ρ^{in} and ρ^{f} are centered at $(0.3, 0.3, 0.3)$ and $(0.7, 0.7, 0.7)$.

Test cases 1, 2 and 3 are discretized using four different meshes, taken from [1], having $N_{\mathcal{T}'} = 224, 896, 3584, 14336$ cells, where each mesh is a refinement of the other. These meshes will be denoted by $\mathcal{T}^0, \mathcal{T}^1, \mathcal{T}^2, \mathcal{T}^3$. The corresponding number of subcells is $N_{\mathcal{T}} = 672, 2688, 10752, 43008$. For each grid, we consider $\Delta t = 1/(K+1)$ with $K+1 = 16, 32, 64, 128$. The typical mesh size h of the mesh $\mathcal{T}^0$ is approximately $1/16$, so that the diagonal pairing of the space and time steps provides a uniform discretization of the time-space domain $[0,1] \times \Omega$. The total number of ϕ and ρ degrees of freedom for these sequence of problems is $n + m \approx (1.4e4, 1.1e5, 9.1e5, 7.3e6)$, which is the size of the linear system in eq. (12). The last test case is solved on Cartesian grids with cubic cells. In this case the operator $\boldsymbol{J}$ ineq. (3) is simply the identity. We consider 2 refinements (both in time and in space) of an initial experiment set on a $8 \times 8 \times 8$ grid and using $\Delta t = 1/(K+1)$ with $K+1 = 8$. In this case, the total number of ϕ and ρ degrees of freedom for a uniform time-space refinement is $n + m \approx (7.7e3, 1.3e5, 2.0e6)$.

Convergence is achieved when the relaxation parameter μ is below $1e-6$, which corresponds to ten iterations of IP, with the choice of parameters described in [38]. The nonlinear system eq. (8) is solved with rather tight tolerance $1e-6$ in order to have a robust implementation for the IP solver. Each IP step requires between 3 and 7 inexact Newton iterations. These numbers are practically insensitive to the mesh and the time step adopted. The total number of Newton steps (which corresponds to the number of linear systems to be solved) ranges between 40 and 50. The preconditioning approaches used to solve these linear systems are described in the following sections.

For each preconditioner, we will present a table reporting the metrics mentioned above, while the relaxation parameter μ is reduced for all combinations of spatial and temporal discretizations. We will show these detailed results only for the second test case, since it captures the main challenges involved in the linear algebra problems. The numerical results for the other three cases will be summarized in section 4.5, where we will compare the performance of the preconditioners in terms of total CPU time.

The code is written in MATLAB® and its source code is available at the link https://github.com/gptod/OT-FV, where it is possible to reproduce the experiments presented in this paper. The AGMG solver is interfaced via MEX®.

4.2 Preconditioner based on the primal Schur complement

In this section we consider a preconditioner for the linear system in eq. (12) based on the primal Schur complement $\boldsymbol{\mathcal{S}}_p = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}^T \boldsymbol{\mathcal{C}}^{-1} \boldsymbol{\mathcal{B}}$, similarly to the approach used in [25] for the solution of the optimal transport problem on graphs with cost equal to the shortest path distance. It is given by

$$\boldsymbol{P}^{-1} = \begin{pmatrix} \boldsymbol{I} & \\ \boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{B}} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\mathcal{S}}_p^{-1} & \\ & -\boldsymbol{\mathcal{C}}^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{C}}^{-1} \\ & \boldsymbol{I} \end{pmatrix}. \tag{13}$$

The application of the preconditioner in eq. (13) requires the inversion of $\boldsymbol{\mathcal{C}}$, which is diagonal, and the (approximate) solution of the linear system

$$\boldsymbol{\mathcal{S}}_p \boldsymbol{x} = \boldsymbol{b}\,, \tag{14}$$

with $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$. Using this preconditioner is essentially equivalent to reducing the linear system in eq. (12) to the variable $\boldsymbol{\delta\phi}$ only, an approach referred to in the literature as fully reduced [10] or condensed [18]. The matrix $\boldsymbol{\mathcal{S}}_p$ is a block tridiagonal matrix. It can be written as

$$\begin{aligned} \boldsymbol{\mathcal{S}}_p = \overbrace{-\boldsymbol{\mathcal{D}iv_x}\mathrm{Diag}\,(\tilde{\boldsymbol{\rho}})\,\boldsymbol{\mathcal{D}_x}}^{\boldsymbol{\mathcal{A}}} \overbrace{-\boldsymbol{\mathcal{D}iv_x}\boldsymbol{\mathcal{G}}^T\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{D}_x}}^{=:\boldsymbol{\mathcal{S}}_{xx}} + \overbrace{\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}_t}^T\boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{J}}^T\boldsymbol{\mathcal{D}_t}\boldsymbol{\mathcal{M}}}^{=:\boldsymbol{\mathcal{S}}_{tt}} \\ + \underbrace{\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}_t}^T\boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{D}_x} + (\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}_t}^T\boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{D}_x})^T}_{=:\boldsymbol{\mathcal{S}}_{tx}+\boldsymbol{\mathcal{S}}_{tx}^T}, \end{aligned}$$

and it may be seen as the discretization of an elliptic time-space operator. In fact, it is the sum of a ρ-weighted spatial Laplacian matrix $\boldsymbol{\mathcal{A}}$, an anisotropic spatial Laplacian $\boldsymbol{\mathcal{S}}_{xx} \approx -\operatorname{div}(\rho/s\,\nabla\phi \otimes \nabla\phi\,\nabla)$, a weighted temporal Laplacian with Neumann boundary condition $\boldsymbol{\mathcal{S}}_{tt}$, and a time-space operator $\boldsymbol{\mathcal{S}}_{tx} + \boldsymbol{\mathcal{S}}_{tx}^T$. We adopt the AGMG solver to solve the linear system in eq. (14), with relative accuracy $\varepsilon_{\text{in}} = 1e-1 > \varepsilon_{\text{out}} = 1e-5$. The value of the inner tolerance ε_{in} has been determined experimentally to distribute the work load between the inner and outer solver in a balanced way.

In table 1 we summarize the results obtained using the preconditioner presented in this section for different combinations of time and space discretizations. We use the metrics described in section 4, where the average number of inner iterations, denoted by **Inner/Outer**, refers to the linear system in eq. (14).

The number of average outer iterations, denoted by **Outer/Lin.sys.**, remains practically constant only for the coarsest and most uniform combinations of grids and time steps. However, in some cases, also **Outer/Lin.sys.** increases when

$K+1$	16	32	64	128	16	32	64	128	16	32	64	128
μ	**Outer/Lin.sys.**				**CPU/Lin.sys.**				**Inner/Outer**			
$\mathcal{T}^0(N_\mathcal{T}=224, N_\mathcal{T}'=672)$ #DOF=(1.4e4, 2.8e4, 5.7e4, 1.1e5)												
1	4	4	4	2	1.8e-1	3.1e-1	9.7e-1	2.3e1	2.2	2.1	2.7	1.5
2e-1	4	4	3	1	2.1e-1	6.1e-1	5.8e0	1.9e1	2.7	2.3	2.6	1.0
4e-2	5	5	4	1	3.3e-1	1.1e0	7.4e0	2.0e1	1.9	2.3	1.9	1.0
8e-3	6	4	4	1	4.8e-1	1.7e0	9.0e0	1.8e1	1.6	2.0	1.8	1.0
2e-3	5	5	4	2	5.9e-1	1.9e0	8.8e0	2.0e1	1.6	1.8	1.8	1.1
3e-4	5	4	4	2	6.4e-1	2.1e0	8.4e0	1.9e1	1.7	2.0	1.7	1.2
6e-5	5	4	4	2	6.8e-1	2.1e0	8.8e0	2.2e1	1.7	1.8	2.0	1.2
1e-5	4	4	4	2	7.0e-1	2.2e0	9.1e0	2.4e1	1.7	2.1	1.9	1.2
3e-6	4	4	4	2	7.1e-1	2.4e0	7.9e0	2.5e1	1.8	1.8	1.7	1.0
5e-7	4	4	4	2	7.8e-1	2.5e0	9.2e0	2.3e1	1.7	1.9	1.9	1.1
	4.5	4.2	4.0	1.7	4.9e-1	1.6e0	7.4e0	2.1e1	1.9	2.0	2.0	1.1
$\mathcal{T}^1(N_\mathcal{T}=896, N_\mathcal{T}'=2688)$ #DOF=(5.6e4, 1.1e5, 2.3e5, 4.6e5)												
1	4	4	4	4	5.7e-1	1.4e0	2.6e0	1.4e1	2.7	2.3	2.1	3.5
2e-1	3	3	3	3	6.5e-1	1.7e0	1.2e1	5.8e2	2.7	2.7	2.9	2.9
4e-2	5	5	5	4	1.1e0	3.8e0	5.1e1	9.0e2	2.6	2.1	2.3	2.2
8e-3	5	6	5	4	1.9e0	8.6e0	1.0e2	1.1e3	3.3	2.1	1.9	1.7
2e-3	7	6	5	4	3.4e0	1.3e1	1.4e2	1.0e3	6.0	2.0	1.8	1.6
3e-4	6	6	5	4	6.3e0	1.5e1	1.5e2	1.0e3	14.6	2.1	2.0	1.9
6e-5	6	5	4	4	1.1e1	1.8e1	1.6e2	1.0e3	32.2	2.5	2.1	1.9
1e-5	6	5	4	4	3.0e1	1.9e1	1.7e2	1.1e3	95.0	2.5	2.1	1.8
3e-6	6	5	4	4	4.2e1	2.2e1	1.7e2	1.0e3	122.8	2.3	1.9	1.9
5e-7	9	5	4	4	7.9e1	2.7e1	1.9e2	1.1e3	151.4	2.5	2.1	2.7
	5.5	4.8	4.3	3.7	1.5e1	1.2e1	1.0e2	8.6e2	49.0	2.3	2.1	2.2
$\mathcal{T}^2(N_\mathcal{T}=3584, N_\mathcal{T}'=10752)$ #DOF=(2.3e5, 4.6e5, 9.1e5, 1.8e6)												
1	3	3	4	4	2.3e0	5.0e0	1.0e1	2.2e1	3.2	2.9	2.4	2.3
2e-1	3	3	3	3	2.9e0	6.3e0	1.5e1	2.1e3	3.4	3.2	3.1	3.3
4e-2	4	4	5	9	4.9e0	2.6e1	2.1e2	1.3e3	4.0	4.2	2.5	128.5
8e-3	5	5	6	20	1.1e1	9.6e1	1.2e3	2.0e3	4.2	5.0	2.4	160.1
2e-3	6	6	6	55	1.9e1	1.3e2	1.5e3	6.2e3	3.1	7.7	2.0	198.8
3e-4	6	8	6	†	2.5e1	2.0e2	2.3e3	†	3.3	18.1	2.2	†
6e-5	6	6	4	†	3.1e1	3.8e2	2.2e3	†	6.4	65.2	2.6	†
1e-5	6	7	4	†	4.5e1	7.1e2	1.7e3	†	13.1	133.8	2.7	†
3e-6	6	9	4	†	7.3e1	1.1e3	1.7e3	†	28.9	161.5	2.9	†
5e-7	5	16	4	†	1.3e2	2.1e3	1.8e3	†	70.9	179.3	2.6	†
	4.7	6.3	4.6	†	2.7e1	4.1e2	1.1e3	†	12.2	79.4	2.5	†
$\mathcal{T}^3(N_\mathcal{T}=14336, N_\mathcal{T}'=43008)$ #DOF=(9.0e5, 1.8e6, 3.7e6, 7.3e6)												
1	3	3	3	3	9.3e0	2.0e1	5.0e1	9.8e1	3.5	3.4	2.9	2.8
2e-1	2	3	3	3	1.1e1	2.5e1	6.0e1	4.7e2	4.0	3.9	3.6	3.3
4e-2	4	4	4	7	2.7e1	1.5e2	3.4e3	1.8e3	5.6	4.6	22.9	84.2
8e-3	5	5	9	12	7.4e1	5.0e2	2.6e3	3.7e3	5.7	5.2	154.9	105.4
2e-3	6	7	18	26	1.4e2	2.4e3	4.9e3	1.1e4	4.5	3.3	182.5	152.1
3e-4	6	6	58	59	2.1e2	2.7e3	1.5e4	3.6e4	4.0	3.2	199.0	198.9
6e-5	5	6	†	162	2.6e2	2.1e3	†	9.5e4	4.3	4.1	†	200.0
1e-5	5	5	†	†	3.2e2	2.4e3	†	†	4.5	7.1	†	†
3e-6	5	5	†	†	3.5e2	2.7e3	†	†	4.4	14.9	†	†
5e-7	5	5	†	†	3.6e2	3.7e3	†	†	4.6	41.8	†	†
	4.3	4.7	†	†	1.5e2	1.4e3	†	†	4.6	7.9	†	†

