A NEW APPROACH TO IMPROVE ILL-CONDITIONED PARABOLIC OPTIMAL CONTROL PROBLEM VIA TIME DOMAIN DECOMPOSITION

Mohamed-Kamel Riahi

A NEW APPROACH TO IMPROVE ILL-CONDITIONED PARABOLIC OPTIMAL CONTROL PROBLEM VIA TIME DOMAIN DECOMPOSITION

Mohamed Kamel RIAHI*[1]

[1] Department of mathematical science, New Jersey Institute of Technology, University Heights Newark, New Jersey, USA.

January 14, 2015

Abstract. In this paper we present a new steepest-descent type algorithm for convex optimization problems. Our algorithm pieces the unknown into sub-blocs of unknowns and considers a partial optimization over each sub-bloc. In quadratic optimization, our method involves Newton technique to compute the step-lengths for the sub-blocs resulting descent directions. Our optimization method is fully parallel and easily implementable, we first presents it in a general linear algebra setting, then we highlight its applicability to a parabolic optimal control problem, where we consider the blocs of unknowns with respect to the time dependency of the control variable. The parallel tasks, in the last problem, turn"on" the control during a specific time-window and turn it "off" elsewhere. We show that our algorithm significantly improves the computational time compared with recognized methods. Convergence analysis of the new optimal control algorithm is provided for an arbitrary choice of partition. Numerical experiments are presented to illustrate the efficiency and the rapid convergence of the method.

Steepest descent method, Newton method, ill conditioned Optimal control, time domain decomposition.

1. Introduction

Typically the improvement of iterative methods is based on an implicit transformation of the original linear system in order to get a new system which has a condition number ideally close to one see [11, 13, 25] and references therein. This technique is known as preconditioning. Modern preconditioning techniques such as algebraic multilevel e.g. [20, 24] and domain decomposition methods e.g. [23, 27, 4, 15] attempt to produce efficient tools to accelerate convergence. Other techniques have introduced a different definition of the descent directions, for example, CG-method, GMRES, FGMRES, BFGS, or its limited memory version l-BFGS see for instance [25]. Others approaches (e.g. [5],[12] and [28] without being exhaustive) propose different formulas for the line-search in order to enhance the optimization procedure.

The central investigation of this paper is the enhancement of the iterations of the steepest descent algorithm via an introduction of a new formulation for the line-search. Indeed, we show how to achieve an optimal vectorized step-length for a given set of descent directions. Steepest descent methods [7] are usually used for solving, for example, optimization problems, control with partial differential equations (PDEs) constraints and inverse problems. Several approaches have been developed in the cases of constrained and unconstrained optimization.

It is well-known that the algorithm has a slow convergence rate with ill-conditioned problems because the number of iterations is proportional to the condition number of the problem. The method of J.Barzila and J.Borwein [2] based on two-point step-length for the steepest-descent method for approximating the secant equation avoids this handicap. Our method is very different

*Mohamed Kamel RIAHI : riahi@njit.edu http://web.njit.edu/~riahi

because first, it is based on a decomposition of the unknown and proposes a set of bloc descent directions, and second because it is general where it can be coupled together with any least-square-like optimization procedure.

The theoretical basis of our approach is presented and applied to the optimization of a positive definite quadratic form. Then we apply it on a complex engineering problem involving control of system governed by PDEs. We consider the optimal heat control which is known to be ill-posed in general (and well-posed under some assumptions) and presents some particular theoretical and numerical challenges. We handle the ill-posedness degree of the heat control problem by varying the regularization parameter and apply our methods in the handled problem to show the efficiency of our algorithm. The distributed- and boundary-control cases are both considered.

This paper is organized as follows: In Section 2, we present our method in a linear algebra framework to highlight its generality. Section 3 is devoted to the introduction of the optimal control problem with constrained PDE on which we will apply our method. We present the Euler-Lagrange-system associated to the optimization problem and give the explicit formulation of the gradient in both cases of distributed- and boundary-control. Then, we present and explain the parallel setting for our optimal control problem. In Section 4, we perform the convergence analysis of our parallel algorithm. In Section 5, we present the numerical experiments that demonstrate the efficiency and the robustness of our approach. We make concluding remarks in Section 6. For completeness, we include calculus results in the Appendix.

Let Ω be a bounded domain in $\mathbb{R}^3$, and $\Omega_c \subset \Omega$, the boundary of Ω is denoted by $\partial\Omega$. We denote by $\Gamma \subset \partial\Omega$ a part of this boundary. We denote $\langle .,. \rangle_2$ (respectively $\langle .,. \rangle_c$ and $\langle .,. \rangle_\Gamma$) the standard $L^2(\Omega)$ (respectively $L^2(\Omega_c)$ and $L^2(\Gamma)$) inner-product that induces the $L^2(\Omega)$-norm $\|.\|_2$ on the domain Ω (respectively $\|\cdot\|_c$ on Ω_c and $\|\cdot\|_\Gamma$ on Γ).

In the case of finite dimensional vector space in $\mathbb{R}^m$, the scalar product $a^T b$ of a and b (where a^T stands for the transpose of a) is denoted by $\langle .,. \rangle_2$ too. The scalar product with respect to the matrix A, i.e. $\langle x, Ax \rangle_2$ is denote by $\langle x, x \rangle_A$ and its induced norm is denoted by $\|x\|_A$. The transpose of the operator A is denoted by A^T. The Hilbert space $L^2(0,T;L^2(\Omega_c))$ (respectively $L^2(0,T;L^2(\Gamma))$) is endowed by the scalar product $\langle .,. \rangle_{c,I}$ (respectively $\langle .,. \rangle_{\Gamma,I}$) that induces the norm $\|.\|_{c,I}$ (respectively $\|.\|_{\Gamma,I}$).

2. Enhanced steepest descent iterations

The steepest descent algorithm minimizes at each iteration the quadratic function $q(x) = \|x - x^\star\|_A^2$, where A is assumed to be a symbiotic positive definite (SPD) matrix and $x^\star$ is the minimum of q. The vector $-\nabla q(x)$ is locally the descent direction that yields the fastest rate of decrease of the quadratic form q. Therefore all vectors of the form $x+\theta\nabla q(x)$, where θ is a suitable negative real value, minimize q. The choice of θ is found by looking for the $\min_{s<0} q(x+s\nabla q(x))$ with the use of a line-search technique. In the case where q is a quadratic form θ is given by $-\|\nabla q(x)\|_2^2/\|\nabla q(x)\|_A^2$. We recall in Algorithm 1 the steepest descent algorithm; *Convergence* is a boolean variable based on estimation of the residual vector $r^k < \epsilon$, where ϵ is the stopping criterion.

Our method proposes to modify the stepe 5. of Algorithm 1. It considers the step-length $\theta \in \mathbb{R}_-\backslash\{0\}$ as a vector in $\mathbb{R}_-^{\hat{n}}\backslash\{\mathbf{0}\}$ where $\hat{n}$ is an integer such that $1 \leq \hat{n} \leq size(x)$, we shall denote this new vector as $\Theta_{\hat{n}}$.

In the following, it is assumed that for a giving vector $x \in \mathbb{R}^m$, the integer $\hat{n}$ divides m with null rest. In this context, let us introduce the identity operators $I_{\mathbb{R}^m}$ witch is an m-by-m matrix and its partition (partition of unity) given by the projection operators $\{\boldsymbol{\pi}_n\}_{n=1}^{\hat{n}}$: projectors from

Algorithm 1: Steepest descent.

Input: x^0;
$k = 0$;
while *Convergence* **do**
  $r^k = \nabla q^k := \nabla q(x^k)$;
  Compute Ar^k;
  Compute $\theta^k = -\|r^k\|_2^2/\|r^k\|_A^2$;
  $x^{k+1} = x^k + \theta^k r^k$;
  $k = k + 1$;
end

$\mathbb{R}^m$ into a set of canonical basis $\{e_i\}_i$. These operators are defined for $1 \leq n \leq \hat{n}$ by

$$\begin{aligned} \boldsymbol{\pi}_n \quad & : \mathbb{R}^m \to \mathbb{R}^{\frac{m}{\hat{n}}} \\ & x \mapsto \boldsymbol{\pi}_n(x) = \sum_{i=(n-1)\times\frac{m}{\hat{n}}+1}^{n\times\frac{m}{\hat{n}}} \langle e_i, x\rangle_2 \, e_i. \end{aligned}$$

For reading conveniences, we define $\tilde{x}_n$ a vector in $\mathbb{R}^m$ such that $\tilde{x}_n := \boldsymbol{\pi}_n(x)$. The concatenation of $\tilde{x}_n$ for all $1 \leq n \leq \hat{n}$ is denoted by

$$\hat{x}_{\hat{n}} = \bigoplus_{n=1}^{\hat{n}} \boldsymbol{\pi}_n(x) = \bigoplus_{n=1}^{\hat{n}} \tilde{x}_n \in \mathbb{R}^m.$$

We remark that $\boldsymbol{\pi}_n$ satisfy $\bigoplus_{n=1}^{\hat{n}} \boldsymbol{\pi}_n = I_{\mathbb{R}^m}$.

Recall the gradient $\nabla_x = (\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_m})^T$, and define the bloc gradient $\nabla_{\hat{x}_{\hat{n}}} = \left(\nabla_{\tilde{x}_1}^T, \ldots, \nabla_{\tilde{x}_{\hat{n}}}^T\right)^T$, where obviously $\nabla_{\tilde{x}_n}^T = (\frac{\partial}{\partial x_{(n-1)\times\frac{m}{\hat{n}}+1}}, \ldots, \frac{\partial}{\partial x_{n\times\frac{m}{\hat{n}}}})^T$. In the spirit of this decomposition we investigate, in the sequel, the local descent directions as the bloc partial derivatives with respect to the bloc-variables $(\tilde{x}_n)_{n=1}^{n=\hat{n}}$. We aim, therefore, at finding $\Theta_{\hat{n}} = (\theta_1, \ldots, \theta_{\hat{n}})^T \in \mathbb{R}^{\hat{n}}$ that ensures the $\min_{(\theta_n)_n<0} q\left(\hat{x}_{\hat{n}}^k + \bigoplus_{n=1}^{\hat{n}} \theta_n \nabla_{\tilde{x}_n} q(\hat{x}_{\hat{n}}^k)\right)$.

We state hereafter a motivating result, which its proof is straightforward because the spaces are embedded. Let us first, denote by

$$\begin{aligned} \Phi_{\hat{n}}(\Theta_{\hat{n}}): \quad & \mathbb{R}^{\hat{n}} \to \mathbb{R}_+ \\ & \Theta_{\hat{n}} \mapsto q\left(\hat{x}_{\hat{n}} + \bigoplus_{n=1}^{\hat{n}} \theta_n \nabla_{\tilde{x}_n} q(\hat{x}_{\hat{n}})\right) \end{aligned} \tag{1}$$

which is quadratic because q is.

Theorem 2.1. *According to the definition of $\Phi_{\hat{n}}(\Theta_{\hat{n}})$ (see Eq.(1)) we immediately have*

$$\min_{\mathbb{R}^p} \Phi_p(\Theta_p) \leq \min_{\mathbb{R}^q} \Phi_q(\Theta_q) \quad \forall q < p.$$

The new algorithm we discuss in this paper proposes to define a sequence $(\hat{x}_{\hat{n}}^k)_k$ of vectors that converges to $x^\star$ unique minimizer of the quadratic form q. The update formulae reads:

$$\tilde{x}_n^{k+1} = \tilde{x}_n^k + \theta_n^k \nabla_{\tilde{x}_n} q(\hat{x}_{\hat{n}}^k),$$

where we recall that $\hat{n}$ is an arbitrarily chosen integer. Then $\hat{x}_{\hat{n}}^{k+1} = \bigoplus_{n=1}^{\hat{n}} \tilde{x}_n^{k+1}$.

