

# Variable Reduction as a Nonlinear Preconditioning Approach for Optimization Problems

Gabriele Ciaramella<sup>[0000-0002-5877-4426]</sup> and Tommaso Vanzan<sup>[0000-0001-7554-4692]</sup>

## 1 Introduction

In this manuscript, we consider the unconstrained optimization problem

$$\min_{z \in \mathbb{R}^n} J(z), \quad (1)$$

where  $J : \mathbb{R}^n \rightarrow \mathbb{R}$  is a twice continuously differentiable cost function bounded from below and whose minimizers satisfy the nonlinear optimality conditions

$$\nabla J(z) = 0. \quad (2)$$

The direct solution of (2) using the Newton method can, in general, be hard, and may suffer from both poor initial guesses and a high computational cost. For these reasons, several algorithms (e.g., first-order methods, SQP, Quasi-Newton methods, trust-region methods) have been developed. From a Domain Decomposition (DD) community point of view, it is instead natural to study nonlinear preconditioning strategies to improve the convergence of Newton's method. This is one of the most active research areas within the community as shown by the much participated minisymposium on the topic at the last conference at KAUST and the numerous contributions to the last conference proceedings, see, e.g., [4, 11, 12, 13]. As nicely summarized in [15, §1.2], nonlinear preconditioning strategies to solve a general nonlinear system  $F(z) = 0$  can be divided into left approaches (see, e.g., [5, 10, 7]), which replace the original system with an equivalent one  $G(F(z)) = 0$  having the

---

Gabriele Ciaramella  
MOX, Dipartimento di Matematica, Politecnico di Milano, Italy, e-mail: gabriele.ciaramella@polimi.it, Member of the Indam GNCS.

Tommaso Vanzan  
Dipartimento di Scienze Matematiche, Politecnico di Torino, Italy, e-mail: tommaso.vanzan@polito.it, Member of the Indam GNCS.

same roots but that is easier to solve, and right approaches (see, e.g., [14, 6]) which instead change the variables of the system by solving  $F(G(\mathbf{y})) = 0$ , with  $z = G(\mathbf{y})$ .

The goal of this contribution is to present an alternative approach to use a right-preconditioning approach, called nonlinear elimination, within the optimization field. Specifically, we use nonlinear elimination not to precondition the optimality system (2), but to derive a reduced objective function which can then be minimized with *any* preferred optimization algorithm. The approach is promising both in terms of the theoretical analysis that can be developed and of computational efficiency.

To better frame our discussion in a DD setting, throughout this manuscript, we assume that the variable  $z$  can be suitably split into  $z = (\mathbf{x}, \mathbf{y})$ , with  $\mathbf{x} \in \mathbb{R}^{n_x}$ ,  $\mathbf{y} \in \mathbb{R}^{n_y}$  with  $n_x + n_y = n$ . We will not delve here into how this decomposition is derived: for certain optimization problems,  $z$  is naturally decomposed into subsets of variables with clear different roles (as in risk-averse PDE-constrained optimization which originally motivated this work [8, §5]), see also [15, §1.5] for an overview of spatial-, fields-, and physics-based decompositions in nonlinear preconditioning. We will however discuss that the efficacy of the methods analyzed strongly depends on a good choice for the variable partition.

## 2 Nonlinear elimination and a right preconditioned gradient descent

Given the decomposition  $z = (\mathbf{x}, \mathbf{y})$ , (2) can be reformulated as

$$\nabla_{\mathbf{x}}J(\mathbf{x}, \mathbf{y}) = 0, \quad \nabla_{\mathbf{y}}J(\mathbf{x}, \mathbf{y}) = 0. \quad (3)$$

Even though we are considering an unconstrained optimization problem, we could artificially<sup>1</sup> constrain a set of variables in terms of the others. Assuming that for every  $\mathbf{x}$  the equation  $\nabla_{\mathbf{y}}J(\mathbf{x}, \mathbf{y}) = 0$  admits a unique solution  $\mathbf{y}$  and that  $\nabla_{\mathbf{y}\mathbf{y}}J(\mathbf{x}, \mathbf{y})$  is invertible for every  $(\mathbf{x}, \mathbf{y})$ , we can nonlinearly eliminate the  $\mathbf{y}$  variable by considering the differentiable implicit map  $h : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$  such that  $\nabla_{\mathbf{y}}J(\mathbf{x}, h(\mathbf{x})) = 0$ . We are then left to solve the reduced nonlinear equation  $\tilde{F}(\mathbf{x}) := \nabla_{\mathbf{x}}J(\mathbf{x}, h(\mathbf{x})) = 0$ , and to do so we may use the Newton iteration

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (J\tilde{F}(\mathbf{x}^k))^{-1}\nabla_{\mathbf{x}}J(\mathbf{x}^k, h(\mathbf{x}^k)), \quad (4)$$

where a straight calculation using implicit differentiation shows that

$$J\tilde{F}(\mathbf{x}) = \nabla_{\mathbf{x}\mathbf{x}}J(\mathbf{x}, h(\mathbf{x})) - \nabla_{\mathbf{y}\mathbf{x}}J(\mathbf{x}, h(\mathbf{x}))\nabla_{\mathbf{y}\mathbf{y}}J(\mathbf{x}, h(\mathbf{x}))^{-1}\nabla_{\mathbf{x}\mathbf{y}}J(\mathbf{x}, h(\mathbf{x})).$$

Iteration (4) can be possibly globalized (or damped) by a line-search which would require the repeated (possibly inexact) evaluation of  $h(\cdot)$  (and thus the solution of

---

<sup>1</sup> In contrast with cases where one can often eliminate some variables using the given constraints, problem (1) is unconstrained, and the elimination arises from the optimality conditions.

the equations  $\nabla_{\mathbf{y}}J(\mathbf{x}, h(\mathbf{x})) = 0$ ). What we have just described coincides exactly with the well-known nonlinear elimination method (see, e.g. [14, 6]) applied to (3).