Table 1: Numerical results using the preconditioner based on the primal Schur complement. Each sub-table reports the results for a given mesh, from the coarsest (top) to the finest (bottom), and for different time discretizations ($\Delta t = 1/(K+1)$). Each sub-table reports, while μ is reduced (leftmost column), the averaged outer iteration and CPU time per linear system (**Outer/Lin.sys.** and **CPU/Lin.sys.**) and the averaged inner iterations per outer iteration **Inner/Outer** for solving the linear system in eq. (14) (metrics defined in section 4). A final row summarizes the averages on the whole simulation. We highlighted in gray the time-space combination providing the uniform discretization. The † symbol denotes those IP steps where the linear solver failed.

the average number of inner iterations **Inner/Outer** get closer to 200 (the limit we fixed for the inner solver), which means that, in some cases, the inner solver did not reach the accuracy $\varepsilon_{\text{in}} = 1e-1$. For the cases with more degrees of freedom, this led to a huge CPU time (almost $1e5$ seconds per linear system) and to failures occurring as μ is reduced.

The average number of inner iterations **Inner/Outer** is strongly influenced by the relationship between the mesh and the time step $\Delta t = 1/(K+1)$. The phenomenon is particularly evident for mesh $\mathcal{T}^2$. For $K+1 = 32$, the average number of inner iterations increases progressively while μ is reduced, reaching ≈ 179 at the last IP iteration. For $K+1 = 128$, **Inner/Outer** increases reaching the 200 iterations limit. For $K+1 = 64$ instead, which corresponds to the most balanced time-space scaling, this number remains between 2.0 and 2.7. The same phenomenon occurs using $\mathcal{T}^1$, passing from $K+1 = 16$ to $K+1 = 32$. We attribute this to an improper construction of the coarse matrix sequence used by the multigrid solver due to the poor scaling of the temporal and spatial components of $\boldsymbol{\mathcal{S}}_p$. The AGMG is based on an aggregation-based coarsening approach, hence if these components are unbalanced the coarse matrices may not reflect the infinite-dimensional operators discretized by our problem, compromising the performance of the multigrid solver.

This phenomenon can only get worse in the last IP iterations due to presence of the term $\boldsymbol{\mathcal{C}}^{-1} = \boldsymbol{\mathcal{M}'}^{-1}\text{Diag}\,(\boldsymbol{\rho})\,\text{Diag}\,(\boldsymbol{s})^{-1}$ in $\boldsymbol{\mathcal{S}}_p = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{B}}^T$. In fact, the term $\boldsymbol{\mathcal{C}}^{-1}$ contains entries varying by several order of magnitude due to the relaxed complementarity condition $\boldsymbol{s} \odot \boldsymbol{\rho} \approx \mu\mathbf{1}$. This also means that, on those entries where $\boldsymbol{\rho}$ remain strictly positive, the anisotropic Laplacian $\boldsymbol{\mathcal{S}}_{xx}$ scales like $1/\mu$, making the linear system eq. (14) more difficult to solve since this anisotropic term becomes the dominant part of $\boldsymbol{\mathcal{S}}_p$. Similar matrices and analogous considerations can be found in [10, 11, 32, 37].

Furthermore, even in best-case scenarios, we experimented that the preprocessing phase is the most demanding in terms of CPU time. Typically, more than 50% of the time is spent building the sequence of coarse matrices, reaching peaks of 90% during the last IP steps using the finest discretization.

In order to cope with all the issues presented so far, we tried to neglect some components of $\boldsymbol{\mathcal{S}}_p$. We tried to remove the extra-diagonal or lower-diagonal part of $\boldsymbol{\mathcal{S}}_p$, or the time-space operator $\boldsymbol{\mathcal{S}}_{tx} + \boldsymbol{\mathcal{S}}_{tx}^T$ so that the remaining terms form a weighted time-space Laplacian (similarly to what done in [32]). However, none of these approaches worked. All numerical experiments suggest that none of these terms can be neglected without altering the spectral property of the matrix, and consequently affecting the effectiveness of the preconditioner.

Summarizing the results obtained in this section, the preconditioner based on the primal Schur complement $\boldsymbol{\mathcal{S}}_p = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{C}}^{-1}\boldsymbol{\mathcal{B}}$ lacks both in efficiency and in robustness, due to the complexity of the PDE behind the linear system in eq. (14), in particular when the relaxation parameter μ goes to zero.

4.3 The SIMPLE preconditioner

In this section, we recall the classical SIMPLE preconditioner described in [42, 43] and based on the approximation of the inverse of the dual Schur complement $\boldsymbol{\mathcal{S}} = -(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^{-1}\boldsymbol{\mathcal{B}}^T)$ (we recall that, formally, we cannot write $\boldsymbol{\mathcal{A}}^{-1}$ since the matrix $\boldsymbol{\mathcal{A}}$ is singular). The SIMPLE preconditioner is given by

$$\boldsymbol{P}^{-1} = \begin{pmatrix} \boldsymbol{I} & -\hat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{B}}^T \\ & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\mathcal{A}}}^{-1} & \\ & \hat{\boldsymbol{\mathcal{S}}}^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \\ -\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1} & \boldsymbol{I} \end{pmatrix}$$

where

$$\hat{\boldsymbol{\mathcal{A}}} = \text{Diag}\,(\boldsymbol{\mathcal{A}})\,, \quad \hat{\boldsymbol{\mathcal{S}}} = -(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{B}}^T)\,.$$

This preconditioner is referred to as constraint preconditioner in the literature studying linear solvers for IP methods [37]. The main computational cost of applying this preconditioner is to solve linear systems in the form

$$\hat{\boldsymbol{\mathcal{S}}}\boldsymbol{y} = \boldsymbol{c}\,. \tag{15}$$

The matrix $\hat{\boldsymbol{\mathcal{S}}}$ can be formed explicitly and is block tridiagonal.

During the latest steps of the IP method, when $\mu \to 0$, linear system eq. (15) may become ill-conditioned since the diagonal matrix $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{M}'}\text{Diag}\,(\boldsymbol{\rho})^{-1}\,\text{Diag}\,(\boldsymbol{s})$ can contain terms that vary by several orders of magnitude, since $\boldsymbol{\rho} \odot \boldsymbol{s} \approx \mu\mathbf{1} \to \mathbf{0}$. To cope with this issue, we scale both sides of eq. (15) by $\text{Diag}\,(\boldsymbol{\rho})$ and we solve the linear system

$$-(\boldsymbol{\mathcal{M}'}\text{Diag}\,(\boldsymbol{s}) + \text{Diag}\,(\boldsymbol{\rho})\,\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{B}}^T)\boldsymbol{y} = \text{Diag}\,(\boldsymbol{\rho})\,\boldsymbol{c}\,. \tag{16}$$

This scaling may seem a dangerous procedure if small entries appears in $\boldsymbol{\rho}$, however it is used within the preconditioner application, where the potential inaccuracies do not affect the convergence of FGMRES to the solution of the linear

system. We use the AGMG solver with tolerance $\varepsilon_{\text{in}} = 1e-1$ to solve the linear system in eq. (16) and denote by $\hat{\boldsymbol{\mathcal{S}}}^{-1}_{\varepsilon_{\text{in}}}$ the resulting (nonlinear) operator. We found experimentally that this value provided the best performance. Further reductions of ε_{in} led only to an increase of the number of inner iterations but no reduction in the outer loop iterations.

The results for the SIMPLE preconditioner are summarized in table 2. Despite its simplicity, the proposed preconditioner turned out to be robust (in particular for small values of μ) but not particularly efficient. Some failures occurred only at the initial IP steps using the finest grid. In this cases, we restarted the solver using the data obtained with the $\boldsymbol{BB}$-preconditioner in order to study the behavior of the SIMPLE preconditioner close to the optimal solution. The average number of inner iterations **Inner/Outer** increases only sightly as $\mu \to 0$, independently of the time step and the mesh size used or the IP step considered. The preprocessing time required by the AGMG solver is limited, approximately accounting for 1% of the CPU time spent in the linear system solution. The average number of outer iterations **Outer/Lin.sys.** is affected only mildly by the number of time steps $K+1$.

Remarkably, the preconditioner becomes more efficient as the IP relaxation term μ goes to zero. However, **Outer/Lin.sys.** more than doubles at each mesh refinement. Both phenomena can also be explained by looking at the structure of the preconditioned matrix, which is given by

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T \\ \boldsymbol{\mathcal{B}} & -\boldsymbol{\mathcal{C}} \end{pmatrix} \boldsymbol{P}^{-1} = \begin{pmatrix} \boldsymbol{I} + (\boldsymbol{\mathcal{A}}\hat{\boldsymbol{\mathcal{A}}}^{-1} - \boldsymbol{I})(\boldsymbol{I} + \boldsymbol{\mathcal{B}}^T\hat{\boldsymbol{\mathcal{S}}}^{-1}_{\varepsilon_{\text{in}}}\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}) & (\boldsymbol{I} - \boldsymbol{\mathcal{A}}\hat{\boldsymbol{\mathcal{A}}}^{-1})\boldsymbol{\mathcal{B}}^T\hat{\boldsymbol{\mathcal{S}}}^{-1}_{\varepsilon_{\text{in}}} \\ (\boldsymbol{I} - \hat{\boldsymbol{\mathcal{S}}}\hat{\boldsymbol{\mathcal{S}}}^{-1}_{\varepsilon_{\text{in}}})\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1} & \hat{\boldsymbol{\mathcal{S}}}\hat{\boldsymbol{\mathcal{S}}}^{-1}_{\varepsilon_{\text{in}}} \end{pmatrix},$$

and estimating its eigenvalues. In order to do this, let us first observe that the preconditioned matrix is approximately a block upper triangular matrix, whose bottom-right block becomes close to the identity plus a perturbation of order $\varepsilon_{\text{in}} = 1e-1$, hence there are m eigenvalues close to one.