We shall explain now how one can accurately computes the vector step-length $\Theta_{\hat{n}}^k$ at each iteration k. It is assumed that q is a quadratic form. From Eq.(1) using the chain rule, we obtain

the Jacobian vector $\Phi'_{\hat{n}}(\Theta_{\hat{n}}) \in \mathbb{R}^{\hat{n}}$ given by

$$(\Phi'_{\hat{n}}(\Theta_{\hat{n}}))_j = \left(\nabla_{\tilde{x}_j} q(\hat{x}^k_{\hat{n}})\right)^T \nabla_{\tilde{x}_j} q\left(\hat{x}^k_{\hat{n}} + \bigoplus_{n=1}^{\hat{n}} \theta_n \nabla_{\tilde{x}_n} q(\hat{x}^k_{\hat{n}})\right) \in \mathbb{R}, \tag{2}$$

and the Hessian matrix $\Phi''_{\hat{n}}(\Theta_{\hat{n}}) \in \mathbb{R}^{\hat{n}\times\hat{n}}$ is given by

$$(\Phi''_{\hat{n}}(\Theta_{\hat{n}}))_{i,j} = \left(\nabla_{\tilde{x}_j} q(\hat{x}^k_{\hat{n}})\right)^T \left(\nabla_{\tilde{x}_i}\nabla_{\tilde{x}_j} q\left(\hat{x}^k_{\hat{n}} + \bigoplus_{n=1}^{\hat{n}} \theta_n \nabla_{\tilde{x}_n} q(\hat{x}^k_{\hat{n}})\right)\right) \nabla_{\tilde{x}_j} q(\hat{x}^k_{\hat{n}}).$$

It is worth noticing that the matrix $\nabla_{\tilde{x}_i}\nabla_{\tilde{x}_j} q\left(\hat{x}^k_{\hat{n}} + \bigoplus_{n=1}^{\hat{n}} \theta_n \nabla_{\tilde{x}_n} q(\hat{x}^k_{\hat{n}})\right)$ is a bloc portion of the Hessian matrix A. However if the gradient $\nabla_{\tilde{x}_n} q \in \mathbb{R}^{\frac{m}{\hat{n}}}$ assumes an extension by zero (denoted by $\widetilde{\nabla}_{\tilde{x}_i} q$) to $\mathbb{R}^m$ so the matrix $\Phi''_{\hat{n}}(\Theta_{\hat{n}})$ has therefore the simplest implementable form

$$(\Phi''_{\hat{n}}(\Theta_{\hat{n}}))_{i,j} = (\widetilde{\nabla}_{\tilde{x}_j} q(\hat{x}^k_{\hat{n}}))^T A \widetilde{\nabla}_{\tilde{x}_j} q(\hat{x}^k_{\hat{n}}). \tag{3}$$

We thus have the expansion $\Phi_{\hat{n}}(\Theta^k_{\hat{n}}) = \Phi_{\hat{n}}(\mathbf{0}) + (\Theta^k_{\hat{n}})^T \Phi_{\hat{n}}(\mathbf{0}) + \frac{1}{2}(\Theta^k_{\hat{n}})^T \Phi''_{\hat{n}}(\mathbf{0})\Theta^k_{\hat{n}}$, with $\mathbf{0} := (0, .., 0)^T \in \mathbb{R}^{\hat{n}}$. Then the vector $\Theta^k_{\hat{n}}$ that annuls the gradient writes:

$$\Theta^k_{\hat{n}} = -\Phi''_{\hat{n}}(\mathbf{0})^{-1} \Phi'_{\hat{n}}(\mathbf{0}). \tag{4}$$

Algorithm 1 has therefore a bloc structure which can be solved in parallel. This is due to the fact that partial derivatives can be computed independently. The new algorithm is thus as follows (see Algorithm 2)

Algorithm 2: Enhanced steepest descent.

$k = 0$;
Input: $\hat{x}^0_{\hat{n}} \in \mathbb{R}^m$;
while *Convergence* **do**
  forall the $1 \leq n \leq \hat{n}$ **do**
    $\tilde{x}^k_n = \boldsymbol{\pi}_n(\hat{x}^k_{\hat{n}})$;
    $r_n = \nabla_{\tilde{x}^k_n} q(\hat{x}^k_{\hat{n}})$;
    $resize(r_n)$ (i.e. extension by zero means simply project on $\mathbb{R}^m$);
  end
  Assemble $\Phi'_{\hat{n}}(\mathbf{0})$ with element $(\Phi'_{\hat{n}}(\mathbf{0}))_j = r_j^T r_j$ according to Eq.(2);
  Assemble $\Phi''_{\hat{n}}(\mathbf{0})$ with element $(\Phi''_{\hat{n}}(\mathbf{0}))_{i,j} = r_i^T A r_j$ according to Eq.(3);
  Compute $\Theta^k_{\hat{n}}$ solution of Eq.(4);
  Update $\hat{x}^{k+1}_{\hat{n}} = \hat{x}^k_{\hat{n}} + \bigoplus_n \theta_n \nabla_{\tilde{x}_n} q(\hat{x}^k_{\hat{n}})$;
  $k = k + 1$;
end

3. Application to a parabolic optimal control problem

In this part we are interested in the application of Algorithm 2 in a finite element computational engineering problem involving optimization with constrained PDE. In particular, we deal with the optimal control problem of a system, which is governed by the heat equation. We shall present two types of control problems. The first concerns the distributed optimal control and

the second concerns the Dirichlet boundary control. The main difference from the algorithm just presented in linear algebra is that the decomposition is applied on the time domain when the control. This technique is not classical, we may refer to a similar approaches that has been proposed for the time domain decomposition in application to the control problem, for instance [19, 17, 18] which basically they use a variant of the parareal in time algorithm [15].

3.1. **Distributed optimal control problem.** Let us briefly present the steepest descent method applied to the following optimal control problem: find $v^\star$ such that

$$J(v^\star) = \min_{v\in L^2(0,T;L^2(\Omega_c))} J(v), \tag{5}$$

where J is a quadratic cost functional defined by

$$J(v) = \frac{1}{2}\|y(T) - y^{target}\|_2^2 + \frac{\alpha}{2}\int_I \|v\|_c^2 dt, \tag{6}$$

where y^{target} is a given target state and $y(T)$ is the state variable at time $T > 0$ of the heat equation controlled by the variable v over $I := [0, T]$. The Tikhonov regularization parameter α is introduced to penalize the control's L^2-norm over the time interval I. The optimality system of our problem reads:

$$\begin{cases} \partial_t y - \sigma\Delta y = \mathcal{B}v, & \text{on } I\times\Omega,\\ y(t=0) = y_0. \end{cases} \tag{7}$$

$$\begin{cases} \partial_t p + \sigma\Delta p = 0, & \text{on } I\times\Omega,\\ p(t=T) = y(T) - y^{target}. \end{cases} \tag{8}$$

$$\nabla J(v) = \alpha v + \mathcal{B}^T p = 0, \text{ on } I\times\Omega. \tag{9}$$

In the above equations, the operator $\mathcal{B}$ is a linear operator that distributes the control in Ω_c, obviously $\mathcal{B}$ stands for the indicator of $\Omega_c \subset \Omega$, the state variable p stands for the Lagrange multiplier (adjoint state) solution of the backward heat equation Eq.(8), Eq.(7) is called the forward heat equation.

3.2. **Dirichlet boundary optimal control problem.** In this subsection we are concerned with the PDE constrained Dirichlet boundary optimal control problem, where we aim at minimizing the cost functional J_Γ defined by

$$J_\Gamma(v_\Gamma) = \frac{1}{2}\|y_\Gamma(T) - y^{target}\|_2^2 + \frac{\alpha}{2}\int_I \|v_\Gamma\|_\Gamma^2 dt, \tag{10}$$

where the control variable v_Γ is only acting on the boundary $\Gamma \subset \partial\Omega$. Here too, y^{target} is a given target state (not necessary equal the one defined in the last subsection !) and $y_\Gamma(T)$ is the state variable at time $T > 0$ of the heat equation controlled by the variable v_Γ during the time interval $I := [0, T]$. As before α is a regularization term. The involved optimality system reads

$$\begin{cases} \partial_t y_\Gamma - \sigma\Delta y_\Gamma = f & \text{on } I\times\Omega\\ y_\Gamma = v_\Gamma & \text{on } I\times\Gamma\\ y_\Gamma = g & \text{on } I\times\{\partial\Omega\backslash\Gamma\}\\ y_\Gamma(0) = y_0 \end{cases} \tag{11}$$

$$\begin{cases} \partial_t p_\Gamma + \sigma\Delta p_\Gamma = 0 & \text{on } I\times\Omega\\ p_\Gamma = 0 & \text{on } I\times\partial\Omega\\ p_\Gamma(T) = y_\Gamma(T) - y^{target} \end{cases} \tag{12}$$

$$\nabla J_\Gamma(v_\Gamma) = \alpha v_\Gamma - (\nabla p_\Gamma)^T\vec{n} = 0 \quad \text{on } I\times\Gamma, \tag{13}$$

where $f \in L^2(\Omega)$ is any source term, $g \in L^2(\Gamma)$ and $\vec{n}$ is the outward unit normal on Γ. the state variable p_Γ stands for the Lagrange multiplier (adjoint state) solution of the backward heat

equation Eq.(12). Both functions f and g will be given explicitly for each numerical test that we consider in the numerical experiment section.

3.3. **Steepest descent algorithm for optimal control of constrained PDE.** In the optimal control problem, the evaluation of the gradient as it is clear in Eq.(9) (respectively (13)) requires the evaluation of the time dependent Lagrange multiplier p (respectively p_Γ). This fact, makes the steepest descent optimization algorithm slightly differs from the Algorithm 1 already presented.

Let us denote by k the current iteration superscript. We suppose that v^0 is known. The first order steepest descent algorithm updates the control variable as follows:

$$v^k = v^{k-1} + \theta^{k-1}\nabla J(v^{k-1}), \text{ for } k \geq 1, \quad \text{for the distributed control} \tag{14}$$

respectively as

$$v_\Gamma^k = v_\Gamma^{k-1} + \theta_\Gamma^{k-1}\nabla J_\Gamma(v_\Gamma^{k-1}), \text{ for } k \geq 1, \quad \text{for the Dirichlet control} \tag{15}$$

The step-length $\theta^{k-1} \in \mathbb{R}^-\backslash\{0\}$ in the direction of the gradient $\nabla J(v^{k-1}) = \alpha v^{k-1} + \mathcal{B}^T p^{k-1}$ (respectively $\nabla J_\Gamma(v_\Gamma^{k-1}) = \alpha v_\Gamma - (\nabla p_\Gamma)^T \vec{n}$) is computed as :

$$\theta^{k-1} = -\|\nabla J(v^{k-1})\|^2_{c,I}/\|\nabla J(v^{k-1})\|^2_{\nabla^2 J} \quad \text{for the distributed control.}$$

respectively as

$$\theta_\Gamma^{k-1} = -\|\nabla J_\Gamma(v_\Gamma^{k-1})\|^2_{c,I}/\|\nabla J_\Gamma(v_\Gamma^{k-1})\|^2_{\nabla^2 J_\Gamma} \quad \text{for the Dirichlet control.}$$

The above step-length θ^{k-1} (respectively θ_Γ^{k-1}) is optimal (see e.g. [8]) in the sense that it minimizes the functional $\theta \to J(v^{k-1} + \theta\nabla J(v^{k-1}))$ (respectively $\theta \to J_\Gamma(v_\Gamma^{k-1} + \theta\nabla J_\Gamma(v_\Gamma^{k-1}))$). The rate of convergence of this technique is $\left(\frac{\underline{\kappa}-1}{\underline{\kappa}+1}\right)^2$, where $\underline{\kappa}$ is the condition number of the quadratic form, namely the Hessian of the cost functional J (respectively J_Γ).

3.4. **Time-domain decomposition algorithm.** Consider $\hat{n}$ subdivisions of the time interval $I = \cup_{n=1}^{\hat{n}} I_n$, consider also the following convex cost functional J:

$$J(v_1, v_2, .., v_{\hat{n}}) = \frac{1}{2}\|\mathcal{Y}(T) - y^{target}\|_2^2 + \frac{\alpha}{2}\sum_{n=1}^{\hat{n}}\int_{I_n}\|v_n\|_c^2 dt, \tag{16}$$

$$J_\Gamma(v_{1,\Gamma}, v_{2,\Gamma}, .., v_{\hat{n},\Gamma}) = \frac{1}{2}\|\mathcal{Y}_\Gamma(T) - y^{target}\|_2^2 + \frac{\alpha}{2}\sum_{n=1}^{\hat{n}}\int_{I_n}\|v_n\|_\Gamma^2 dt, \tag{17}$$

where $v_n, n = 1, ..., \hat{n}$ are control variables with time support included in $I_n, n = 1, ..., \hat{n}$. The state $\mathcal{Y}(T)$ (respectively $\mathcal{Y}_{Gamma}$) stands for the sum of state variables $\mathcal{Y}_n$ (respectively $\mathcal{Y}_{n,\Gamma}$) which are time-dependent state variable solution to the heat equation controlled by the variable v_n (respectively $v_{n,\Gamma}$). . Obviously because the control is linear the state $\mathcal{Y}$ depends on the concatenation of controls $v_1, v_2, .., v_{\hat{n}}$ namely $v = \sum_{n=1}^{n=\hat{n}} v_n$.

Let us define $\Theta_{\hat{n}} := (\theta_1, \theta_2, ..., \theta_{\hat{n}})^T$ where $\theta_n \in \mathbb{R}_-\backslash\{0\}$. For any admissible control $w = \sum_n^{\hat{n}} w_n$, we also define $\varphi_{\hat{n}}(\Theta_{\hat{n}}) := J(v + \sum_{n=1}^{\hat{n}} \theta_n w_n)$, which is quadratic. We have:

$$\varphi_{\hat{n}}(\Theta_{\hat{n}}) = \varphi_{\hat{n}}(\mathbf{0}) + \Theta_{\hat{n}}^T\nabla\varphi_{\hat{n}}(\mathbf{0}) + \frac{1}{2}\Theta_{\hat{n}}^T\nabla^2\varphi_{\hat{n}}(\mathbf{0})\Theta_{\hat{n}}, \tag{18}$$

where $\mathbf{0} = (0, ..., 0)^T$. Therefore we can write $\nabla\varphi_{\hat{n}}(\Theta_{\hat{n}}) \in \mathbb{R}^{\hat{n}}$ as $\nabla\varphi_{\hat{n}}(\Theta_{\hat{n}}) = D(v, w) + H(v, w)\Theta_{\hat{n}}$, where the Jacobian vector and the Hessian matrix are given respectively by:

$$\begin{aligned} D(v,w) &:= (\langle\nabla J(v), \boldsymbol{\pi}_1(w)\rangle_c, \ldots, \langle\nabla J(v), \boldsymbol{\pi}_{\hat{n}}(w)\rangle_c)^T \in \mathbb{R}^{\hat{n}}, \\ H(v,w) &:= (H_{n,m})_{n,m}, \text{ for } H_{n,m} = \langle\boldsymbol{\pi}_n(w), \boldsymbol{\pi}_m(w)\rangle_{\nabla^2 J}. \end{aligned}$$

Here, $(\boldsymbol{\pi}_n)$ is the restriction over the time interval I_n, indeed $\boldsymbol{\pi}_n(w)$ has support on I_n and assumes extension by zero in I. The solution $\Theta^{\star}_{\hat{n}}$ of $\nabla\varphi_{\hat{n}}(\Theta_{\hat{n}}) = \mathbf{0}$ can be written in the form:

$$\Theta^{\star}_{\hat{n}} = -H^{-1}(v,w)D(v,w). \tag{19}$$

In the parallel distributed control problem, we are concerned with the following optimality system:

$$\begin{cases} \partial_t \mathcal{Y}_n - \sigma\Delta\mathcal{Y}_n = \mathcal{B}v_n, \quad \text{on } I\times\Omega, \\ \mathcal{Y}_n(t=0) = \delta^0_n y_0. \end{cases} \tag{20}$$

$$\mathcal{Y}(T) = \sum_{n=1}^{\hat{n}} \mathcal{Y}_n(T) \tag{21}$$

$$\begin{cases} \partial_t \mathcal{P} + \sigma\Delta\mathcal{P} = 0, \quad \text{on } I\times\Omega, \\ \mathcal{P}(t=T) = \mathcal{Y}(T) - y^{target}. \end{cases} \tag{22}$$

$$\nabla J(\sum_{n=1}^{\hat{n}} v_n) = \mathcal{B}^T\mathcal{P} + \alpha\sum_{n=1}^{\hat{n}} v_n = 0, \text{ on } I\times\Omega. \tag{23}$$

where δ^0_n stands for the function taking value "1" only if $n=0$, else it takes the value "0". The Dirichlet control problem we are concerned with:

$$\left\{\begin{array}{rcll} \partial_t \mathcal{Y}_{n,\Gamma} - \sigma\Delta\mathcal{Y}_{n,\Gamma} & = & f & \text{on } I\times\Omega \\ \mathcal{Y}_{n,\Gamma} & = & v_{n,\Gamma} & \text{on } I\times\Gamma \\ \mathcal{Y}_{n,\Gamma} & = & g & \text{on } I\times\{\partial\Omega\backslash\Gamma\} \\ \mathcal{Y}_{n,\Gamma}(0) & = & \delta^0_n y_0. & \end{array}\right. \tag{24}$$

$$\mathcal{Y}_{\Gamma}(T) = \sum_{n=1}^{\hat{n}} \mathcal{Y}_{n,\Gamma}(T) \tag{25}$$

$$\left\{\begin{array}{rcll} \partial_t \mathcal{P}_{\Gamma} + \sigma\Delta\mathcal{P}_{\Gamma} & = & 0 & \text{on } I\times\Omega \\ \mathcal{P}_{\Gamma} & = & 0 & \text{on } I\times\partial\Omega \\ \mathcal{P}_{\Gamma}(T) & = & & \mathcal{Y}_{\Gamma}(T) - y^{target}. \end{array}\right. \tag{26}$$

$$\nabla J_{\Gamma}(\sum_{n=1}^{\hat{n}} v_{n,\Gamma}) = -\big(\nabla\mathcal{P}_{\Gamma}\big)^T\vec{n} + \alpha\sum_{n=1}^{\hat{n}} v_{n,\Gamma} = 0 \quad \text{on } \ I\times\Gamma. \tag{27}$$

The resolution of Eqs. (20) and (24) with respect to n are fully performed in parallel over the time interval I. It is recalled that the superscript k denotes the iteration index. The update formulae for the control variable v^k is given by:

$$v^k_n = v^{k-1}_n + \theta^{k-1}_n\mathcal{B}^T\mathcal{P}^{k-1} + \alpha\sum_{n=1}^{\hat{n}} v^{k-1}_n.$$

respectively as

$$v^k_{n,\Gamma} = v^{k-1}_{n,\Gamma} + \theta^{k-1}_{n,\Gamma} - \big(\nabla\mathcal{P}^{k-1}_{\Gamma}\big)^T\vec{n} + \alpha\sum_{n=1}^{\hat{n}} v^{k-1}_{n,\Gamma}.$$

> We shall drop in the following the index $_{\Gamma}$ of the cost functional J. This index would be only used to specify which cost function is in consideration. unless the driven formulation apply for distributed as well as boundary control.

We show hereafter how to assemble vector step-length $\Theta^k_{\hat{n}}$ at each iteration. For the purposes of notation we denote by H_k the k-th iteration of the Hessian matrix $H(\nabla J(v^k), \nabla J(v^k))$ and by D_k the k-th iteration of the Jacobian vector $D(\nabla J(v^k), \nabla J(v^k))$. The line-search is performed

with quasi-Newton techniques that uses at each iteration k a Hessian matrix H_k and Jacobian vector D_k defined respectively by:

$$D_k := \left(\langle \nabla J(v^k), \boldsymbol{\pi}_1\left(\nabla J(v^k)\right)\rangle_c, .., \langle \nabla J(v^k), \boldsymbol{\pi}_{\hat{n}}\left(\nabla J(v^k)\right)\rangle_c\right)^T, \tag{28}$$

$$(H_k)_{n,m} := \langle \boldsymbol{\pi}_n\left(\nabla J(v^k)\right), \boldsymbol{\pi}_m\left(\nabla J(v^k)\right)\rangle_{\nabla^2 J}. \tag{29}$$

The spectral condition number of the Hessian matrix $\nabla^2 J$ is denoted as: $\underline{\kappa} = \underline{\kappa}(\nabla^2 J) := \lambda_{max}\lambda_{min}^{-1}$, with $\lambda_{max} := \lambda_{max}(\nabla^2 J)$ the largest eigenvalue of $\nabla^2 J$ and $\lambda_{min} := \lambda_{min}(\nabla^2 J)$ its smallest eigenvalue.

According to Eq.(19) we have

$$\Theta_{\hat{n}}^k = -H_k^{-1} D_k. \tag{30}$$

From Eq.(18) we have:

$$J(v^{k+1}) = J(v^k) + (\Theta_{\hat{n}}^k)^T D_k + \frac{1}{2}(\Theta_{\hat{n}}^k)^T H_k \Theta_{\hat{n}}^k. \tag{31}$$

Our parallel algorithm to minimize the cost functional Eq.(16) and (17), is stated as follows (see Algorithm 3).

Algorithm 3: Enhanced steepest descent algorithm for the optimal control problem.

0 **Input**: v^0
1 **while** *Convergence* **do**
2 **forall the** $1 \geq n \geq \hat{n}$ **do**
3 Solve $\mathcal{Y}_n(T)(v_n^k)$ of Eq.(20)(respectively Eq.(24)) **in parallel** for all $1 \leq n \leq \hat{n}$;
4 **end**
5 Compute $\mathcal{P}(t)$ with the backward problem according to Eq.(22) (respectively Eq.(26)) ;
6 **forall the** $1 \geq n \geq \hat{n}$ **do**
7 Compute $(D_k)_n$ of Eq.(28) **in parallel** for all $1 \leq n \leq \hat{n}$;
8 **end**
9 Gather $(D_k)_n$ from processor n, $2 \leq n \leq \hat{n}$ to **master processor**;
10 Assemble the Hessian matrix H_k according to Eq.(29) with **master processor**;
11 Compute the inversion of H_k and calculate $\Theta_{\hat{n}}^k$ using Eq.(30);
12 Broadcast θ_n^k from **master processor** to all slaves processors;
13 Update time-window-control variable v_n^{k+1} **in parallel** as :

$$v_n^{k+1} = v_n^k + \theta_n^k \boldsymbol{\pi}_n\left(\nabla J(v^k)\right) \quad \text{for all } 1 \leq n \leq \hat{n},$$

  and go to step 2;
14 $k = k + 1$;
15 **end**

Since $(v_n)_n$ has disjoint time-support, thanks to the linearity, the notation $e_n\big(\nabla J(v^k)\big)$ is nothing but $\nabla J(v_n^k)$, where v^k is the concatenation of $v_1^k, \ldots, v_{\hat{n}}^k$. In Algorithm 3 steps 9, 10, 11, 12 and 13 are trivial tasks in regards to computational effort.

4. Convergence analysis of Algorithm 3

This section provides the proof of convergence of Algorithm 3. In the sequel, we suppose that $\|\nabla J(v^k)\|_c$ does not vanish; otherwise the algorithm has already converged.

proposition 4.1. *The increase in value of the cost functional J between two successive controls v^k and v^{k+1} is bounded below by:*

$$J(v^k) - J(v^{k+1}) \geq \frac{1}{2\kappa(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|_{\nabla^2 J}^2}. \tag{32}$$

Proof. Using Eq.(30) and Eq.(31), we can write:

$$J(v^k) - J(v^{k+1}) = \frac{1}{2} D_k^T H_k^{-1} D_k. \tag{33}$$

Preleminaries: From the definition of the Jacobian vector D_k we have

$$\begin{aligned}
\|D_k\|_2^2 &= \sum_{n=1}^{\hat{n}} \langle \nabla J(v^k), \boldsymbol{\pi}_n(\nabla J(v^k)) \rangle_c^2, \\
&= \sum_{n=1}^{\hat{n}} \langle \boldsymbol{\pi}_n(\nabla J(v^k)), \boldsymbol{\pi}_n(\nabla J(v^k)) \rangle_c^2, \\
&= \sum_{n=1}^{\hat{n}} \|e_n(\nabla J(v^k))\|_c^4, \\
&= \|\nabla J(v^k)\|_c^4.
\end{aligned}$$

Furthermore since H_k is an SPD matrix we have $\lambda_{\min}(H_k^{-1}) = \frac{1}{\lambda_{\max}(H_k)}$, from which we deduce: $\frac{1}{\lambda_{\min}(H_k)} \geq \frac{1}{\frac{1}{\hat{n}} 1_{\hat{n}}^T H_k 1_{\hat{n}}}$. Moreover, we have:

$$\begin{aligned}
D_k^T H_k^{-1} D_k &= \frac{D_k^T H_k^{-1} D_k}{\|D_k\|_2^2} \|D_k\|_2^2 \geq \lambda_{min}(H_k^{-1}) \|D_k\|_2^2 \\
&= \lambda_{min}(H_k^{-1}) \lambda_{min}(H_k) \frac{\|\nabla J(v^k)\|_c^4}{\lambda_{min}(H_k)} \\
&\geq \frac{\lambda_{min}(H_k)}{\lambda_{max}(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\frac{1}{\hat{n}} 1_{\hat{n}}^T H_k 1_{\hat{n}}} \\
&= \frac{\hat{n}}{\kappa(H_k)} \|\nabla J(v^k)\|_{\nabla^2 J}^{-2} \|\nabla J(v^k)\|_c^4.
\end{aligned}$$

Since the partition number $\hat{n}$ is greater than or equal to 1, we conclude that :

$$D_k^T H_k^{-1} D_k \geq \frac{\|\nabla J(v^k)\|_{\nabla^2 J}^{-2} \|\nabla J(v^k)\|_c^4}{\kappa(H_k)}. \tag{34}$$

Hence, using Eq.(33) we get the stated result. □

Theorem 4.2. *For any partition $\hat{n}$ of sub intervals, the control sequence $(v^k)_{k\geq 1}$ of Algorithm 3 converges to the optimal control v^k unique minimizer of the quadratic functional J. Furthermore we have:*

$$\|v^k - v^\star\|_{\nabla^2 J}^2 \leq r^k \|v^0 - v^\star\|_{\nabla^2 J}^2,$$

where the rate of convergence $r := \left(1 - \frac{4\underline{\kappa}}{\kappa(H_k)(\underline{\kappa}+1)^2}\right)$ satisfies $0 \leq r < 1$.