The upper left block is the identity plus a perturbation matrix (which should be as small as possible to have an ideal preconditioner) given by product of two factors. The first factor is

$$\boldsymbol{\mathcal{A}}\hat{\boldsymbol{\mathcal{A}}}^{-1} - \boldsymbol{I} = \text{Block Diag}\left(\boldsymbol{A}^k \text{Diag}\left(\boldsymbol{A}^k\right)^{-1} - \boldsymbol{I}_{N_\mathcal{T}}\right).$$

Since the matrices $(\boldsymbol{A}^k)_{k=1,\dots,K+1}$ are weighted Laplacian matrices, a diagonal preconditioner becomes a poor preconditioner for large $\boldsymbol{A}^k$ (see [33] for precise estimates). Hence the largest eigenvalue of the first factor will increase while refining in space, but not in time, by the block diagonal structure of matrix $\boldsymbol{\mathcal{A}}$. The second factor, which we would like to keep as small as possible to compensate the first one, is $\boldsymbol{I} + \boldsymbol{Q}$ with $\boldsymbol{Q} = \boldsymbol{\mathcal{B}}^T\hat{\boldsymbol{\mathcal{S}}}^{-1}_{\varepsilon_{\text{in}}}\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}$. Using the formula in eq. (16) and the complementarity condition $\boldsymbol{\rho} \odot \boldsymbol{s} \approx \mu\mathbf{1}$, we can approximate it as follows:

$$\begin{aligned} \boldsymbol{I} + \boldsymbol{Q} &= \boldsymbol{I} - \boldsymbol{\mathcal{B}}^T(\boldsymbol{\mathcal{M}}'\text{Diag}\,(\boldsymbol{s}) + \text{Diag}\,(\boldsymbol{\rho})\,\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{B}}^T)^{-1}\text{Diag}\,(\boldsymbol{\rho})\,\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1} \\ &\approx \boldsymbol{I} - \boldsymbol{\mathcal{B}}^T(\boldsymbol{\mathcal{M}}'\mu + \text{Diag}\,(\boldsymbol{\rho})^2\,\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{B}}^T)^{-1}\text{Diag}\,(\boldsymbol{\rho})^2\,\boldsymbol{\mathcal{B}}\hat{\boldsymbol{\mathcal{A}}}^{-1}. \end{aligned}$$

If we set $\mu = 0$ in this formula, the resulting matrix has 0 or 1 eigenvalues since it is idempotent. This gives a qualitative explanation of why the SIMPLE preconditioner performs better as $\mu \to 0$. Nevertheless, this approach is not viable to tackle large problems due to the quadratic complexity with respect to the number of spatial degrees of freedom.

4.4 The BB-preconditioner

We present now the third preconditioner considered in this paper. Its design started from the idea of using the following block triangular preconditioner (see for example [9]),

$$\boldsymbol{P}_t = \begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T \\ & -\hat{\boldsymbol{\mathcal{S}}} \end{pmatrix},$$

where $\hat{\boldsymbol{\mathcal{S}}}$ should ideally equal the dual Schur complement $\boldsymbol{\mathcal{S}} = -(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^{-1}\boldsymbol{\mathcal{B}}^T)$ or some approximation of it (we again recall that, formally, we cannot write $\boldsymbol{\mathcal{A}}^{-1}$ since the matrix $\boldsymbol{\mathcal{A}}$ is singular). Given a vector $(\boldsymbol{c}; \boldsymbol{d}) \in \mathbb{R}^{n+m}$, one application of this preconditioner requires to compute the vector $(\boldsymbol{x}; \boldsymbol{y}) \in \mathbb{R}^{n+m}$ solving

$$-\hat{\boldsymbol{\mathcal{S}}}\boldsymbol{y} = \boldsymbol{d}, \tag{17}$$

$$\boldsymbol{\mathcal{A}}\boldsymbol{x} = \boldsymbol{b} = \boldsymbol{c} - \boldsymbol{\mathcal{B}}^T\boldsymbol{y}. \tag{18}$$

This approach is particularly attractive from the computational point of view, since matrix $\boldsymbol{\mathcal{A}}$ in eq. (28) is block diagonal and each block $\boldsymbol{A}^k$ is a weighted Laplacian. Thus, we can efficiently get the solution $\boldsymbol{x}$ by solving separately $K+1$ linear systems in the form $\boldsymbol{A}^k\boldsymbol{x}^k = \boldsymbol{b}^k$ using multigrid solvers. However, two issues arise.

K+1	16	32	64	128	16	32	64	128	16	32	64	128
μ	**Outer/Lin.sys.**				**CPU/Lin.sys.**				**Inner/Outer**			
$\mathcal{T}^0(N_\mathcal{T}=224, N_\mathcal{T}'=672)$					#DOF=(1.4e4, 2.8e4, 5.7e4, 1.1e5)							
1	108	110	106	107	3.8e-1	7.3e-1	1.6e0	4.1e0	1.0	1.0	1.8	2.1
2e-1	72	72	73	74	3.1e-1	6.3e-1	1.4e0	3.4e0	1.1	1.6	2.4	2.9
4e-2	65	65	66	66	3.1e-1	6.4e-1	1.5e0	3.5e0	1.7	2.7	3.5	4.2
8e-3	56	58	60	60	2.9e-1	6.4e-1	1.5e0	3.8e0	2.2	3.1	4.0	4.8
2e-3	52	54	48	52	2.9e-1	6.4e-1	1.4e0	3.7e0	2.3	3.1	4.2	5.2
3e-4	52	43	47	48	3.1e-1	5.8e-1	1.4e0	3.8e0	2.5	3.4	4.4	6.5
6e-5	42	41	41	39	3.0e-1	5.8e-1	1.3e0	3.1e0	2.9	4.4	5.6	7.3
1e-5	41	42	40	43	2.8e-1	6.3e-1	1.3e0	3.8e0	3.8	5.5	6.6	8.5
3e-6	43	55	50	68	3.2e-1	2.0e0	1.8e0	6.5e0	3.8	19.5	7.9	9.7
5e-7	54	72	67	74	3.8e-1	2.0e0	2.7e0	9.7e0	3.8	14.0	8.7	14.1
	60	63	61	65	3.2e-1	8.6e-1	1.6e0	4.4e0	2.1	5.0	4.3	5.7
$\mathcal{T}^1(N_\mathcal{T}=896, N_\mathcal{T}'=2688)$					#DOF=(5.6e4, 1.1e5, 2.3e5, 4.6e5)							
1	250	230	229	230	3.1e0	5.9e0	1.2e1	2.8e1	1.0	1.0	1.1	1.9
2e-1	136	144	147	148	1.9e0	4.3e0	9.8e0	2.3e1	1.0	1.2	1.6	2.4
4e-2	114	116	119	119	1.7e0	4.0e0	9.7e0	2.3e1	1.4	1.8	3.0	3.9
8e-3	90	107	107	106	1.5e0	4.0e0	9.9e0	2.4e1	1.8	2.2	3.8	4.7
2e-3	89	91	89	91	1.6e0	3.8e0	9.5e0	2.4e1	2.0	2.9	4.3	5.3
3e-4	102	85	84	79	1.9e0	3.8e0	9.7e0	2.3e1	2.0	3.0	4.5	5.4
6e-5	105	97	82	74	2.0e0	4.7e0	1.0e1	2.3e1	2.0	3.1	4.8	5.9
1e-5	100	86	79	78	2.1e0	4.2e0	1.1e1	2.6e1	2.2	3.2	6.1	7.1
3e-6	91	74	78	85	2.1e0	4.2e0	1.2e1	3.4e1	2.7	4.7	7.9	10.2
5e-7	88	74	96	115	2.1e0	4.7e0	1.6e1	5.2e1	3.1	5.3	8.3	12.3
	120	118	118	119	2.0e0	4.4e0	1.1e1	2.7e1	1.7	2.2	3.4	4.7
$\mathcal{T}^2(N_\mathcal{T}=3584, N_\mathcal{T}'=10752)$					#DOF=(2.3e5, 4.6e5, 9.1e5, 1.8e6)							
1	516	459	497	475	2.6e1	4.8e1	1.2e2	2.7e2	1.0	1.0	1.0	1.0
2e-1	300	298	319	317	1.7e1	3.5e1	8.4e1	2.1e2	1.0	1.0	1.1	1.6
4e-2	206	222	234	221	1.3e1	2.9e1	7.2e1	1.8e2	1.5	1.4	1.9	3.0
8e-3	190	164	167	169	1.3e1	2.4e1	5.8e1	1.5e2	1.8	1.8	2.4	4.0
2e-3	185	161	150	154	1.4e1	2.5e1	5.6e1	1.6e2	2.6	2.0	3.2	4.8
3e-4	223	170	168	139	1.8e1	2.8e1	6.9e1	1.6e2	3.0	2.5	3.5	5.2
6e-5	228	222	157	141	2.0e1	3.8e1	7.1e1	1.8e2	3.0	2.6	3.7	5.5
1e-5	255	183	156	145	2.3e1	3.4e1	7.4e1	1.9e2	3.0	2.7	3.9	5.7
3e-6	250	189	162	136	2.4e1	3.8e1	8.1e1	1.9e2	3.1	2.9	4.1	6.2
5e-7	252	160	140	140	2.5e1	3.3e1	7.4e1	2.2e2	3.1	3.0	4.2	6.9
	272	236	237	224	1.9e1	3.4e1	7.8e1	2.0e2	1.9	1.7	2.1	3.1
$\mathcal{T}^3(N_\mathcal{T}=14336, N_\mathcal{T}'=43008)$					#DOF=(9.0e5, 1.8e6, 3.7e6, 7.3e6)							
1	1003	919	959	1045	2.2e2	4.9e2	1.1e3	3.8e3	1.0	1.0	1.0	1.0
2e-1	1124	†	†	†	2.7e2	†	†	†	1.0	†	†	†
4e-2	†	†	540	503	†	†	6.9e2	2.0e3	†	†	1.4	1.8
8e-3	368	388	404	347	1.1e2	2.5e2	5.6e2	1.5e3	1.8	1.8	1.8	2.4
2e-3	397	313	279	318	1.6e2	2.5e2	4.2e2	1.5e3	2.8	2.8	2.1	3.5
3e-4	578	348	297	290	3.2e2	2.9e2	4.8e2	1.5e3	3.3	3.0	2.7	3.9
6e-5	419	483	327	319	3.1e2	4.3e2	5.7e2	1.7e3	3.9	3.3	3.0	4.0
1e-5	407	364	395	284	3.6e2	3.6e2	7.2e2	1.6e3	4.2	3.8	3.0	4.2
3e-6	389	301	318	251	3.8e2	3.1e2	6.0e2	1.5e3	4.4	3.9	3.0	4.6
5e-7	410	305	411	253	3.5e2	3.3e2	8.1e2	1.6e3	4.9	4.0	3.0	5.0
	†	†	†	†	†	†	†	†	†	†	†	†

Table 2: Numerical results using the SIMPLE preconditioner. Each sub-table reports the results for a given mesh, from the coarsest (top) to the finest (bottom), and for different time discretizations ($\Delta t = 1/(K+1)$). Each sub-table reports, while μ is reduced (leftmost column), the averaged outer iteration and CPU time per linear system (**Outer/Lin.sys.** and **CPU/Lin.sys.**) and the averaged inner iterations per outer iteration **Inner/Outer** for solving the linear system in eq. (16) (metrics defined in section 4). A final row summarizes the averages on the whole simulation. We highlighted in gray the time-space combination providing the uniform discretization. The † symbol denotes those IP steps where the linear solver failed.