Proof. We denote by $v^\star$ the optimal control that minimizes J. The equality

$$J(v) = J(v^\star) + \frac{1}{2}\langle v - v^\star, v - v^\star\rangle_{\nabla^2 J} = J(v^\star) + \frac{1}{2}\|v - v^\star\|^2_{\nabla^2 J},$$

holds for any control v; in particular we have:

$$\begin{aligned} J(v^{k+1}) &= J(v^\star) + \frac{1}{2}\|v^{k+1} - v^\star\|^2_{\nabla^2 J}, \\ J(v^k) &= J(v^\star) + \frac{1}{2}\|v^k - v^\star\|^2_{\nabla^2 J}. \end{aligned}$$

Consequently, by subtracting the equations above, we obtain

$$J(v^{k+1}) - J(v^k) = \frac{1}{2}\|v^{k+1} - v^\star\|^2_{\nabla^2 J} - \frac{1}{2}\|v^k - v^\star\|^2_{\nabla^2 J}. \tag{35}$$

Since J is quadratic, we have $\nabla^2 J(v^k - v^\star) = \nabla J(v^k)$, that is $v^k - v^\star = (\nabla^2 J)^{-1}\nabla J(v^k)$. Therefore we deduce:

$$\begin{aligned} \|v^k - v^\star\|^2_{\nabla^2 J} &= \langle v^k - v^\star, v^k - v^\star\rangle_{\nabla^2 J} \\ &= \langle v^k - v^\star, \nabla^2 J, v^k - v^\star\rangle_c \\ &= \langle (\nabla^2 J)^{-1}\nabla J(v^k), \nabla^2 J, (\nabla^2 J)^{-1}\nabla J(v^k)\rangle_c \\ &= \langle \nabla J(v^k), (\nabla^2 J)^{-1}, \nabla J(v^k)\rangle_c \\ &= \|\nabla J(v^k)\|^2_{(\nabla^2 J)^{-1}}. \end{aligned} \tag{36}$$

Because of Eq.(33), we also have

$$J(v^{k+1}) - J(v^k) = -\frac{1}{2} D_k^T H_k^{-1} D_k.$$

Using Eq.(35) and the above, we find that:

$$\|v^{k+1} - v^\star\|^2_{\nabla^2 J} = \|v^k - v^\star\|^2_{\nabla^2 J} - D_k^T H_k^{-T} D_k.$$

Moreover, according to Eqs (34)-(36), we obtain the following upper bound:

$$\begin{aligned} \|v^{k+1} - v^\star\|^2_{\nabla^2 J} &\leq \|v^k - v^\star\|^2_{\nabla^2 J} - \frac{1}{\kappa(H_k)}\frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|^2_{\nabla^2 J}} \\ &\leq \|v^k - v^\star\|^2_{\nabla^2 J}\Big(1 - \frac{1}{\kappa(H_k)}\frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|^2_{\nabla^2 J}\|\nabla J(v^k)\|^2_{(\nabla^2 J)^{-1}}}\Big). \end{aligned} \tag{37}$$

Using the Kantorovich inequality [14, 1] (see also The Appendix) :

$$\frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|^2_{\nabla^2 J}\|\nabla J(v^k)\|^2_{(\nabla^2 J)^{-1}}} \geq \frac{4\lambda_{max}\lambda_{min}}{(\lambda_{max} + \lambda_{min})^2}. \tag{38}$$

Then

$$1 - \frac{1}{\kappa(H_k)}\frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|^2_{\nabla^2 J}\|\nabla J(v^k)\|^2_{(\nabla^2 J)^{-1}}} \leq 1 - \frac{4\underline{\kappa}}{\kappa(H_k)(\underline{\kappa}+1)^2}.$$

Finally we obtain the desired results for any partition to $\hat{n}$ subdivision, namely

$$\|v^k - v^\star\|^2_{\nabla^2 J} \leq \Big(1 - \frac{4\underline{\kappa}}{\kappa(H_k)(\underline{\kappa}+1)^2}\Big)^k \|v^0 - v^\star\|^2_{\nabla^2 J}.$$

The proof is therefore complete. □

Remark 4.1. *Remark that the proof stands correct for the boundary control, need just to change the subscript "c" indicating the distributed control region Ω_c, replace it by "Γ" to indicate the boundary control on $\Gamma \subset \partial\Omega$.*

Remark 4.2. *Remark that for $\hat{n} = 1$, we immediately get the condition number $\kappa(H_k) = 1$ and we recognize the serial steepest gradient method, which has convergence rate $\left(\frac{\kappa-1}{\kappa+1}\right)^2$.*

It is difficult to pre-estimate the spectral condition number $\kappa(H_k)(\hat{n})$ (is a function of $\hat{n}$) that play an important role and contribute to the evaluation of the rate of convergence as our theoretical rate of convergence stated. We present in what follows numerical results that demonstrate the efficiency of our algorithm, Tests consider examples of well-posed and ill-posed control problem.

5. Numerical experiments

We shall present the numerical validation of our method in tow stages. In the first stage, we consider a linear algebra framework where we construct a random matrix-based quadratic cost function that we minimize using Algorithm 2. In the second stage, we consider the two optimal control problems presented in sections 3.1 and in 3.2 for the distributed- and Dirchlet boundary-control respectively. In both cases we minimize a quadratic cost function properly defined for each handled control problem.

5.1. **Linear algebra program.** This subsection treat basically the implementation of Algorithm 2. The program was implemented using the scientific programming language Scilab [26]. We consider the minimization of a quadratic form q where the matrix A is an SPD m-by-m matrix and a real vector $b \in \mathbb{R}^m \cap rank(A)$ are generated by hand (see below for their constructions). We aim at solving iteratively the linear system $Ax = b$, by minimizing

$$q(x) = \frac{1}{2}x^T Ax - x^T b. \tag{39}$$

Let us denote by $\hat{n}$ the partition number of the unknown $x \in \mathbb{R}^m$. The partition is supposed to be uniform and we assume that $\hat{n}$ divides m with a null rest.

We give in Table 1 a Scilab function that builds the vector step-length $\Theta_{\hat{n}}^k$ as stated in Eq. (4). In the practice we randomly generate an SPD sparse matrix $A = (\alpha + \gamma m) I_{\mathbb{R}^m} + R$, where $0 < \alpha < 1$, $\gamma > 1$, $I_{\mathbb{R}^m}$ is the m-by-m identity matrix and R is a symmetric m-by-m random matrix. This way the matrix A is symmetric and diagonally dominant, hence SPD. It is worthy noticing that the role of α is regularizing when rapidly vanishing eigenvalues of A are generated randomly. This technique helps us to manipulate the coercivity of the handled problem hence its spectral condition number.

For such matrix A we proceed to minimize the quadratic form defined in Eq.(39) with several $\hat{n}$-subdivisions.

The improvement quality of the algorithm against the serial case $\hat{n} = 1$ in term of iteration number is presented in Figure. 1. In fact, the left hand side of Figure. 1 presents the cost function minimization versus the iteration number of the algorithm where several choices of partition on $\hat{n}$ are carried out. In the right hand side of the Figure. 1 we give the logarithmic representation of the relative error $\frac{\|x^k - x^\star\|_2}{\|x^\star\|_2}$, where $x^\star$ is the exact solution of the linear system at hand.

5.2. **Heat optimal control program.** We discuss in this subsection the implementation results of Algorithm 3 for the optimization problems presented in section 3. Our tests deal with the 2D-heat equation on the bounded domain $\Omega = [0,1] \times [0,1]$. We consider, three types of test problems in both cases of distributed and Dirichlet controls. Tests vary according to the theoretical difficulty of the control problem [6, 3, 10]. Indeed, we vary the regularization parameter α and also change the initial and target solutions in order to handle more severe control problems as has been tested for instance in [6].

Numerical tests concern the minimization of the quadratic cost functionals $J(v)$ and $J_\Gamma(v_\Gamma)$ using Algorithm 3. It is well known that in the case α vanishes the control problem becomes

```
function [P]=Build_Hk(n,A,b,xk,dJk)
m=size(A,1); l=m/n; ii=modulo(m,n);
if ii~=0 then
    printf("Please chose an other n!");
    abort;
end
dJkn=zeros(m,n); Dk=[];
for i=1:n
  dJkn((i-1)*l+1:i*l,i)= dJk( (i-1)*l+1:i*l );
  Dk(i)=dJkn(:,i)'*(A*xk-b);
end
Hk=[,];
 for i=1:n
  for j=i:n
    Hktmp=A*dJkn(:,j);
    Hk(i,j)=dJkn(:,i)'*Hktmp;
    Hk(j,i)=Hk(i,j);
  end
end
theta=-Hk\Dk;
P = eye(m,m);
  for i=1:n
    P( (i-1)*l+1:i*l , (i-1)*l+1:i*l )=theta(i).*eye(l,l);
  end
endfunction
```

.

TABLE 1. Scilab function to build the vector step length, for the linear algebra program.

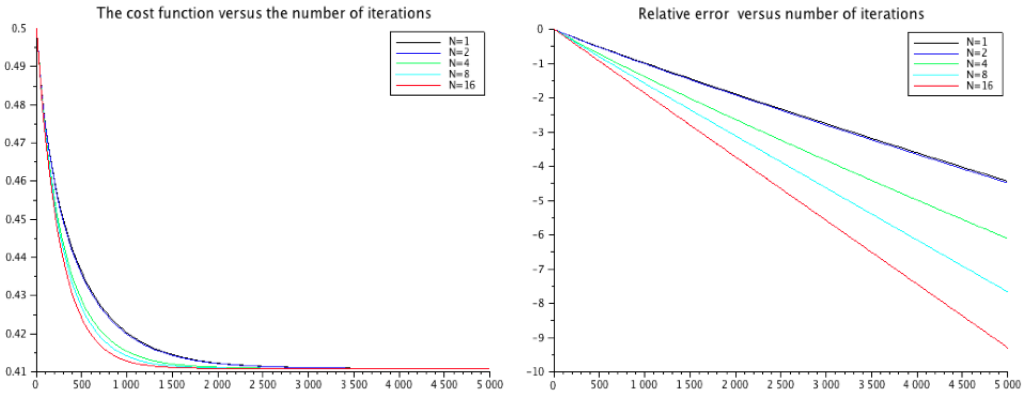

FIGURE 1. Performance in term of iteration number: Several decomposition on $\hat{n}$. Results from the linear algebra Scilab program.

an "approximated" controllability problem. Therefore the control variable tries to produce a solution that reaches as close as it "can" the target solution. With this strategy, we accentuate the ill-conditioned degree of the handled problem. We also consider an improper-posed problems for the controllability approximation, where the target solution doesn't belong to the space of the reachable solutions. No solution exists thus for the optimization problem i.e. no control exists that enables reaching the given target !

For reading conveniences and in order to emphasize the role of the parameter α on numerical tests, we tag problems that we shall consider as $\mathcal{P}_i^\alpha$ where the index i refers to the problem among $\{1,2,3,4\}$. The table below resumes all numerical test that we shall experiences

–	Minimize $J(v)$ distributed control	Minimize $J_\Gamma(v_\Gamma)$ boundary control
Moderate $\alpha = 1\times 10^{-02}$	well-posed problem corresponding data in $(\mathcal{P}_1^\alpha)$, $(\mathcal{P}_2^\alpha)$	ill-posed problem corresponding data in $(\mathcal{P}_3^\alpha)$
Vanishing $\alpha = 1\times 10^{-08}$	ill-posed problem corresponding data in $(\mathcal{P}_1^\alpha)$, $(\mathcal{P}_2^\alpha)$	sever ill-posed problem corresponding data in $(\mathcal{P}_3^\alpha)$
Solution does not exist	sever ill-posed problem corresponding data in $(\mathcal{P}_4^\alpha)$	sever ill-posed problem corresponding data in $(\mathcal{P}_4^\alpha)$

We suppose from now on that the computational domain Ω is a polygonal domain of the plane $\mathbb{R}^2$. We then introduce a triangulation $\mathcal{T}_h$ of Ω; the subscript h stands for the largest length of the edges of the tringles that constitute $\mathcal{T}_h$. The solution of the heat equation at a given time t belongs to $H^1(\Omega)$. The source terms and other variables are elements of $L^2(\Omega)$. Those infinite dimensional spaces are therefore approximated with the finite-dimensional space V_h, characterized by $\mathbb{P}_1$ the space of the polynomials of degree ≤ 1 in two variables (x_1, x_2). We have $V_h := \{u_h |\, u_h \in C^0(\overline{\Omega}), u_{h_{|K}} \in \mathbb{P}_1, \text{ for all } K \in \mathcal{T}_h\}$. In addition, Dirichlet boundary conditions (where the solution is in $H_0^1(\Omega)$ i.e. vanishing on boundary $\partial\Omega$) are taken into account via penalization of the vertices on the boundaries. The time dependence of the solution is approximated via the implicit Euler scheme. The inversion operations of matrices is performed by the UMFPAK solver. We use the trapezoidal method in order to approximate integrals defined on the time interval.

The numerical experiments were run using a parallel machine with 24 CPU's AMD with 800 MHz in a Linux environment. We code two FreeFem++ [22] scripts for the distributed and Dirichlet control. We use MPI library in order to achieve parallelism.

Tests that concern the distributed control problem are produced with control that acts on $\Omega_c \subset \Omega$, with $\Omega_c = [0, \frac{1}{3}] \times [0, \frac{1}{3}]$, whereas Dirichlet boundary control problem, the control acts on $\Gamma \subset \partial\Omega$, with $\Gamma = \{(x_1, x_2) \in \partial\Omega, |x_2 = 0\}$. The time horizon of the problem is fixed to $T = 6.4$ and the small time step is $\tau = 0.01$. In order to have a better control of the time evolution we put the diffusion coefficient $\sigma = 0.01$.

5.2.1. *First test problem: Moderate Tikhonov regularization parameter α.* We consider an optimal control problem on the heat equation. The control is considered first to be distributed and then Dirichlet. For the distributed optimal control problem we first use the functions

$$(\mathcal{P}_1^\alpha) \qquad \begin{aligned} y_0(x_1,x_2) &= \exp\big(-\gamma 2\pi\big((x_1-.7)^2+(x_2-.7)^2\big)\big) \\ y^{target}(x_1,x_2) &= \exp\big(-\gamma 2\pi\big((x_1-.3)^2+(x_2-.3)^2\big)\big), \end{aligned}$$

as initial condition and target solution respectively. The real valued γ is introduced to force the Gaussian to have support strictly included in the domain and verify the boundary conditions. The aim is to minimize the cost functional defined in Eq. (6). The decay of the cost function with respect to the iterations of our algorithm is presented in Figure. 2 on the left side, and the same results are given with respect to the computational CPU's time (in sec) on the right side. We show that the algorithm accelerates with respect to the partition number $\hat{n}$ and also preserves the accuracy of the resolution. Indeed, all tests independently of $\hat{n}$ always converge to the unique solution. This is in agreement with Theorem (4.2), which proves the convergence of the algorithm to the optimal control (unique if it exists [16]) for an arbitrary partition choice $\hat{n}$.