The first one is due to the fact that each block $\boldsymbol{A}^k$ is singular, thus we need to ensure that

$$\boldsymbol{b}^k \in \mathrm{Im}(\boldsymbol{A}^k) = \mathrm{Ker}(\boldsymbol{A}^k)^\perp = \left\{\mathbf{1} \in \mathbb{R}^{N_\mathcal{T}}\right\}^\perp, \quad k = 1, \dots K+1\,, \tag{19}$$

for the linear systems to be admissible. This condition may not hold for the right-hand side $\boldsymbol{b} = \boldsymbol{c} - \boldsymbol{\mathcal{B}}^T\boldsymbol{y}$ in eq. (18) for any application of the preconditioner, because $\mathrm{Im}(\boldsymbol{\mathcal{B}}^T)$ is not orthogonal to $\mathrm{Ker}(\boldsymbol{\mathcal{A}})$ (see remark 2). Even if the orthogonality condition in eq. (19) is satisfied, or imposed by orthogonalization, there are multiple solutions, which differ by an additive constant, and thus we also need a criteria to select one. This is not trivial since the solution $\boldsymbol{\delta\phi}$ of eq. (12) must also solve the second block of equations, $\boldsymbol{\mathcal{B}\delta\phi} - \boldsymbol{\mathcal{C}\delta\rho} = \tilde{\boldsymbol{g}}$. In the next section, we describe how to deal with this compatibility issue with a reformulation of the continuous nonlinear system in eq. (2).

A second, harder, issue is the design of a good approximation of the inverse of the matrix $\boldsymbol{\mathcal{S}} = -(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^{-1}\boldsymbol{\mathcal{B}}^T)$, used in the solution of the linear system eq. (17). First, $\boldsymbol{\mathcal{S}}$ is not well defined since $\boldsymbol{\mathcal{A}}$ is singular and $\mathrm{Im}(\boldsymbol{\mathcal{B}}^T)$ is not orthogonal to $\mathrm{Ker}(\boldsymbol{\mathcal{A}})$. Moreover, even overcoming this problem, the matrix $\boldsymbol{\mathcal{S}}$ is block tridiagonal but with dense blocks, due to the presence of the pseudo-inverse of the Laplacian matrices $\boldsymbol{A}^k$, thus too costly to form and invert. This second issue is addressed in section 4.4.3.

4.4.1 Splitting formulation

In order to cope with the first issue, we split the solution ϕ of eq. (2) into two components, a new potential $\bar{\phi} : \Omega \times [0,1] \to \mathbb{R}$ satisfying the constraint $\int_\Omega \bar{\phi}(t)\,dx = 0$ for all $t \in [0,1]$ and a correction term:

$$\phi(t,x) = \bar{\phi}(t,x) + \int_0^t \lambda(s)ds \tag{20}$$

where $\lambda : [0,1] \to \mathbb{R}$ depends only on the time variable. In the new unknowns $(\bar{\phi}, \rho, s, \lambda)$ the system of PDEs (2) becomes

$$\begin{aligned}
&-\partial_t\rho - \mathrm{div}_x\left(\rho\,\nabla_x\,\bar{\phi}\right) = 0,\\
&\partial_t\bar{\phi} + \frac{|\,\nabla_x\,\bar{\phi}|^2}{2} + s + \lambda = 0,\\
&\rho \geq 0,\ s \geq 0,\ \rho s = 0,\\
&\int_\Omega \bar{\phi}\,dx = 0.
\end{aligned} \tag{21}$$

Following the same steps described in section 2, the (relaxed) discrete counterpart of equation in eq. (21) is a nonlinear system of equations with unknowns $(\bar{\boldsymbol{\phi}}, \boldsymbol{\rho}, \boldsymbol{s}, \boldsymbol{\lambda}) \in (\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^K)$ given by

$$\begin{aligned}
F_{\boldsymbol{\phi}}(\bar{\boldsymbol{\phi}}, \boldsymbol{\rho}) &= \left(F^1_{\boldsymbol{\phi}}; \dots; F^{K+1}_{\boldsymbol{\phi}}\right) &&= \mathbf{0} \in \mathbb{R}^n\,,\\
\bar{F}_{\boldsymbol{\rho}}(\bar{\boldsymbol{\phi}}, \boldsymbol{\rho}, \boldsymbol{s}, \boldsymbol{\lambda}) &= \left(\bar{F}^1_{\boldsymbol{\rho}}; \dots; \bar{F}^K_{\boldsymbol{\rho}}\right) &&= \mathbf{0} \in \mathbb{R}^m\,,\\
F_{\boldsymbol{s}}(\boldsymbol{\rho}, \boldsymbol{s}) &= \left(F^1_{\boldsymbol{s}}; \dots; F^K_{\boldsymbol{s}}\right) - \mu\mathbf{1} &&= \mathbf{0} \in \mathbb{R}^m\,,\\
F_{\boldsymbol{\lambda}}(\bar{\boldsymbol{\phi}}) &= \left(F^1_{\boldsymbol{\lambda}}; \dots; F^K_{\boldsymbol{\lambda}}\right) &&= \mathbf{0} \in \mathbb{R}^{K+1},
\end{aligned}$$

where $F^k_{\boldsymbol{\phi}}$ and $F^k_{\boldsymbol{s}}$ are defined in eqs. (5) and (7), while $\bar{F}^k_{\boldsymbol{\rho}}$ and $F^k_{\boldsymbol{\lambda}}$ are given by

$$\begin{aligned}
\bar{F}^k_{\boldsymbol{\rho}}(\bar{\boldsymbol{\phi}}, \boldsymbol{\rho}, \boldsymbol{s}, \boldsymbol{\lambda}) &:= F^k_{\boldsymbol{\rho}}(\boldsymbol{\rho}, \bar{\boldsymbol{\phi}}, \boldsymbol{s}) + |\boldsymbol{c'}|\boldsymbol{\lambda}^k\,, &&k = 1, \dots, K,\\
\bar{F}^k_{\boldsymbol{\lambda}}(\bar{\boldsymbol{\phi}}^k) &:= |\boldsymbol{c}|^T\bar{\boldsymbol{\phi}}^k\,, &&k = 1, \dots, K+1,
\end{aligned}$$

with $F^k_{\boldsymbol{\rho}}$ as given in eq. (6). Note that there are $n + 2m + K + 1$ equations and $n + 2m + K$ unknowns. The additional equation fixes the global constant for the potential $\bar{\boldsymbol{\phi}}$. The linear system arising from the Newton method becomes

$$\begin{pmatrix}
\boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T & & & \\
\boldsymbol{\mathcal{B}} & & \boldsymbol{\mathcal{M}'} & \boldsymbol{\mathcal{M}'\mathcal{E}'}^T & \\
 & \mathrm{Diag}\,(\boldsymbol{s}) & \mathrm{Diag}\,(\boldsymbol{\rho}) & & \\
\boldsymbol{\mathcal{E}\mathcal{M}} & & & &
\end{pmatrix}
\begin{pmatrix}
\boldsymbol{\delta\bar{\phi}}\\ \boldsymbol{\delta\rho}\\ \boldsymbol{\delta s}\\ \boldsymbol{\delta\lambda}
\end{pmatrix}
=
\begin{pmatrix}
\boldsymbol{f}\\ \bar{\boldsymbol{g}}\\ \boldsymbol{h}\\ \boldsymbol{i}
\end{pmatrix}
= -
\begin{pmatrix}
\boldsymbol{F}_{\boldsymbol{\phi}}\\ \bar{\boldsymbol{F}}_{\boldsymbol{\rho}}\\ \boldsymbol{F}_{\boldsymbol{s}} - \mu\mathbf{1}\\ \boldsymbol{F}_{\boldsymbol{\lambda}}
\end{pmatrix}, \tag{22}$$

where matrices $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{K+1,(K+1)N_\mathcal{T}}$ and $\boldsymbol{\mathcal{E}'} \in \mathbb{R}^{K,KN_{\mathcal{T}'}}$ are given by

$$\boldsymbol{\mathcal{E}} = \begin{pmatrix}
\mathbf{1}^T_{N_\mathcal{T}} & & \\
 & \ddots & \\
 & & \mathbf{1}^T_{N_\mathcal{T}}
\end{pmatrix}, \quad
\boldsymbol{\mathcal{E}'} = \begin{pmatrix}
\mathbf{1}^T_{N_{\mathcal{T}'}} & & \\
 & \ddots & \\
 & & \mathbf{1}^T_{N_{\mathcal{T}'}}
\end{pmatrix}.$$

Once the vectors $\boldsymbol{\delta\bar{\phi}}$ and $\boldsymbol{\delta\lambda}$ in eq. (22) are found, $\boldsymbol{\delta\phi}$ can be retrieved in the original system eq. (12) using the formula

$$\boldsymbol{\delta\phi}^k = \boldsymbol{\delta\bar{\phi}}^k + \sum_{i=1}^{k} \Delta t^i \boldsymbol{\delta\lambda}^i,$$

which is the finite-dimensional counterpart of eq. (20).

4.4.2 New saddle point linear system

After eliminating the unknown $\boldsymbol{\delta s}$ as in section 3.1, the linear system in eq. (22) becomes

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T & \\ \boldsymbol{\mathcal{B}} & -\boldsymbol{\mathcal{C}} & \boldsymbol{\mathcal{M}'\mathcal{E}'}^T \\ \boldsymbol{\mathcal{E}\mathcal{M}} & & \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta\bar{\phi}} \\ \boldsymbol{\delta\rho} \\ \boldsymbol{\delta\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\tilde{g}} \\ \boldsymbol{i} \end{pmatrix}. \tag{23}$$

The variable $\boldsymbol{\delta\lambda}$ can be eliminated as well by multiplying by $\boldsymbol{\mathcal{E}'}$ the second block of equations in eq. (22), yielding

$$\boldsymbol{\mathcal{E}'}(\boldsymbol{\mathcal{B}\delta\bar{\phi}} - \boldsymbol{\mathcal{C}\delta\rho}) + \underbrace{\boldsymbol{\mathcal{E}'\mathcal{M}'\mathcal{E}'}^T}_{|\Omega|\boldsymbol{I}_K} \boldsymbol{\delta\lambda} = \boldsymbol{\mathcal{E}'\tilde{g}}\,.$$

Plugging this expression in eq. (23), the linear system in the unknowns $(\boldsymbol{\delta\bar{\phi}}; \boldsymbol{\delta\rho})$ becomes

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T \\ \boldsymbol{\mathcal{P}\mathcal{B}} & -\boldsymbol{\mathcal{P}\mathcal{C}} \\ \boldsymbol{\mathcal{E}\mathcal{M}} & \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta\bar{\phi}} \\ \boldsymbol{\delta\rho} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\mathcal{P}\tilde{g}} \\ \boldsymbol{i} \end{pmatrix}. \tag{24}$$

where the matrix $\boldsymbol{\mathcal{P}}$, which is given by

$$\boldsymbol{\mathcal{P}} := \boldsymbol{I} - \frac{1}{|\Omega|} \boldsymbol{\mathcal{M}'\mathcal{E}'}^T \boldsymbol{\mathcal{E}'},$$

is a projector (indeed $\boldsymbol{\mathcal{P}}^2 = \boldsymbol{\mathcal{P}}$). Moreover, its transpose $\boldsymbol{\mathcal{P}}^T$ is the discretization of the zero mean-projector, which maps a function $g : [0,1] \times \Omega \to \mathbb{R}$ to $g - \int_\Omega g/|\Omega|$.