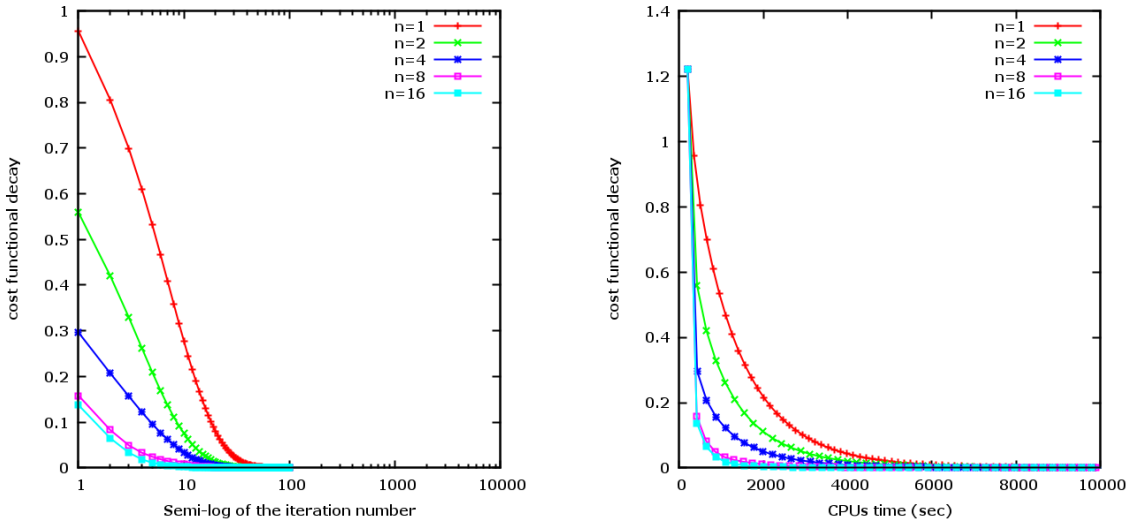

FIGURE 2. First test problem, for $\mathcal{P}_1^\alpha$: Normalized and shifted cost functional values versus iteration number (left) and versus computational time (right) for several values of $\hat{n}$ (i.e. the number of processors used).

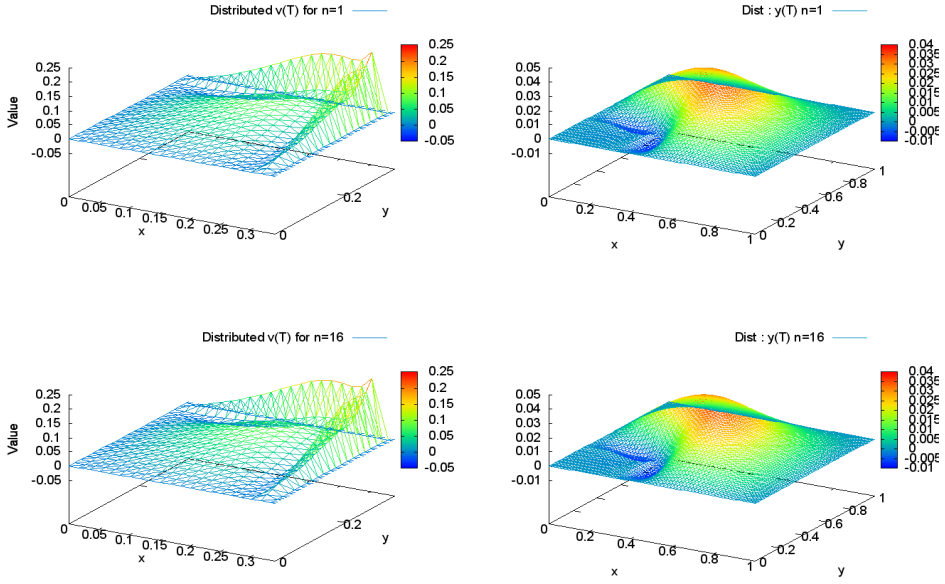

FIGURE 3. Snapshots in $\hat{n} = 1$, 16 of the distributed optimal control on the left columns and its corresponding controlled final state at time T: $y(T)$ on the right columns. The test case corresponds to the control problem $\mathcal{P}_1^\alpha$, where α is taken as $\alpha = 1 \times 10^{-02}$. Same result apply for different choice of $\hat{n}$.

We test a second problem with an a priori known solution of the heat equation. The considered problem has

$$(\mathcal{P}_2^\alpha) \qquad \begin{aligned} y_0(x_1,x_2) &= \sin(\pi x_1)\sin(\pi x_2) \\ y^{target}(x_1,x_2) &= \exp(-2\pi^2\sigma T)\sin(\pi x_1)\sin(\pi x_2), \end{aligned}$$

as initial condition and target solution respectively. Remark that the target solution is taken as a solution of the heat equation at time T. The results of this test are presented in Figure. 4, which shows the decay in values of the cost functional versus the iterations of the algorithm on the left side and versus the computational CPU's time (in sec) on the right side.

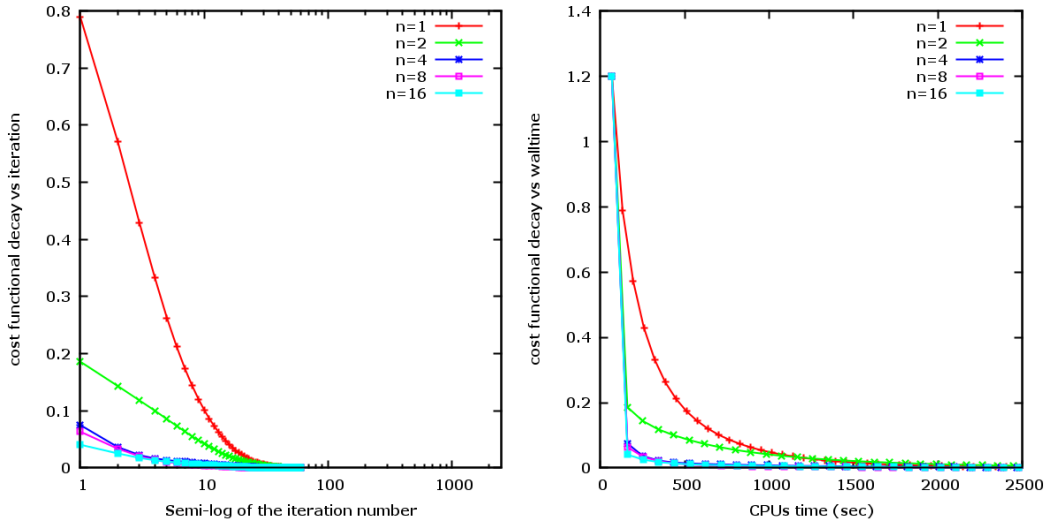

FIGURE 4. First test problem, for $\mathcal{P}_2^\alpha$: Normalized cost functional values versus computational CPU time for several values of $\hat{n}$ (i.e. the number of processors used).

We give in Figure. 3 and Figure. 5 several rows value snapshots (varying the $\hat{n}$) of the control and its corresponding controlled final solution $y(T)$. Notice the stability and the accuracy of the method with any choice of $\hat{n}$. In particular the shape of the resulting optimal control is unique as well as the controlled solution $y(T)$ doesn't depend on $\hat{n}$.

For the Dirichlet boundary control problem we choose the following functions as source term, initial condition and target solution:

$$(\mathcal{P}_3^\alpha) \qquad \begin{aligned} f(x_1,x_2,t) &= 3\pi^3\sigma\exp(2\pi^2\sigma t)(sin(\pi x_1)+sin(\pi x_2)) \\ y_0(x_1,x_2) &= \pi(sin(\pi x_1)+sin(\pi x_2)) \\ y^{target}(x_1,x_2) &= \pi\exp(2\pi^2\sigma)(sin(\pi x_1)+sin(\pi x_2)), \end{aligned}$$

respectively. Because of the ill-posed character of this problem, its optimization leads to results with hight contrast in scale. We therefore preferred to summarize the optimizations results in Table 3 instead of Figures.

Remark 5.1. *Because of the linearity and the superposition property of the heat equation, it can be shown that problems ($\mathcal{P}_2^\alpha$ and $\mathcal{P}_3^\alpha$) mentioned above are equivalent to a control problem which has null target solution.*

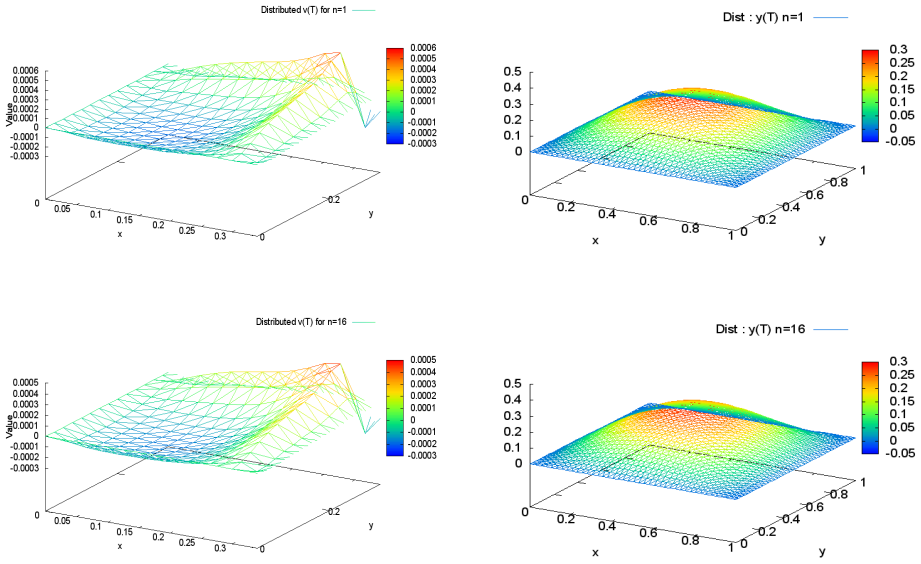

FIGURE 5. Snapshots in $\hat{n} = 1, 16$ of the distributed optimal control on the left columns and its corresponding controlled final state at time T: $y(T)$ on the right columns. The test case corresponds to the control problem $\mathcal{P}_2^\alpha$, where $\alpha = 1 \times 10^{-02}$. Same results apply for different choice of $\hat{n}$.

Test problem	Results					
$\mathcal{P}_1^\alpha$	$\alpha = 1 \times 10^{-02}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations k	100	68	63	49	27
	walltime in sec	15311.6	15352.3	14308.7	10998.2	6354.56
	$\|\mathcal{Y}^k(T) - y^{target}\|_2 / \|y^{target}\|_2$	0.472113	0.472117	0.472111	0.472104	0.472102
	$\int_{(0,T)} \|v^k\|_c^2 dt$	0.0151685	0.0151509	0.0151727	0.0152016	0.015214
$\mathcal{P}_2^\alpha$	$\alpha = 1 \times 10^{-02}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations k	60	50	45	40	35
	walltime in sec	3855.21	3726.28	4220.92	3778.13	3222.78
	$\|\mathcal{Y}^k(T) - y^{target}\|_2 / \|y^{target}\|_2$	8.26×10^{-08}	8.26×10^{-08}	8.15×10^{-08}	8.15×10^{-08}	8.14×10^{-08}
	$\int_{(0,T)} \|v^k\|_c^2 dt$	1.68×10^{-07}	1.68×10^{-07}	1.72×10^{-07}	1.72×10^{-07}	1.72×10^{-07}
$\mathcal{P}_2^\alpha$	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations k	60	50	40	30	20
	walltime in sec	3846.23	4654.34	3759.98	2835.31	1948.4
	$\|\mathcal{Y}^k(T) - y^{target}\|_2 / \|y^{target}\|_2$	3.93×10^{-08}	1.14×10^{-08}	5.87×10^{-09}	2.04×10^{-09}	1.76×10^{-09}
	$\int_{(0,T)} \|v^k\|_c^2 dt$	5.42×10^{-07}	4.13×10^{-06}	2.97×10^{-04}	3.64×10^{-03}	2.51×10^{-03}

TABLE 2. Results' summary of Algorithm 3 applied on the distributed control problems $\mathcal{P}_1^\alpha$ and $\mathcal{P}_2^\alpha$.

5.2.2. *Second test problem: vanishing Tikhonov regularization parameter α.* In this section, we are concerned with the "approximate" controllability of the heat equation, where the regularization parameter α vanishes, practically we take $\alpha = 1 \times 10^{-08}$. In this case, problems $\mathcal{P}_2^\alpha$ and $\mathcal{P}_3^\alpha$, in the continuous setting are supposed to be well posed (see for instances [9, 21]). However, may not be the case in the discretized settings; we refer for instance to [10] (and reference therein) for more details.

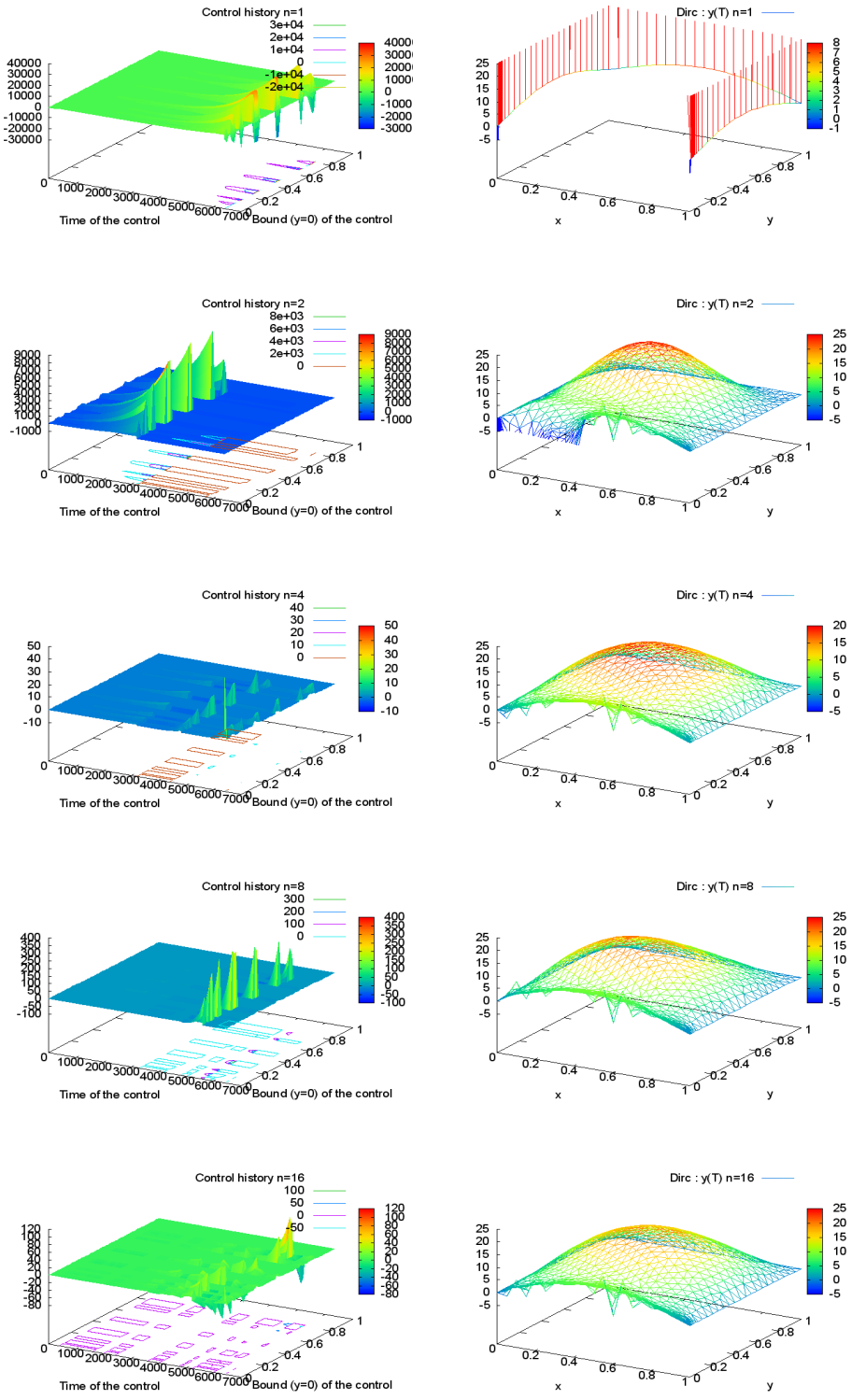

FIGURE 6. Several rows value snapshots in $\hat{n}$ of the Dirichlet optimal control on the left columns and its corresponding controlled final state at time T: $y(T)$ on the right columns. The test case corresponds to the control problem $\mathcal{P}_3^{\alpha}$, where $\alpha = 1 \times 10^{-02}$.

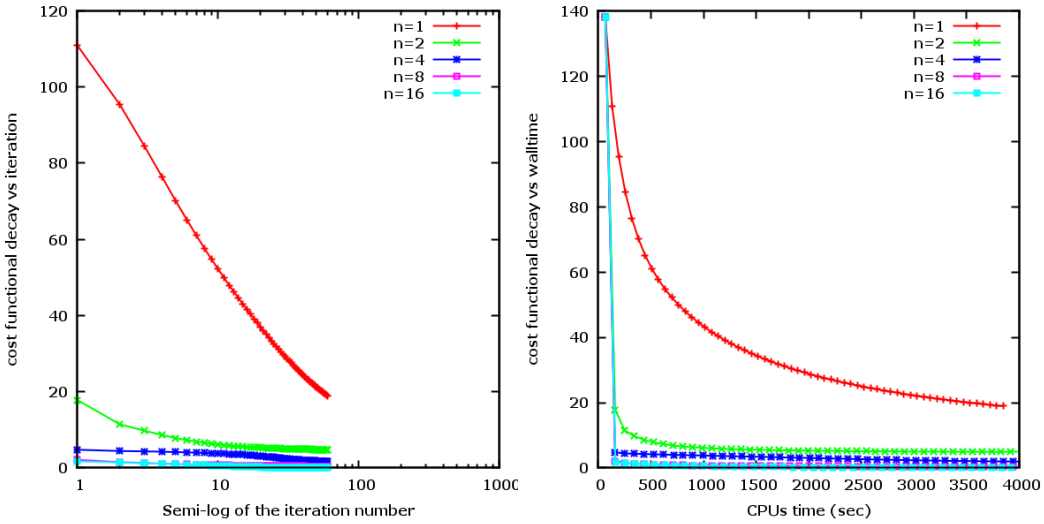

FIGURE 7. Normalized and shifted cost functional values versus computational CPU time for several values of $\hat{n}$ (i.e. the number of processors used), Distributed control problem $\mathcal{P}_2^\alpha$ whith $\alpha = 1 \times 10^{-08}$.

Test problem	Results					
$\mathcal{P}_3^\alpha$	$\alpha = 1 \times 10^{-02}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	40	40	30	18	10
	walltime in sec	12453.9	12416.1	9184.28	5570.54	3158.97
	$\|\mathcal{Y}_\Gamma(T) - y^{target}\|_2 / \|y^{target}\|_2$	$8.54 \times 10^{+06}$	0.472488	0.0538509	0.0533826	0.0534024
	$\int_{(0,T)} \|v\|_\Gamma^2 dt$	$2.79 \times 10^{+08}$	$1.96 \times 10^{+07}$	31.4193	138.675	275.08
$\mathcal{P}_3^\alpha$	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	40	40	30	27	10
	walltime in sec	1248.85	1248.97	916.232	825.791	325.16
	$\|\mathcal{Y}_\Gamma(T) - y^{target}\|_2 / \|y^{target}\|_2$	$8.85 \times 10^{+06}$	0.151086	0.0292072	0.0278316	0.0267375
	$\int_{(0,T)} \|v\|_\Gamma^2 dt$	$7.92 \times 10^{+08}$	$2.30 \times 10^{+07}$	$1.27 \times 10^{+07}$	$1.47 \times 10^{+07}$	$1.58 \times 10^{+06}$

TABLE 3. Results' summary of Algorithm 3 applied on the Dirichlet boundary control problem $\mathcal{P}_3^\alpha$.

Table 2 contains the summarized results for the convergence of the distributed control problem. On the one hand, we are interested in the error given by our algorithm for several choices of partition number $\hat{n}$. On the other hand, we give the $L^2(0,T;L^2(\Omega_c))$ of the control. We notice the improvement in the quality of the algorithm in terms of both time of execution and control energy consumption, namely the quantity $\int_{(0,T)} \|v^k\|_c^2 dt$. In fact, for the optimal control framework ($\mathcal{P}_1^\alpha$ and $\mathcal{P}_2^\alpha$ with $\alpha = 1 \times 10^{-02}$), we see that, for a fixed stopping criterion, the algorithm is faster and consume the same energy independently of $\hat{n}$. In the approximate controllability framework ($\mathcal{P}_2^\alpha$ with $\alpha = 1 \times 10^{-08}$ vanishes), we note first that the general accuracy of the controlled solution (see the error $\|\mathcal{Y}^k(T) - y^{target}\|_2 / \|y^{target}\|_2$) is improved as $\alpha = 1 \times 10^{-08}$ compered with $\alpha = 1 \times 10^{-02}$. Second, we note that the error diminishes when increasing $\hat{n}$, the energy consumption rises however. The scalability in CPU's time and number of iteration shows the enhancement of our method when it is applied (i.e. for $\hat{n} > 1$).

Table 3 contains the summarized results at the convergence of the Dircichlet boundary control problem. This problem is known in the literature for its ill-posedness, where it may be singular

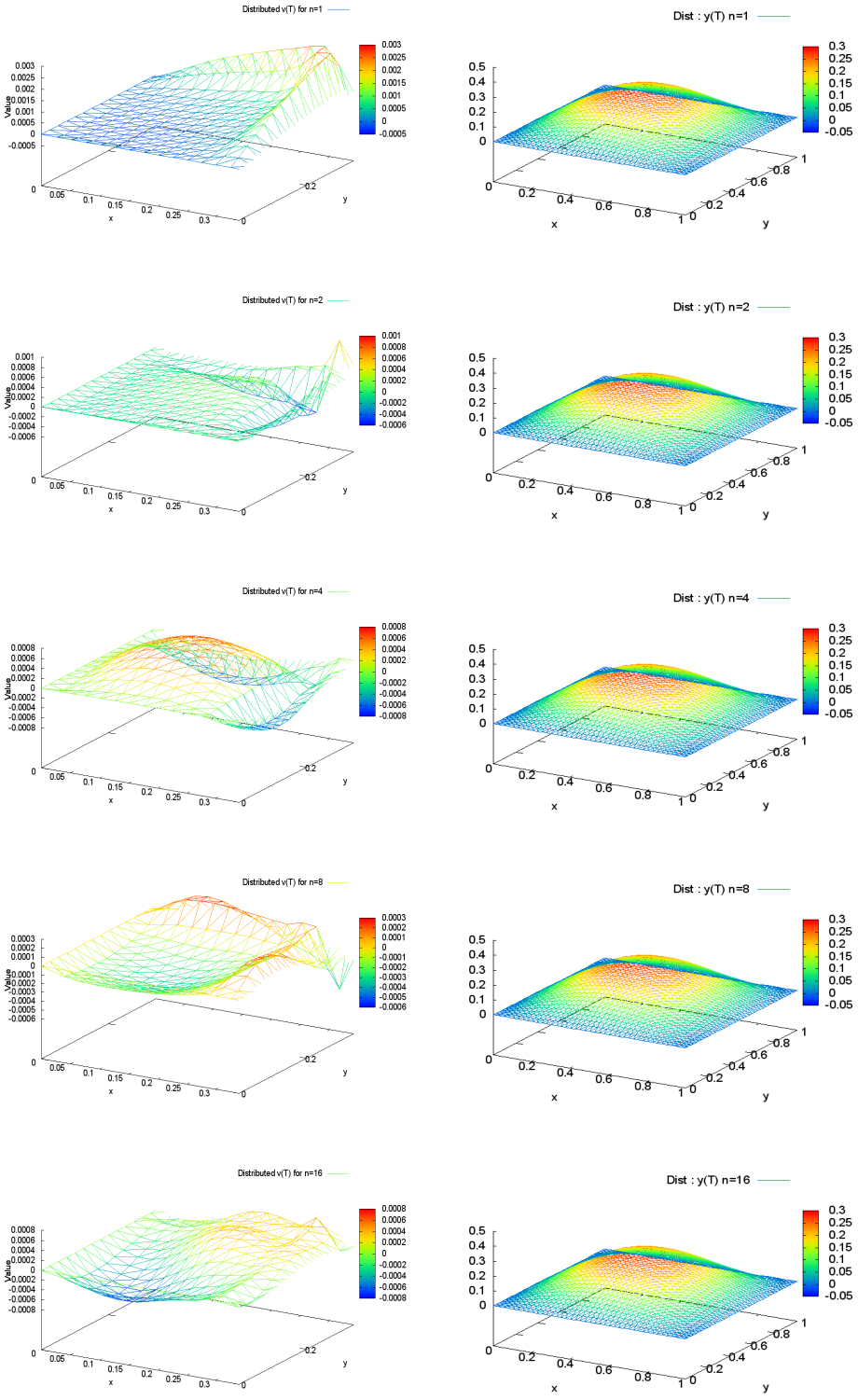

FIGURE 8. Several rows value snapshots in $\hat{n}$ of the distributed optimal control on the left columns and its corresponding controlled final state at time T: ($\mathcal{Y}(T)$ on the right columns. The test case corresponds to the control problem $\mathcal{P}_2^{\alpha}$, where $\alpha = 1 \times 10^{-08}$.