The linear system in (24) is not over-determined since $\mathrm{Ker}(\boldsymbol{\mathcal{A}}) \subset \mathrm{Ker}(\boldsymbol{\mathcal{P}\mathcal{B}})$ and the rows of $\boldsymbol{\mathcal{E}\mathcal{M}}$ are $K+1$ vectors linearly independent from the rows of $\boldsymbol{\mathcal{A}}$ or $\boldsymbol{\mathcal{P}\mathcal{B}}$. Recalling remark 3, $\boldsymbol{\delta\rho} \in \mathrm{Ker}(\boldsymbol{\mathcal{P}})$ or equivalently $\boldsymbol{\delta\rho} \in \mathrm{Im}(\boldsymbol{\mathcal{P}}^T)$ and, since $\boldsymbol{\mathcal{P}}^T$ is also a projector, we can write $\boldsymbol{\delta\rho} = \boldsymbol{\mathcal{P}}^T\boldsymbol{\delta\rho}$. Thus, we can rewrite system (24) as

$$\begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T \\ \boldsymbol{\mathcal{P}\mathcal{B}} & -\boldsymbol{\mathcal{P}\mathcal{C}\mathcal{P}}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta\hat{\phi}} \\ \boldsymbol{\delta\rho} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{\mathcal{P}\tilde{g}} \end{pmatrix}, \tag{25}$$

where we neglect the constraints $\boldsymbol{\mathcal{E}\mathcal{M}\delta\bar{\phi}} = \boldsymbol{i}$. For any solution $(\boldsymbol{\delta\hat{\phi}}; \boldsymbol{\delta\rho})$, the solution $(\boldsymbol{\delta\bar{\phi}}; \boldsymbol{\delta\rho})$ of eq. (24) can be recovered by simply correcting the mean of $\boldsymbol{\delta\hat{\phi}}$.

The linear system in eq. (25) does not suffer from the first issue mentioned above associated to the singularity of matrix $\boldsymbol{\mathcal{A}}$. In fact, note that $\mathrm{Im}(\boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T) \perp \mathrm{Ker}(\boldsymbol{\mathcal{A}})$, since $\mathrm{Ker}(\boldsymbol{\mathcal{A}}) \subset \mathrm{Ker}(\boldsymbol{\mathcal{P}\mathcal{B}})$. This ensures that the dual Schur complement $\boldsymbol{\mathcal{S}} = -\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}\mathcal{A}}^+\boldsymbol{\mathcal{B}}^T)\boldsymbol{\mathcal{P}}^T$ (where $\boldsymbol{\mathcal{A}}^+$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{\mathcal{A}}$) is well defined. Moreover the following (ideal) block triangular preconditioner

$$\boldsymbol{P}_t = \begin{pmatrix} \boldsymbol{\mathcal{A}} & \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T \\ & -\boldsymbol{\mathcal{S}} \end{pmatrix} \tag{26}$$

is well defined as well. In fact, given a vector $(\boldsymbol{c}; \boldsymbol{d})$ with $\boldsymbol{c} \in \mathrm{Ker}(\boldsymbol{\mathcal{A}})^\perp$ and $\boldsymbol{d} \in \mathrm{Im}(\boldsymbol{\mathcal{P}})$, the application of this preconditioner requires to compute the vector $(\boldsymbol{x}; \boldsymbol{y})$ solving the following two linear systems

$$-\boldsymbol{\mathcal{S}}\boldsymbol{y} = \boldsymbol{d}\,, \tag{27}$$

$$\boldsymbol{\mathcal{A}}\boldsymbol{x} = \boldsymbol{b} = \boldsymbol{c} - \boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T\boldsymbol{y}\,. \tag{28}$$

The linear system in eq. (28) is now well defined, since $\boldsymbol{b} \in \mathrm{Ker}(\boldsymbol{\mathcal{A}})^\perp$ at any preconditioner application. In fact, both vectors $\boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T\boldsymbol{y}$ and $\boldsymbol{c}$ belong to $\mathrm{Ker}(\boldsymbol{\mathcal{A}})^\perp$, the first vector because $\mathrm{Im}(\boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T) \subset \mathrm{Ker}(\boldsymbol{\mathcal{A}})^\perp$, the second vector since it is a linear combination of the vector $\boldsymbol{f}$ in eq. (25) and vectors in $\mathrm{Im}(\boldsymbol{\mathcal{A}}) \cup \mathrm{Im}(\boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{P}}^T)$, all belonging to $\mathrm{Ker}(\boldsymbol{\mathcal{A}})^\perp$.

4.4.3 Approximating the inverse of the dual Schur complement

We turn now to the issue of approximating the Schur complement. Our first attempt was to approximate it with $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}(\mathrm{Diag}\,(\boldsymbol{\mathcal{A}}))^{-1}\boldsymbol{\mathcal{B}}^T)\boldsymbol{\mathcal{P}}^T$, but it led to poor performance. We tried to apply the Algebraically stabilized Least Square Commutator proposed in [21, Section 4.2], but this leads to poor results, for different combinations of the relaxation parameters that appear in such approach. We attribute this phenomenon to the fact that the matrix $\boldsymbol{\mathcal{C}}$ is not a stabilization operator for the saddle point system, as assumed in [21].

In order to devise a better approximation of the Schur complement, here we follow the idea in [20, 21, 19] of looking at the infinite-dimensional differential operators associated to the matrices that compose it and try to deduce a possible approximation of its inverse. The following proposition shows a differential identity that goes in this direction.

Proposition 1. *Let $\rho : [0,1] \times \Omega \to \mathbb{R}_{>0}$, $\phi : [0,1] \times \Omega \to \mathbb{R}$, smooth enough. Denoting by $-\Delta_\rho = -\operatorname{div}(\rho \nabla)$, the following operator identity holds:*

$$\begin{aligned}(\partial_t + \operatorname{div}(\cdot\, \nabla \phi))\,(-\Delta_\rho) = -\Delta_\rho\,(\partial_t + \nabla\phi \cdot \nabla) - \operatorname{div}\left((\partial_t \rho + \operatorname{div}(\rho \nabla \phi))\,\nabla\right)\\ +2\operatorname{div}(\rho\, \nabla^2 \phi\, \nabla)\,.\end{aligned} \tag{29}$$

Proof. Consider any function $g_0 : [0,1] \times \Omega \to \mathbb{R}$, such that $\int_\Omega g_0(t,\cdot) = 0$ for all $t \in [0,1]$ and let us define $u : [0,1] \times \Omega \to \mathbb{R}$ given by

$$-\operatorname{div}(\rho \nabla u) = g_0 \tag{30}$$

with zero Neumann boundary conditions. Multiplying both sides of equation eq. (30) by $\partial_i \phi$ (where $\partial_i = \partial_{x_i}$) and taking the divergence, we obtain

$$-\sum_{i,j} \partial_i\left(\partial_i \phi \partial_j(\rho \partial_j u)\right) = \operatorname{div}(g_0 \nabla \phi).$$

Using the identity $\partial_i\phi\partial_j(\rho\partial_j u) = \partial_j(\partial_i\phi\rho\partial_j u) - \partial_j(\partial_i\phi)\rho\partial_j u$, we get

$$\begin{aligned}\operatorname{div}(g_0 \nabla \phi) &= -\sum_{i,j} \partial_i\left(\partial_j(\partial_i\phi\rho\partial_j u) - \partial_{j,i}\phi\rho\partial_j u\right)\\ &= -\sum_{j,i} \partial_j\left(\partial_{i,j} u \rho \partial_i \phi\right) - \sum_{j,i} \partial_j\left(\partial_i(\rho\partial_i\phi)\partial_j u\right) + \operatorname{div}(\rho\,\nabla^2\phi\,\nabla u)\\ &= -\sum_j \partial_j\left(\sum_i \partial_{i,j} u \rho \partial_i \phi\right) - \sum_j \partial_j\left(\sum_i \partial_i(\rho\partial_i\phi)\partial_j u\right)\\ &\qquad + \operatorname{div}(\rho\,\nabla^2\phi\,\nabla u)\\ &= -\operatorname{div}(\rho\,\nabla^2 u\,\nabla\phi) - \operatorname{div}(\operatorname{div}(\rho\nabla\phi)\,\nabla u) + \operatorname{div}(\rho\,\nabla^2\phi\,\nabla u)\,.\end{aligned}$$

Using the identity $\nabla(\nabla\phi\cdot\nabla u) = \nabla^2\phi\,\nabla u + \nabla^2 u\,\nabla\phi$, we obtain

$$\operatorname{div}(g_0\nabla\phi) = -\operatorname{div}(\rho\nabla(\nabla\phi\cdot\nabla u)) - \operatorname{div}(\operatorname{div}(\rho\nabla\phi)\nabla u) + 2\operatorname{div}(\rho\nabla^2\phi\nabla u)\,. \tag{31}$$

Now, differentiating with respect to time the expression in eq. (30), we obtain

$$-\operatorname{div}\left(\partial_t\rho\nabla u + \rho\nabla\partial_t u\right) = \partial_t g_0\,. \tag{32}$$

Combining eq. (31), eq. (32) we obtain

$$\begin{aligned}\partial_t g_0 + \operatorname{div}(g_0\nabla\phi) = -\operatorname{div}(\rho\nabla(\partial_t u + \nabla\phi\cdot\nabla u)) - \operatorname{div}\left((\partial_t\rho + \operatorname{div}(\rho\nabla\phi))\nabla u\right)\\ +2\operatorname{div}(\rho\nabla^2\phi\nabla u)\,.\end{aligned}$$

Since $g_0 = -\Delta_\rho u$ by eq. (30), we proved eq. (29). □

Our approximation of the Schur complement is based on neglecting the terms $-\operatorname{div}\left((\partial_t\rho + \operatorname{div}(\rho\nabla\phi))\nabla\right.$ and $2\operatorname{div}(\rho\nabla^2\phi\nabla)$ on the right-hand side of eq. (29). The first term may be assumed to be small since it contains the continuity equation, which is the first nonlinear equation in the system eq. (2a). This term is small at the initial and final Newton steps within each IP iteration, and experimental observations showed it remains small during the intermediate Newton iterations. The second term can be neglected under certain assumptions. If the optimal transport is a translation this term is null. When the transported measures ρ^{in} and ρ^{f} are Gaussian [48], or more generally log-concave probability densities [15, 17], the largest eigenvalue of $\nabla^2\phi(0,x)$ is bounded. However, note that in general

this term may be non null, as for example in the test case 3. Supposing that both terms can be neglected, we get the following approximate identity:

$$(\partial_t + \mathrm{div}(\cdot\,\nabla\,\phi))\,(-\Delta_\rho) \approx -\Delta_\rho\,(\partial_t + \nabla\,\phi\cdot\nabla)\,.$$