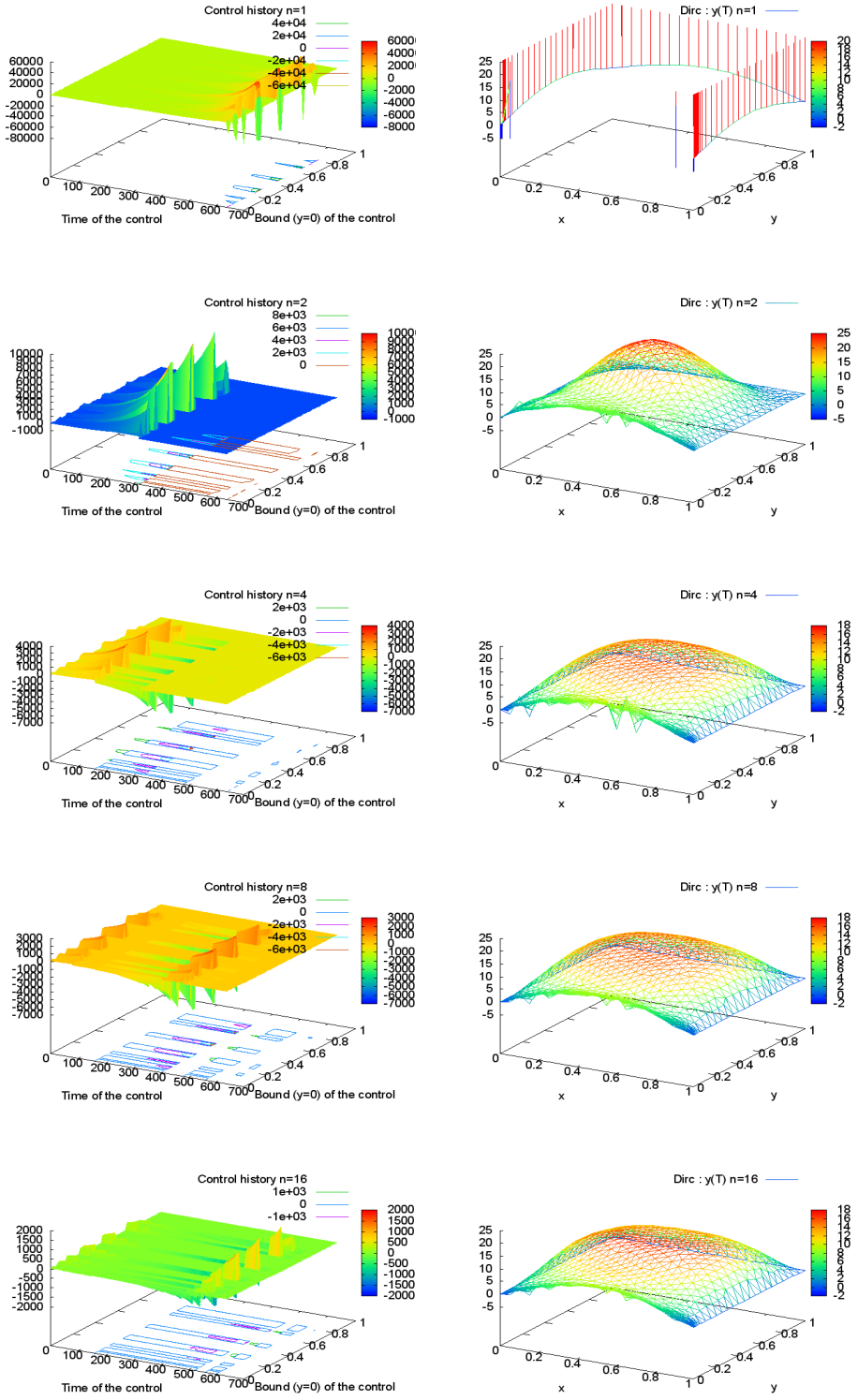

FIGURE 9. Several rows value snapshots in $\hat{n}$ of the Dirichlet optimal control on the left columns and its corresponding controlled final state at time T: $\mathcal{Y}_\Gamma(T)$ on the right columns. The test case corresponds to the control problem $\mathcal{P}_3^\alpha$, where $\alpha = 1 \times 10^{-08}$..

in several cases see [3] and references therein. In fact, it is very sensitive to noise in the data. We show in Table 3 that for a big value of the regularization parameter α our algorithm behaves already as the distributed optimal control for a vanishing α, in the sense that it consumes more control energy to produce a more accurate solution with smaller execution CPU's time. It is worth noting that the serial case $\hat{n} = 1$ fails to reach an acceptable solution, whereas the algorithm behaves well as $\hat{n}$ rises.

We give in Figure. 6 and Figure. 9 several rows value snapshots (varying $\hat{n}$) of the Dirichlet control on Γ. We present in the first column its evolution during $[0, T]$ and on the second column its corresponding controlled final solution $y(T)$ at time T; we scaled the plot of the z-range of the target solution in both Figs.6 and 9.

In each row one sees the control and its solution for a specific partition $\hat{n}$. The serial case $\hat{n} = 1$ leads to a controlled solution which doesn't have the same rank as y^{target}, whereas as $\hat{n}$ rises, we improve the behavior of the algorithm.

It is worth noting that the control is generally active only around the final horizon time T. This is very clear in Figure. 6 and Figure. 9 (see the first row i.e. case $\hat{n} = 1$). The nature of our algorithm, which is based on time domain decomposition, obliges the control to act in subintervals. Hence, the control acts more often and earlier in time (before T) and leads to a better controlled solution $y(T)$.

5.2.3. *Third test problem: Sever ill-posed problem (no solution).* In this test case, we consider a severely ill-posed problem. In fact, the target solution is piecewise Lipschitz continuous, so that it is not regular enough compared with the solution of the heat equation. This implies that in our control problem, both the distributed and the Dirchlet boundary control has no solution. The initial condition and the target solution are given by

$$(\mathcal{P}_4^{\alpha}) \qquad \begin{aligned} y_0(x_1, x_2) &= \pi(sin(\pi x_1) + sin(\pi x_2)) \\ y^{target}(x_1, x_2) &= \min\big(x_1, x_2, (1 - x_1), (1 - x_2)\big), \end{aligned}$$

respectively. A plots of the initial condition and the target solutions are given in Figure. 10.

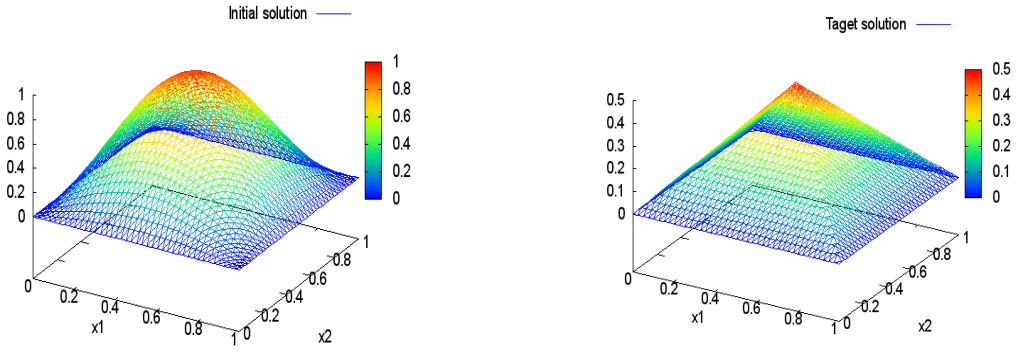

FIGURE 10. Graph of initial and target solution for both distributed and Dirichlet boundary control problem.

In Figures 11 and 12 we plot the controlled solution at time T for the distributed and Dirichlet control problems respectively. We remark that for the distributed control problem the controlled solution is smooth except in Ω_c, where the control is able to fit with the target solution.

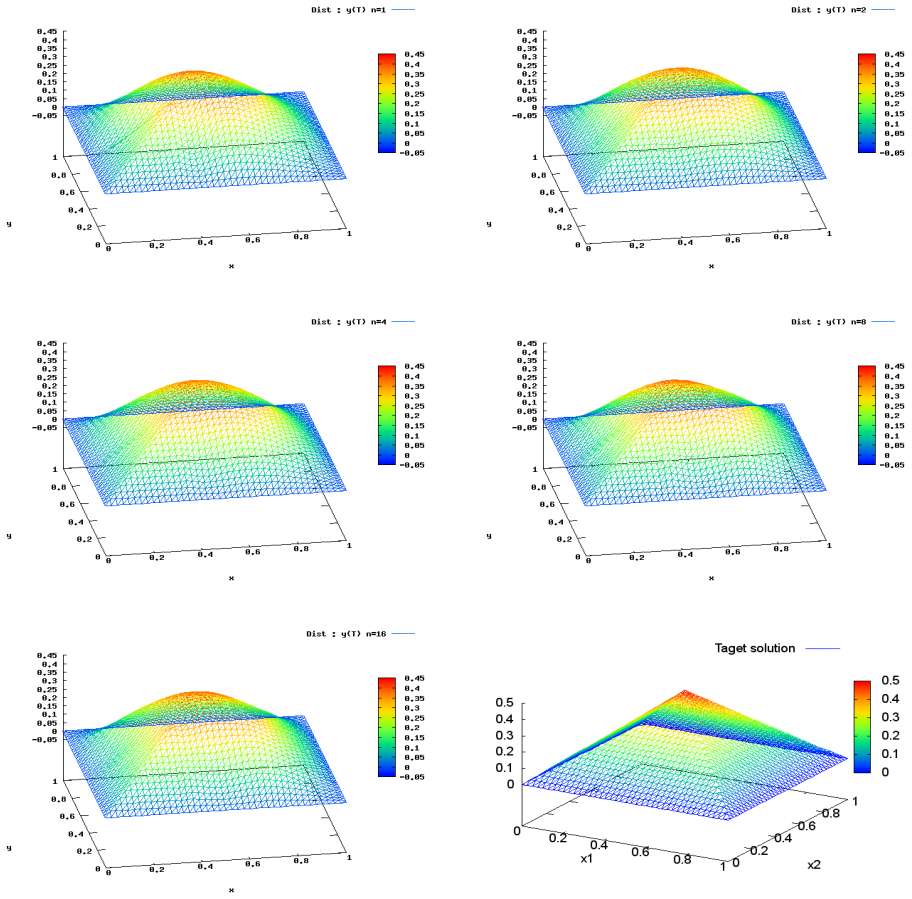

FIGURE 11. Several snapshots in $\hat{n}$ of final state at time T: $\mathcal{Y}(T)$. The test case corresponds to Distributed control sever Ill-posed problem $\mathcal{P}_4^{\alpha}$.

Test problem	Results					
Distributed control $\mathcal{P}_4^{\alpha}$	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	100	68	60	50	40
	walltime in sec	6381.43	6303.67	5548.16	4676.83	3785.97
	$\|\mathcal{Y}(T) - y^{target}\|_2 / \|y^{target}\|_2$	8.16×10^{-03}	5.3×10^{-03}	4.74×10^{-03}	3.95×10^{-03}	3.76×10^{-03}
	$\int_{(0,T)} \|v\|_c^2 dt$	0.34	3.01	52.87	52.77	2660.87
Dirichlet control $\mathcal{P}_4^{\alpha}$	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	25	25	20	4	1
	walltime in sec	848.58	655.40	655.40	146.19	62.87
	$\|\mathcal{Y}_\Gamma(T) - y^{target}\|_2 / \|y^{target}\|_2$	$2.85 \times 10^{+10}$	3055	39.3	0.2	0.067
	$\int_{(0,T)} \|v\|_\Gamma^2 dt$	$6.73 \times 10^{+08}$	$2.17 \times 10^{+07}$	141.62	17.84	26758.5

TABLE 4. Results' summary of Algorithm 3 applied on to both distributed and Dirichlet boundary control for the third test problem $\mathcal{P}_4^{\alpha}$.

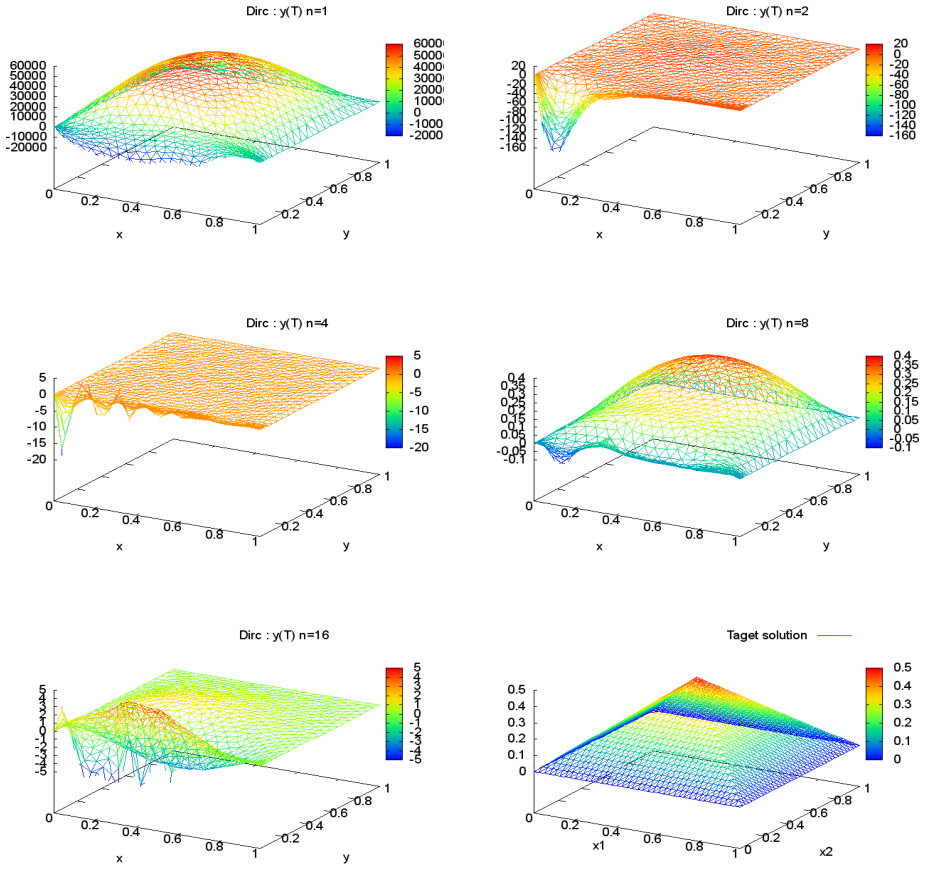

FIGURE 12. Several snapshots in $\hat{n}$ of final state at time T: $\mathcal{Y}_\Gamma(T)$. The test case corresponds to Dirichlet control sever Ill-posed problem $\mathcal{P}_4^\alpha$.