In our finite-dimensional problem, this expression translates into the approximate matrix identity

$$-\boldsymbol{\mathcal{B}}^T\boldsymbol{\mathcal{M}'}^{-1}\tilde{\boldsymbol{\mathcal{A}}} \approx \boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}}\,, \tag{33}$$

where $\tilde{\boldsymbol{\mathcal{A}}} \in \mathbb{R}^{m,m}$ and $\tilde{\boldsymbol{\mathcal{B}}} \in \mathbb{R}^{n,m}$ discretize the operators $-\Delta_\rho$ and $\partial_t + \nabla\,\phi\cdot\nabla$, respectively, similarly to the matrices $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ but with different dimensions. The scaling factors $\boldsymbol{\mathcal{M}'}^{-1}$ and $\boldsymbol{\mathcal{M}}$ in eq. (33) are introduced to match the scaling dimensions of the matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, and $\boldsymbol{\mathcal{B}}^T$. The most natural definition of $\tilde{\boldsymbol{\mathcal{A}}}$ and $\tilde{\boldsymbol{\mathcal{B}}}$ is the following:

$$\tilde{\boldsymbol{\mathcal{A}}} = \text{Block Diag}\left(\left(\tilde{\boldsymbol{A}}^k\right)_{k=1}^K\right), \quad \tilde{\boldsymbol{A}}^k := \mathbf{div}_{\boldsymbol{\mathcal{T}'},\boldsymbol{\mathcal{E}'}}\text{Diag}\left(\boldsymbol{R}_{\mathcal{E}'}\left[\boldsymbol{\rho}^k\right]\right)\boldsymbol{\nabla}_{\boldsymbol{\mathcal{E}'},\boldsymbol{\mathcal{T}'}},$$
$$\tilde{\boldsymbol{\mathcal{B}}} := -\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{J}}\tilde{\boldsymbol{\mathcal{D}}_t}^T + \boldsymbol{\mathcal{G}}\tilde{\boldsymbol{\mathcal{D}}}_{\boldsymbol{x}}\boldsymbol{\mathcal{J}}^T\boldsymbol{\mathcal{H}}^T,$$

where the matrices $\boldsymbol{\mathcal{J}}$, $\boldsymbol{\mathcal{H}}$, and $\boldsymbol{\mathcal{G}}$ are defined in eqs. (10) and (11), while the matrices $\tilde{\boldsymbol{\mathcal{D}}_t} \in \mathbb{R}^{KN_{\mathcal{T}'},(K+1)N_{\mathcal{T}'}}$ and $\tilde{\boldsymbol{\mathcal{D}}_{\boldsymbol{x}}} \in \mathbb{R}^{KN_{\mathcal{E}'},KN_{\mathcal{T}'}}$ are

$$\tilde{\boldsymbol{\mathcal{D}}_t} = \frac{1}{\Delta t}\begin{pmatrix} -\boldsymbol{I}_{N_{\mathcal{T}'}} & \boldsymbol{I}_{N_{\mathcal{T}'}} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{I}_{N_{\mathcal{T}'}} & \boldsymbol{I}_{N_{\mathcal{T}'}} \end{pmatrix}, \quad \tilde{\boldsymbol{\mathcal{D}}_{\boldsymbol{x}}} = \text{Block Diag}\left((\boldsymbol{\nabla}_{\mathcal{E}',\mathcal{T}'})_{k=1}^K\right).$$

Using the approximate identity in eq. (33), the linear system in eq. (27) becomes

$$-\boldsymbol{d} = \boldsymbol{\mathcal{S}}\boldsymbol{y} = \boldsymbol{\mathcal{P}}\left(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^+\boldsymbol{\mathcal{B}}^T\right)\boldsymbol{\mathcal{P}}^T\boldsymbol{y} \approx \boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{C}} - \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}}\tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'})\boldsymbol{\mathcal{P}}^T\boldsymbol{y}\,. \tag{34}$$

Since $\mathrm{Ker}(\boldsymbol{\mathcal{P}}\boldsymbol{\mathcal{M}'}) = \mathrm{Im}(\boldsymbol{\mathcal{E}}^T)$ are discrete functions constant in space, we have that $\mathrm{Im}(\boldsymbol{\mathcal{M}'}\boldsymbol{\mathcal{P}}^T) \subset \mathrm{Im}(\tilde{\boldsymbol{\mathcal{A}}})$ and the last expression is equal to

$$-\boldsymbol{d} = \boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}'}^{-1}\tilde{\boldsymbol{\mathcal{A}}}\tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'} - \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}}\tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'})\boldsymbol{\mathcal{P}}^T\boldsymbol{y}\,.$$

Hence, we only have to solve the linear system

$$-\boldsymbol{d} = \boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}'}^{-1}\tilde{\boldsymbol{\mathcal{A}}} - \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}})\tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'}\boldsymbol{\mathcal{P}}^T\boldsymbol{y}\,,$$

which is compatible since $\boldsymbol{d} \in \mathrm{Im}(\boldsymbol{\mathcal{P}})$. Note that the dense matrix $(\boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{A}}^+\boldsymbol{\mathcal{B}}^T)$ in eq. (34) is replaced by the operator $(\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}'}^{-1}\tilde{\boldsymbol{\mathcal{A}}} - \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}})\tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'}$ which is dense by the presence of the pseudo-inverse of $\tilde{\boldsymbol{\mathcal{A}}}$. But now this operator is composed by two factors. This means that we can approximate the action of their respective inverse by solving a sparse linear system followed by a matrix-vector product. In fact, since the solution $\boldsymbol{\delta\rho}$ in eq. (25) belongs to the image of $\boldsymbol{\mathcal{P}}^T$, we may look for a solution $\boldsymbol{y} \in \mathrm{Im}(\boldsymbol{\mathcal{P}}^T)$. By setting

$$\boldsymbol{z} = \tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'}\boldsymbol{\mathcal{P}}^T\boldsymbol{y} = \tilde{\boldsymbol{\mathcal{A}}}^+\boldsymbol{\mathcal{M}'}\boldsymbol{y}\,,$$

we can compute $\boldsymbol{y}$ by first solving the linear system

$$-\boldsymbol{d} = (\boldsymbol{\mathcal{C}}\boldsymbol{\mathcal{M}'}^{-1}\tilde{\boldsymbol{\mathcal{A}}} - \boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}})\boldsymbol{z} \tag{35}$$

and then computing $\boldsymbol{y} = \boldsymbol{\mathcal{M}'}^{-1}\tilde{\boldsymbol{\mathcal{A}}}\boldsymbol{z}$.

Moreover, in order to cope with the ill-conditioning derived from the presence of the matrix $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{M}'}\mathrm{Diag}\,(\boldsymbol{s}/\boldsymbol{\rho})$, we scale eq. (35) by $\mathrm{Diag}\,(\boldsymbol{\rho})$ and we solve the linear system

$$(\mathrm{Diag}\,(\boldsymbol{s})\,\tilde{\boldsymbol{\mathcal{A}}} - \mathrm{Diag}\,(\boldsymbol{\rho})\,\boldsymbol{\mathcal{B}}\boldsymbol{\mathcal{M}}^{-1}\tilde{\boldsymbol{\mathcal{B}}})\boldsymbol{z} = -\mathrm{Diag}\,(\boldsymbol{\rho})\,\boldsymbol{d}, \tag{36}$$

similarly to what done in eq. (16) for the SIMPLE preconditioner.

We refer to the resulting preconditioner as "$\boldsymbol{BB}$-preconditioner", due to presence of this composed operator in the approximate factorization of the Schur complement. This choice was done in analogy with the "$\boldsymbol{BFB}t$-preconditioner" introduced in [19], whose ideas inspired the preconditioner described in this section.

$K+1$	16	32	64	128	16	32	64	128	16	32	64	128
μ	**Outer/Lin.sys.**				**CPU/Lin.sys.**				**Inner/Outer**			
$\mathcal{T}^0(N_\mathcal{T}=224, N_\mathcal{T}'=672)$ #DOF=(1.4e4, 2.8e4, 5.7e4, 1.1e5)												
1	7	9	10	12	1.2e0	2.5e0	5.4e0	1.4e1	2.1	2.3	2.6	3.2
2e-1	8	9	9	10	1.3e0	2.6e0	5.5e0	1.3e1	2.6	3.2	3.9	4.8
4e-2	12	13	14	15	1.3e0	2.8e0	5.7e0	1.4e1	2.8	3.6	4.5	5.5
8e-3	17	18	21	25	1.4e0	3.0e0	6.4e0	1.8e1	3.1	4.1	5.0	6.3
2e-3	20	25	33	44	1.5e0	3.3e0	7.5e0	2.3e1	3.6	4.4	5.1	6.2
3e-4	30	42	63	88	1.6e0	3.9e0	1.0e1	3.4e1	3.7	4.5	4.7	5.4
6e-5	46	72	117	180	1.9e0	4.9e0	1.5e1	5.9e1	3.9	4.1	4.4	4.4
1e-5	76	127	248	†	2.4e0	6.8e0	2.8e1	†	4.1	4.3	4.0	†
3e-6	136	324	†	†	3.5e0	1.5e1	†	†	4.1	4.8	†	†
5e-7	297	†	†	†	6.2e0	†	†	†	4.3	†	†	†
	58	†	†	†	2.1e0	†	†	†	4.0	†	†	†
$\mathcal{T}^1(N_\mathcal{T}=896, N_\mathcal{T}'=2688)$ #DOF=(5.6e4, 1.1e5, 2.3e5, 4.6e5)												
1	7	8	9	10	1.7e0	3.4e0	8.1e0	2.1e1	2.9	2.3	2.5	3.2
2e-1	8	8	9	10	1.6e0	3.4e0	8.0e0	2.1e1	2.4	2.8	3.6	4.3
4e-2	13	15	15	15	1.8e0	4.0e0	1.0e1	2.6e1	2.3	3.3	4.2	5.4
8e-3	17	20	21	21	1.9e0	4.7e0	1.2e1	3.3e1	2.6	3.6	4.8	5.8
2e-3	22	26	27	30	2.2e0	5.5e0	1.5e1	4.4e1	2.9	4.1	5.4	5.8
3e-4	29	34	38	44	2.6e0	6.5e0	2.0e1	5.9e1	3.2	4.4	5.7	6.0
6e-5	37	46	61	86	3.1e0	8.7e0	2.8e1	1.1e2	3.7	4.7	5.8	6.3
1e-5	52	65	88	157	4.0e0	1.3e1	4.3e1	2.0e2	4.2	5.4	8.1	7.7
3e-6	72	100	188	395	5.0e0	1.7e1	1.0e2	4.9e2	4.5	5.8	11.0	8.1
5e-7	100	154	†	†	6.7e0	2.5e1	†	†	4.9	6.0	†	†
	32	43	†	†	2.9e0	8.4e0	†	†	4.0	5.1	†	†
$\mathcal{T}^2(N_\mathcal{T}=3584, N_\mathcal{T}'=10752)$ #DOF=(2.3e5, 4.6e5, 9.1e5, 1.8e6)												
1	6	7	8	10	2.7e0	6.6e0	1.7e1	5.6e1	3.5	2.9	2.4	2.6
2e-1	9	9	10	10	3.0e0	7.3e0	1.9e1	5.9e1	2.6	2.5	2.9	3.6
4e-2	14	16	18	17	3.9e0	1.0e1	2.9e1	8.6e1	2.9	2.6	3.5	4.4
8e-3	19	22	25	25	4.9e0	1.3e1	4.0e1	1.2e2	3.3	2.7	3.9	4.9
2e-3	25	28	32	30	7.0e0	1.6e1	5.0e1	1.5e2	3.8	2.9	4.3	5.7
3e-4	32	34	42	41	8.7e0	2.0e1	6.5e1	2.0e2	4.4	3.2	4.3	5.8
6e-5	44	43	53	58	1.2e1	2.5e1	8.0e1	2.8e2	4.8	3.4	4.4	6.2
1e-5	62	61	84	103	1.6e1	3.6e1	1.3e2	4.9e2	5.4	3.9	4.7	6.4
3e-6	92	94	130	186	2.5e1	5.8e1	2.0e2	9.6e2	6.7	4.3	5.5	8.8
5e-7	125	143	211	†	3.6e1	9.5e1	3.5e2	†	7.3	5.2	6.5	†
	35	41	54	†	9.7e0	2.6e1	8.7e1	†	5.5	4.0	5.2	†
$\mathcal{T}^3(N_\mathcal{T}=14336, N_\mathcal{T}'=43008)$ #DOF=(9.0e5, 1.8e6, 3.7e6, 7.3e6)												
1	7	7	8	9	8.3e0	2.1e1	5.7e1	2.2e2	4.3	3.6	2.9	2.4
2e-1	9	9	10	11	9.8e0	2.5e1	7.0e1	2.6e2	3.1	2.8	2.7	2.8
4e-2	16	17	18	20	1.5e1	4.0e1	1.1e2	4.4e2	3.3	3.2	2.9	3.7
8e-3	25	29	30	32	2.7e1	6.3e1	1.7e2	7.3e2	4.2	3.4	2.7	3.8
2e-3	34	36	39	48	5.0e1	8.1e1	2.3e2	1.1e3	4.8	3.7	2.9	4.2
3e-4	47	43	52	58	7.6e1	1.0e2	3.0e2	1.3e3	4.9	4.0	2.9	4.3
6e-5	70	54	59	73	1.2e2	1.4e2	3.4e2	1.6e3	5.1	4.7	3.2	4.5
1e-5	110	68	72	100	1.9e2	2.0e2	4.3e2	2.3e3	5.6	5.6	3.7	4.6
3e-6	146	105	107	149	2.8e2	3.2e2	6.5e2	3.4e3	6.4	6.4	4.1	4.9
5e-7	190	162	161	237	3.9e2	6.1e2	1.0e3	5.6e3	7.3	8.5	5.1	5.5
	55	44	52	68	9.4e1	1.3e2	3.2e2	1.6e3	5.8	5.7	3.9	4.8