Remark 5.2. *Out of curiosity, we tested the case where the control is distributed on the whole domain. We see that the control succeeds to fit the controlled solution to the target even if it is not $\mathcal{C}^1(\Omega)$. This is impressive and shows the impact on the results of the regions where the control is distributed.*

We note the stability of the method of the distributed test case. However, the Dirichlet problem test case presents hypersensitivity. In fact, in the case of $\hat{n} = 1$ the algorithm succeeds to fit an acceptable shape of the controlled solution, although still far in the scale. We note that the time domain decomposition leads to a control which gives a good scale of the controlled solution.

In this severely ill-posed problem, we see that some partitions may fail to produce a control that fit the controlled solution to the target. There is an exemption for the case of $\hat{n} = 8$ partitions, where we have a good reconstruction of the target. The summarized results are given in Tables 4.

5.2.4. *Regularization based on the optimal choice of partition.* The next discussion concerns the kind of situation where the partition leads to multiple solutions, which is common in ill-posed

problems. In fact, we discuss a regularization procedure used as an exception handling tool to choose the best partition, giving the best solution of the handled control problem.

It is well known that ill-posed problems are very sensitive to noise, which could be present due to numerical approximation or to physical phenomena. In that case, numerical algorithm may blow-up and fail. We present several numerical tests for the Dirichlet boundary control, which is a non trivial problem numerically. The results show that in general time domain decomposition may improve the results in several cases. But scalability is not guaranteed as it is for the distributed control. We propose a regularization procedure in order to avoid the blow-up and also to guarantee the optimal choice of partition of the time domain. This procedure is based on a test of the monotony of the cost function. In fact, suppose that we possess 64 processors to run the numerical problem. Once we have assembled the Hessian H_k and the Jacobian D_k for the partition $\hat{n} = 64$, we are actually able to get for free the results of the Hessian and the Jacobian for all partitions $\hat{n}$ that divide 64. Hence, we can use the quadratic property of the cost functional in order to predict and test the value of the cost function for the next iteration without making any additional computations. The formulae is given by:

$$J(v^{k+1}) = J(v^k) - \frac{1}{2} D_k^T H_k^{-1} D_k.$$

We present in Algorithm 4 the technique that enables us to reduce in rank and compute a series of Hessians and Jacobians for any partition $\hat{n}$ that divide the available number of processors. An exemple of the applicability of these technique, on a 4-by-4 SPD matrix, is given in Appendix.

Algorithm 4: Reduce in rank of the partition $\hat{n}$

Input: $\hat{n}, H_{\hat{n}}^k, D_{\hat{n}}^k$;
$n = \hat{n}$;
$J_{n/2}^{k+1} = J_n^{k+1}$;
while $J_{n/2}^{k+1} > J_n^k$ **do**
  for $i = 0; i \leq n; i + 2$ **do**
    $\left(D_{n/2}^k\right)_i = \left(D_n^k\right)_i + \left(D_n^k\right)_{i+1}$;
    for $j = 0; j \leq n; j + 2$ **do**
      $\left(H_{n/2}^k\right)_{i,j} = \left(H_n^k\right)_j + \left(H_n^k\right)_{j+1}$;
    end
  end
  Estimation of the cost $J_{n/2}^k$;
  $n = n/2$;
end

6. Conclusion

We have presented in this article a new optimization technique to enhance the steepest descent algorithm via domain decomposition in general and we applied our new method in particular to time-parallelizing the simulation of an optimal heat control problem. We presented its performance (in CPU time and number of iterations) versus the traditional steepest descent algorithm in several and various test problems. The key idea of our method is based on a quasi-Newton technique to perform efficient real vector step-length for a set of descent directions regarding the domain decomposition. The originality of our approach consists in enabling parallel computation where its vector step-length achieves the optimal descent direction in a high dimensional space.

Convergence property of the presented method is provided. Those results are illustrated with several numerical tests using parallel resources with MPI implementation.

Appendix A. Kantorovich matrix inequality

For the sake of completeness, we give in this appendix the Matrix Kantorovich inequality, that justifies the statement of our convergence proof. Assume that $\nabla^2 J$ is symmetric positive definite with smallest and largest eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$ respectively. We give in the following the matrix version of the famous Kantorovich inequality, which reads:

Theorem A.1 (see [14] for more details)**.** *Assume that* $\sum_{n=1}^{\hat{n}} \alpha_n = 1$ *where* $\alpha_n \geq 0$ *and* $\lambda_n > 0 \quad \forall n$*; we have thus :*

$$\sum_{n=1}^{\hat{n}} \alpha_n \lambda_n \sum_{n=1}^{\hat{n}} \frac{\alpha_n}{\lambda_n} \leq \frac{(\lambda_{max} + \lambda_{min})^2}{4\lambda_{max}\lambda_{min}}.$$

By diagonalizing the symmetric positive definite operator H we obtain: $H = P\Lambda P^{-1}$, where P is orthonormal operator (i.e. $P^T = P^{-1}$). Recall Eq.(38) that we rewrite as:

$$\frac{\|\nabla J(v^k)\|^2_{\nabla^2 J} \|\nabla J(v^k)\|^2_{(\nabla^2 J)^{-1}}}{\|\nabla J(v^k)\|^4_c} \leq \frac{(\lambda_{max} + \lambda_{min})^2}{4\lambda_{max}\lambda_{min}}.$$

In order to simplify the expression, we shall use d_k instead of $\nabla J(v^k)$ so that the equation above reads:

$$\frac{d_k^T(\nabla^2 J)d_k \; d_k^T(\nabla^2 J)^{-1}d_k}{(d_k^T d_k)^2} = \frac{d_k^T P^T \Lambda P d_k}{d_k^T P^T P d_k} \frac{d_k P^T \Lambda^{-1} P d_k}{d_k^T P^T P d_k}.$$

Let us define $\mathbf{d}_k := Pd_k$, consequently the above equality becomes:

$$\frac{\mathbf{d}_k^T \Lambda \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{d}_k} \frac{\mathbf{d}_k^T \Lambda^{-1} \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{d}_k} = \sum_{n=1}^{\hat{n}} \frac{(\mathbf{d}_k)_n^2}{\mathbf{d}_k^T \mathbf{d}_k} \lambda_n \sum_{n=1}^{\hat{n}} \frac{(\mathbf{d}_k)_n^2}{\mathbf{d}_k^T \mathbf{d}_k} \frac{1}{\lambda_n}.$$

We then denote by $\alpha_n = \frac{(\mathbf{d}_k)_n^2}{\mathbf{d}_k^T \mathbf{d}_k}$ so that $\sum_{n=1}^{\hat{n}} \alpha_n = 1$, and finally:

$$\frac{d_k^T A d_k \; d_k^T A^{-1} d_k}{(d_k^T d_k)^2} = \sum_{n=1}^{\hat{n}} \alpha_n \lambda_n \sum_{n=1}^{\hat{n}} \frac{\alpha_n}{\lambda_n}.$$

□

Example 1. *Exemple 4-by-4 SPD matrix reduced in rank using the regularization procedure described in Algorithm 4. In order to illustrate the steps of Algorithm 4, we choose a simple example: a matrix 4-by-4 which we are going to reduce recursively in 2-by-2 and in 1-by-1 as follows:*

$$\begin{pmatrix} 6 & 1 & 2 & 3 \\ 1 & 8 & 2 & 4 \\ 2 & 2 & 12 & 7 \\ 3 & 4 & 7 & 16 \end{pmatrix} \mapsto \begin{pmatrix} (6 & 1) & (2 & 3) \\ (1 & 8) & (2 & 4) \\ (2 & 2) & (12 & 7) \\ (3 & 4) & (7 & 16) \end{pmatrix} \mapsto \begin{pmatrix} 7 & 5 \\ 9 & 6 \\ 4 & 19 \\ 7 & 23 \end{pmatrix} \mapsto \begin{pmatrix} 16 & 11 \\ 11 & 42 \end{pmatrix}$$

$$\begin{pmatrix} 16 & 11 \\ 11 & 42 \end{pmatrix} \mapsto \begin{pmatrix} 27 \\ 53 \end{pmatrix} \mapsto (80)$$

References

[1] Jerzy K Baksalary and Simo Puntanen. Generalized matrix versions of the cauchy-schwarz and kantorovich inequalities. *Aequationes Mathematicae*, 41(1):103–110, 1991.

[2] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[3] Faker Ben Belgacem and Sidi Mahmoud Kaber. On the Dirichlet boundary controllability of the one-dimensional heat equation: semi-analytical calculations and ill-posedness degree. *Inverse Problems*, 27(5):055012, 19, 2011.

[4] Petter Bjorstad and William Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, 2004.

[5] Richard H. Byrd, Gabriel Lopez-Calva, and Jorge Nocedal. A line search exact penalty method using steering rules. *Math. Program.*, 133(1-2, Ser. A):39–73, 2012.

[6] C. Carthel, R. Glowinski, and J.-L. Lions. On exact and approximate boundary controllabilities for the heat equation: a numerical approach. *J. Optim. Theory Appl.*, 82(3):429–484, 1994.

[7] Augustin-Louis Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Compte Rendu des Scieances de L'Académie des Sciences XXV*, S'erie A(25):536–538, October 1847.

[8] Philippe G. Ciarlet. *Introduction à l'analyse numérique matricielle et à l'optimisation*. Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master's Degree]. Masson, Paris, 1982.

[9] Jean-Michel Coron and Emmanuel Trélat. Global steady-state controllability of one-dimensional semilinear heat equations. *SIAM J. Control Optim.*, 43(2):549–569, 2004.

[10] Sylvain Ervedoza and Enrique Zuazua. The wave equation: Control and numerics. In *Control of Partial Differential Equations*, Lecture Notes in Mathematics, pages 245–339. Springer Berlin Heidelberg, 2012.

[11] David J Evans. *Preconditioning Methods: Theory and Applications*. Gordon and Breach Science Publishers, Inc., 1983.

[12] Luigi Grippo, Francesco Lampariello, and Stephano Lucidi. A nonmonotone line search technique for newton's method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.

[13] Marcus J Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[14] L. V. Kantorovich. *Functional analysis and applied mathematics*. NBS Rep. 1509. U. S. Department of Commerce National Bureau of Standards, Los Angeles, Calif., 1952. Translated by C. D. Benster.

[15] J Lions, Yvon Maday, and Gabriel Turinici. A"parareal"in time discretization of pde's. *Comptes Rendus de l'Academie des Sciences Series I Mathematics*, 332(7):661–668, 2001.

[16] J.-L. Lions. *Optimal control of systems governed by partial differential equations*. Translated from the French by S. K. Mitter. Die Grundlehren der mathematischen Wissenschaften, Band 170. Springer-Verlag, New York, 1971.

[17] Y. Maday, J. Salomon, and G. Turinici. Monotonic parareal control for quantum systems. *SIAM Journal on Numerical Analysis*, 45(6):2468–2482, 2007.

[18] Yvon Maday, Mohamed-Kamel Riahi, and Julien Salomon. Parareal in time intermediate targets methods for optimal control problems. In Kristian Bredies, Christian Clason, Karl Kunisch, and Gregory von Winckel, editors, *Control and Optimization with PDE Constraints*, volume 164 of *International Series of Numerical Mathematics*, pages 79–92. Springer Basel, 2013.

[19] Yvon Maday and Gabriel Turinici. A parareal in time procedure for the control of partial differential equations. *C. R. Math. Acad. Sci. Paris*, 335(4):387–392, 2002.

[20] Tahir Malas and Levent Gürel. Incomplete lu preconditioning with the multilevel fast multipole algorithm for electromagnetic scattering. *SIAM Journal on Scientific Computing*, 29(4):1476–1494, 2007.

[21] Sorin Micu and Enrique Zuazua. Regularity issues for the null-controllability of the linear 1-d heat equation. *Systems Control Lett.*, 60(6):406–413, 2011.

[22] Olivier Pironneau, Frédéric Hecht, and Jacques Morice. freefem++, www.freefem.org/, 2013.

[23] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations*. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York, 1999. Oxford Science Publications.

[24] Ulrich Rüde. *Mathematical and computational techniques for multilevel adaptive methods*. SIAM, 1993.

[25] Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[26] Scilab Enterprises. *Scilab: Le logiciel open source gratuit de calcul numrique*. Scilab Enterprises, Orsay, France, 2012.

[27] Andrea Toselli and Olof Widlund. *Domain decomposition methods: algorithms and theory*, volume 3. Springer, 2005.

[28] Gonglin Yuan and Zengxin Wei. The Barzilai and Borwein gradient method with nonmonotone line search for nonsmooth convex optimization problems. *Math. Model. Anal.*, 17(2):203–216, 2012.