Table 3: Numerical results using the $\boldsymbol{BB}$-preconditioner. Each sub-table reports the results for a given mesh, from the coarsest (top) to the finest (bottom), and for different time discretizations ($\Delta t = 1/(K+1)$). Each sub-table reports, while μ is reduced (leftmost column), the averaged outer iteration and CPU time per linear system (**Outer/Lin.sys.** and **CPU/Lin.sys.**) and the averaged inner iterations per outer iteration **Inner/Outer** for solving the linear system in eq. (14) (metrics defined in eq. (35)). A final row summarizes the averages on the whole simulation. We highlighted in gray the time-space combination providing the uniform discretization. The † symbol denotes those IP steps where the linear solver failed.

4.4.4 Numerical results

In this section we present the results obtained on test case 2 using the preconditioner described in section 4.4.3. Its application requires the approximate solution of the linear systems in eqs. (28) and (36). The first is solved using the AGMG solver with inner tolerance $\varepsilon_{\text{in}}^{\boldsymbol{S}} = 1e-1$ (determined experimentally). The second involves the solution of $K+1$ weighted Laplacian systems. We solved them with the AGMG solver with tolerance $\varepsilon_{\text{in}}^{\boldsymbol{A}} = 5e-2$. This value, slightly smaller than those adopted in previous cases, improves the robustness of the preconditioner. The number of multigrid iterations for these systems ranges between 2 and 3 in all cases.

In table 3 we summarized the results for the metrics defined in section 4 that we used to measure the preconditioner performance. The number of averaged inner iterations **Inner/Outer** refers to the solution of the linear system eq. (36). This number remains bounded between 2.1 and 8.5 iterations per preconditioner applications, it tends to increase slightly with $\mu \to 0$ but it is not affected by the size of the problem.

Unfortunately, the number of outer iterations **Outer/Lin.sys.** tends to increase as $\mu \to 0$, exceeding in some cases the 400 limit we fixed for the FGMRES iterations. This phenomenon is more pronounced when the temporal scale is finer than the spatial one to. This loss in efficiency for $\mu \to 0$ is more evident in the test cases 2 and 3 whose solutions have compact support, while for the test case 1 it is less pronounced. This suggests that this phenomenon can be related to the degeneracy of the linear system eq. (36), which becomes underdetermined due to the presence of terms that go to zero on both sides of the equation.

Nevertheless, for $\mu \approx 1e-5$, this preconditioner is the only one scaling well with respect to the temporal and spatial discretization. The number of averaged outer iterations **Outer/Lin.sys.** increases only slightly when the mesh is refined or the time steps is halved. This results in a CPU time that scales sightly worse than linearly with respect to the number of degrees of freedom used to discretize the problem.

4.5 Summary of numerical results

In this section, we summarize the numerical results to compare the pros and cons of the preconditioning approaches proposed in this paper. In fig. 1 we compare the total CPU time (y-axis) required to achieve IP relaxation $\mu \approx 1e-5$ (eight IP iterations) with respect to the total number of degrees of freedom used, while halving the time step and the mesh size used. This summarizes the data in tables 1 to 3 with gray background color. We included in this comparison the results obtain for the test cases 1, 3 and 4.

The behavior of the preconditioner based on the primal Schur complement described in section 4.2 is determined by the inner solver used to solve the block linear system 14, which is the finite-dimensional counterpart of a time-space weighted and anisotropic Laplacian. This linear system was solved efficiently with the AGMG solver only for the smallest test cases during the initial IP steps, when the relaxation μ is relatively high. In fact, it did not reach the threshold $\mu \approx 1e-5$ in the finest discretizations for all test cases.

The SIMPLE preconditioner is rather robust with respect to different time steps and relaxation parameters μ. However, it becomes inefficient when large grids are used. In fact, the number of outer iterations and the CPU time approximately scale linearly with respect to the number of time steps and the number of interior point steps, but quadratically with respect to the number of cells. Moreover, it had some failures in the first IP steps for all test cases in the finest discretization. However, the SIMPLE preconditioner is the only one that becomes faster in the last IP iterations. Thus, it may be the only one providing a viable option when accurate solutions of the optimal transport problem are required, possibly combined with more efficient approaches for the initial IP steps.

The $\boldsymbol{BB}$-preconditioner described in section 4.4 is the only one able to tackle efficiently large scale problems, since the number of inner iterations required per each Krylov step remains rather constant, while the number of outer iterations increases only slightly using smaller time steps and finer grids. This holds as long as the IP tolerance is approximately $1e-5$. Using smaller values, the number of inner and outer iterations tends to increase, leading in some cases to the solver's failure. The reason causing this performance degradation is not clear and needs further investigations.

5 Conclusions

We presented different preconditioners for solving via iterative methods the saddle point linear systems arising when computing solutions of the Benamou-Brenier formulation using an IP method. The first preconditioner presented, based on the primal Schur complement, is neither efficient nor robust since its application requires the solution of a complicated linear system involving an anisotropic weighted Laplacian in the time-space domain. The second preconditioner, called SIMPLE, was shown to be remarkably robust but with a CPU time that scales quadratically with respect to the number of spatial degrees of freedom. The most efficient approach turned out to be a block triangular

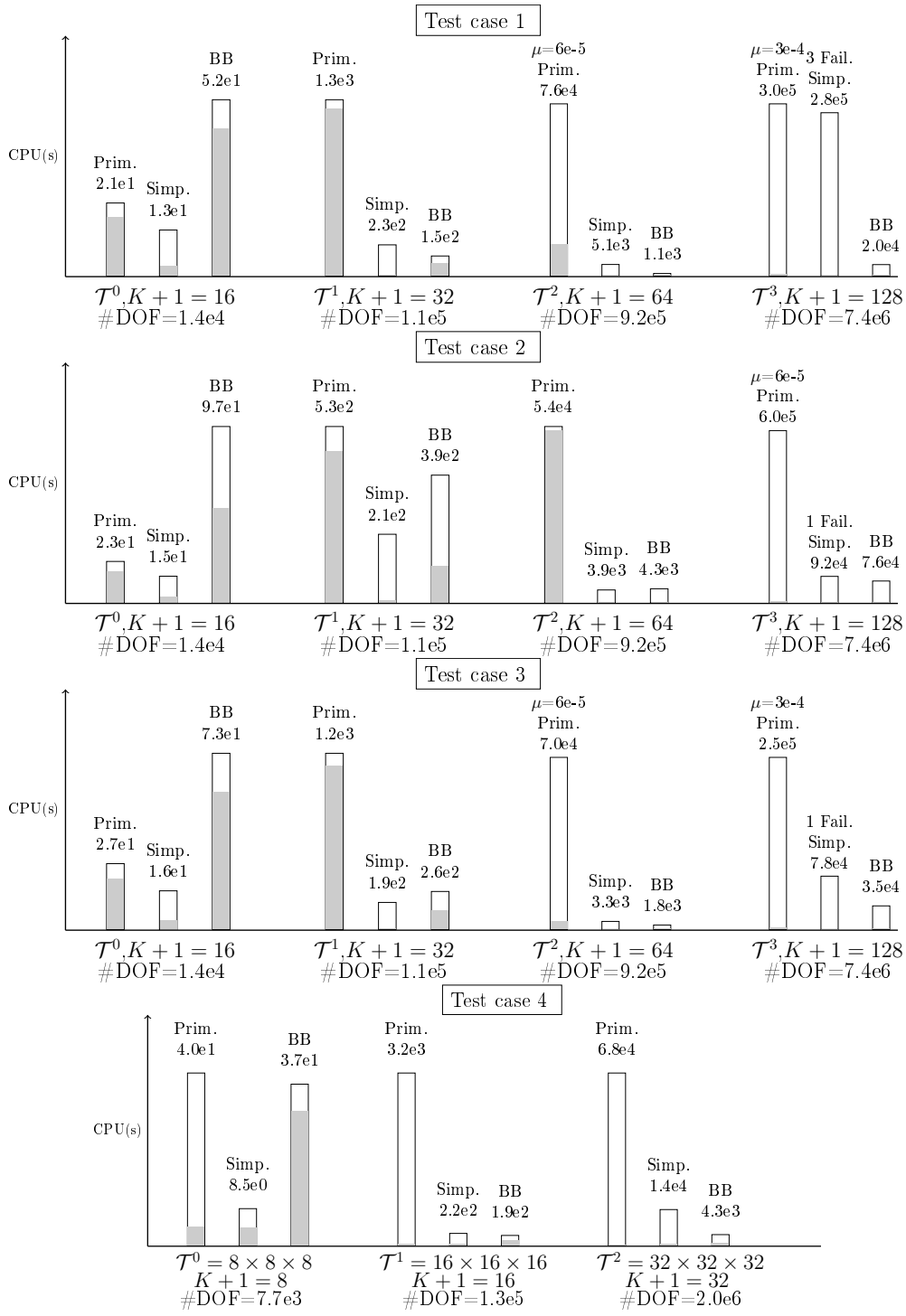

Figure 1: Comparison of CPU time spent in solving linear algebra problems to reach an IP relaxation $\mu \approx 1e-5$ using the preconditioners approaches presented in this paper. They are denoted by Prim. (the preconditioner based on the primal Schur complement in section 4.2, Simp. (the SIMPLE preconditioner in section 4.3) and BB (the $\boldsymbol{BB}$-preconditioner). The results refer to the test cases 1,2,3 and 4 (from top to bottom panel) and using different time-space discretization (left to right). The columns height is normalized by the maximum among the three preconditioners. The gray portion of the column denotes the part of the preprocessing time of each preconditioner. If $\mu \approx 1e-5$ was not achieved, we report the value μ reached. For the SIMPLE preconditioner some intermediate IP steps failed. We report the failures number and we do not sum any CPU time to the total cost, since the $\boldsymbol{BB}$-preconditioner appears to be the best among the three presented anyway.

preconditioner where the inverse of the dual Schur complement is approximated exploiting a partial commutation of its components, which we named $\boldsymbol{BB}$-preconditioner. Although this approach loses efficiency in the latest IP steps, it is the only one scaling well with respect to the time and space discretization size, both in 2d and 3d problems. Up to a reasonable value $\mu \approx 1e-5$ for the IP relaxation parameter, our numerical experiments showed that the problem can be solved with a good scaling of the CPU time with respect to the number of degrees of freedom. Combining it with further tuning strategies of the IP method (e.g. predictor-corrector methods, multilevel approach, parallelization) we believe it has the potential to provide highly efficient solvers for the dynamical optimal transport problem. Moreover, we remark that the same strategy can be applied to variations of the problem which consider further penalization of the density in eq. (1) (such as those considered in [45]) without major modifications. The deterioration of the preconditioner for smaller values of μ requires further investigation and it is left for future works.

Acknowledgments

The bulk of this work was completed while E.F was affiliated with Univ. Lille, Inria, CNRS, UMR 8524 - Laboratoire Paul Painlevé, as member of the RAPSODI team. E.F. was partially funded by the Marie Skłodowska-Curie Action (HE MSCA PF 101103631). A.N. was partially funded by the Labex CEMPI (ANR-11-LABX-0007-01). The work of M. B. was supported in part by PNRR MUR Project PE0000013-FAIR.

References

[1] *Fvcav, benchmark*, 2008.

[2] Y. ACHDOU, F. CAMILLI, AND I. CAPUZZO-DOLCETTA, *Mean field games: numerical methods for the planning problem*, SIAM Journal on Control and Optimization, 50 (2012), pp. 77–109.

[3] Y. ACHDOU AND V. PEREZ, *Iterative strategies for solving linearized discrete mean field games systems*, Networks and Heterogeneous Media, 7 (2012), pp. 197–217.

[4] L. AMBROSIO, E. BRUÉ, AND D. SEMOLA, *Lectures on Optimal Transport*, Springer, 2021.

[5] L. AMBROSIO, N. GIGLI, AND G. SAVARÉ, *Gradient flows: in Metric Spaces and in the Space of Probability Measures*, Springer Science & Business Media, 2005.

[6] A. BARADAT AND H. LAVENANT, *Regularized unbalanced optimal transport as entropy minimization with respect to branching brownian motion*, arXiv preprint arXiv:2111.01666, (2021).

[7] S. BELLAVIA, *Inexact interior-point method*, Journal of Optimization Theory and Applications, 96 (1998), pp. 109–121.

[8] J.-D. BENAMOU AND Y. BRENIER, *A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem*, Numerische Mathematik, 84 (2000), pp. 375–393.

[9] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.

[10] M. BENZI, E. HABER, AND L. TARALLI, *Multilevel algorithms for large-scale interior point methods*, SIAM Journal on Scientific Computing, 31 (2009), pp. 4152–4175.

[11] ——, *A preconditioning technique for a class of pde-constrained optimization problems*, Advances in Computational Mathematics, 35 (2011), pp. 149–173.

[12] L. BERGAMASCHI, J. GONDZIO, AND G. ZILLI, *Preconditioning indefinite systems in interior point methods for optimization*, Computational Optimization and Applications, 28 (2004), pp. 149–171.

[13] P. BOCHEV AND R. B. LEHOUCQ, *On the finite element solution of the pure neumann problem*, SIAM Review, 47 (2005), pp. 50–66.

[14] S. BOYD AND L. VANDENBERGHE, *Convex Optimization*, Cambridge University Press, 2004.

[15] L. A. CAFFARELLI, *Monotonicity properties of optimal transportation and the FKG and related inequalities*, Communications in Mathematical Physics, 214 (2000), pp. 547–563.

[16] L. CHIZAT, G. PEYRÉ, B. SCHMITZER, AND F.-X. VIALARD, *An interpolating distance between optimal transport and fisher–rao metrics*, Foundations of Computational Mathematics, 18 (2018), pp. 1–44.

[17] M. COLOMBO, A. FIGALLI, AND Y. JHAVERI, *Lipschitz changes of variables between perturbations of log-concave measures*, Annali Scuola Normale Superiore - Classe di scienze, (2017), pp. 1491–1519.

[18] M. D'APUZZO, V. DE SIMONE, AND D. DI SERAFINO, *On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods*, Computational Optimization and Applications, 45 (2010), pp. 283–310.

[19] H. ELMAN, *Preconditioning strategies for models of incompressible flow*, Journal of Scientific Computing, 25 (2005), pp. 347–366.

[20] H. ELMAN, V. E. HOWLE, J. SHADID, R. SHUTTLEWORTH, AND R. TUMINARO, *Block preconditioners based on approximate commutators*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1651–1668.

[21] H. ELMAN, V. E. HOWLE, J. SHADID, D. SILVESTER, AND R. TUMINARO, *Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations*, SIAM Journal on Scientific Computing, 30 (2007), pp. 290–311.

[22] H. ELMAN, D. SILVESTER, AND A. J. WATHEN, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Oxford university press, 2014.

[23] M. ERBAR, M. RUMPF, B. SCHMITZER, AND S. SIMON, *Computation of optimal transport on discrete metric measure spaces*, Numerische Mathematik, 144 (2020), pp. 157–200.

[24] R. EYMARD, T. GALLOUËT, AND R. HERBIN, *Finite volume methods*, Handbook of numerical analysis, 7 (2000), pp. 713–1018.

[25] E. FACCA AND M. BENZI, *Fast iterative solution of the optimal transport problem on graphs*, SIAM Journal on Scientific Computing, 43 (2021), pp. A2295–A2319.

[26] E. FACCA, F. CARDIN, AND M. PUTTI, *Towards a stationary Monge-Kantorovich dynamics: The Physarum Polycephalum experience*, SIAM Journal on Applied Mathematics, 78 (2018), pp. 651–676.

[27] E. FACCA, S. DANERI, F. CARDIN, AND M. PUTTI, *Numerical solution of Monge–Kantorovich equations via a dynamic formulation*, Journal of Scientific Computing, 82 (2020), pp. 1–26.

[28] A. GALICHON, *Optimal transport methods in economics*, in Optimal Transport Methods in Economics, Princeton University Press, 2016.

[29] P. GLADBACH, E. KOPFER, AND J. MAAS, *Scaling limits of discrete optimal transport*, SIAM Journal on Mathematical Analysis, 52 (2020), pp. 2759–2802.

[30] J. GONDZIO, *Interior point methods 25 years later*, European Journal of Operational Research, 218 (2012), pp. 587–601.

[31] ——, *Convergence analysis of an inexact feasible interior point method for convex quadratic programming*, SIAM Journal on Optimization, 23 (2013), pp. 1510–1527.

[32] E. HABER AND R. HORESH, *A multilevel method for the solution of time dependent optimal transport*, Numerical Mathematics: Theory, Methods and Applications, 8 (2015), pp. 97–111.

[33] L. KAMENSKI, W. HUANG, AND H. XU, *Conditioning of finite element equations with arbitrary anisotropic meshes*, Mathematics of computation, 83 (2014), pp. 2187–2211.

[34] H. LAVENANT, *Unconditional convergence for discretizations of dynamical optimal transport*, Mathematics of Computation, 90 (2021), pp. 739–786.

[35] H. LAVENANT, S. CLAICI, E. CHIEN, AND J. SOLOMON, *Dynamical optimal transport on discrete surfaces*, ACM Transactions on Graphics (TOG), 37 (2018), pp. 1–16.

[36] L. MÉTIVIER, R. BROSSIER, F. KPADONOU, J. MESSUD, AND A. PLADYS, *A review of the use of optimal transport distances for high resolution seismic imaging based on the full waveform*, MathematicS In Action, 11 (2022), pp. 3–42.

[37] B. MORINI, V. SIMONCINI, AND M. TANI, *A comparison of reduced and unreduced KKT systems arising from interior point methods*, Computational Optimization and Applications, 68 (2017), pp. 1–27.

[38] A. NATALE AND G. TODESCHI, *Computation of optimal transport with finite volumes*, ESAIM: Mathematical Modelling and Numerical Analysis, 55 (2021), pp. 1847–1871.

[39] ——, *A mixed finite element discretization of dynamical optimal transport*, Journal of Scientific Computing, 91 (2022), pp. 1–26.

[40] Y. NOTAY, *An aggregation-based algebraic multigrid method*, Electronic Transactions on Numerical Analysis, 37 (2010), pp. 123–146.

[41] N. PAPADAKIS, G. PEYRÉ, AND E. OUDET, *Optimal transport with proximal splitting*, SIAM Journal on Imaging Sciences, 7 (2014), pp. 212–238.

[42] S. PATANKAR AND D. SPALDING, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, International Journal of Heat and Mass Transfer, 15 (1972), pp. 1787–1806.

[43] S. V. PATANKAR, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Pub. Corp. ; McGraw-Hill Washington : New York, 1980.

[44] G. PEYRÉ, M. CUTURI, ET AL., *Computational optimal transport: With applications to data science*, Foundations and Trends® in Machine Learning, 11 (2019), pp. 355–607.

[45] A. PORRETTA, *Regularizing effects of the entropy functional in optimal transport and planning problems*, Journal of Functional Analysis, 284 (2023), p. 109759.

[46] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific & Statistical Computing, 14 (1993), pp. 461–469.

[47] F. SANTAMBROGIO, *Optimal Transport for Applied Mathematicians*, Birkäuser, NY, 55 (2015).

[48] S. I. VALDIMARSSON, *On the hessian of the optimal transport potential*, Annali della Scuola Normale Superiore di Pisa - Classe di Scienze, Ser. 5, 6 (2007), pp. 441–456.

[49] H. A. VAN DER VORST, *Iterative Krylov Methods for Large Linear Systems*, no. 13, Cambridge University Press, 2003.

[50] C. VILLANI, *Topics in Optimal Transportation*, Graduate studies in mathematics, American Mathematical Society, 2003.

[51] S. J. WRIGHT, *Primal-Dual Interior-Point Methods*, SIAM, 1997.