However, since our original task is to solve an optimization problem, we remark that the variable elimination could be performed not only on the optimality system (3), but directly on the objective function. In other words, we consider the reduced cost function  $\tilde{J} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$  defined as  $\tilde{J}(\mathbf{x}) := J(\mathbf{x}, h(\mathbf{x}))$ . While a Newton's method applied to the optimality condition of  $\tilde{J}$  is equivalent to the nonlinear elimination approach recalled above for (3)<sup>2</sup>, we are now actually free to choose our preferred algorithm to minimize  $\tilde{J}$ . In particular, we here focus on a Gradient Descent (GD) method applied to  $\tilde{J}$ , summarized by Alg. 1, and that from now on we call *right* Preconditioned Gradient Descent (PGD) method, since we are performing a change of variables that guarantees that both the cost function and its gradient are always evaluated (even within the line-search) on points that satisfy a subset of the optimality conditions in (3). This is contrast with the most popular approach to precondition GD, that is to multiply the gradient by a suitable invertible matrix which rescales the descent directions. This latter approach can be interpreted as a left preconditioning of the gradient descent method.

**Data:** Initial guess  $\mathbf{x}^0$ , tolerance Tol.

**Result:** Stationary point  $\mathbf{x}^*$ .

Set  $k = 0$ ;

**while**  $\|\nabla\tilde{J}(\mathbf{x}^k)\| > \text{Tol}$  **do**

    Compute a step length  $t$  such that  $\tilde{J}(\mathbf{x}^k - t\nabla\tilde{J}(\mathbf{x}^k)) < \tilde{J}(\mathbf{x}^k)$ ;  
    Set  $\mathbf{x}^{k+1} = \mathbf{x}^k - t\nabla\tilde{J}(\mathbf{x}^k)$ ;

**end**

**Algorithm 1:** Right preconditioned gradient descent

In the rest of this section, we study the convergence properties of Alg. 1 in a few relevant settings. As a first case study, we consider the quadratic cost function

$$J(\mathbf{z}) = \frac{1}{2}\mathbf{z}^\top A\mathbf{z} - \mathbf{b}^\top \mathbf{z} + c = \frac{1}{2}(\mathbf{x}, \mathbf{y})^\top \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} - (\mathbf{b}_1 \ \mathbf{b}_2)^\top \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} + c, \quad (5)$$

$A$  being a s.p.d. matrix. It is well known (see, e.g., [2, §1.3.2]) that for a quadratic problem there is an optimal choice for the step length (available in closed formula so that no line-search is needed) such that GD generates iterates satisfying

$$\|\mathbf{x}^k - \mathbf{x}^*\|_2 \leq \sqrt{\kappa_2(A)} \left( \frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^k \|\mathbf{x}^0 - \mathbf{x}^*\|_2. \quad (6)$$

The convergence rate then depends strongly on the conditioning of  $A$ . If we assume that  $A_{11}$  is well conditioned while  $A_{22}$  is ill conditioned, we may think about *eliminating* the  $\mathbf{y}$  variables, deriving a reduced optimization problem only with respect

<sup>2</sup> Notice that  $\nabla\tilde{J}(\mathbf{x}) = \nabla_{\mathbf{x}}J(\mathbf{x}, h(\mathbf{x})) + \nabla_{\mathbf{y}}J(\mathbf{x}, h(\mathbf{x}))h'(\mathbf{x}) = \nabla_{\mathbf{x}}J(\mathbf{x}, h(\mathbf{x})) = \tilde{\mathbf{F}}(\mathbf{x})$ , since by definition  $h$  is such that  $\nabla_{\mathbf{y}}J(\mathbf{x}, h(\mathbf{x})) = 0$ .

to the  $\mathbf{x}$  variables, and hope to recover a fast convergence of GD applied to the reduced optimization problem. To do so, we focus on the optimality condition of (5), corresponding to the linear system  $Az = \mathbf{b}$  from which, using static condensation, we may express  $\mathbf{y} = h(\mathbf{x}) = -A_{22}^{-1}A_{21}\mathbf{x} + A_{22}^{-1}\mathbf{b}_2$ . Inserting  $\mathbf{y} = h(\mathbf{x})$  into  $J(\mathbf{x}, \mathbf{y})$  we obtain the reduced cost function

$$\tilde{J}(\mathbf{x}) := \frac{1}{2}\mathbf{x}^\top S\mathbf{x} + \tilde{\mathbf{b}}^\top \mathbf{x} + \tilde{\mathbf{c}}, \quad (7)$$

where  $S := A_{11} - A_{12}A_{22}^{-1}A_{21}$  is the Schur complement and  $\tilde{\mathbf{b}}, \tilde{\mathbf{c}}$  are suitable vectors. Since the reduced cost function  $\tilde{J}$  is still quadratic, the iterates of the right preconditioned GD (with the optimal step length) satisfy

$$\|\mathbf{x}^k - \mathbf{x}^*\|_2 \leq \sqrt{\kappa_2(S)} \left( \frac{\kappa_2(S) - 1}{\kappa_2(S) + 1} \right)^k \|\mathbf{x}^0 - \mathbf{x}^*\|_2. \quad (8)$$

Observing that

$$\begin{aligned} \lambda_{\max}(S) &:= \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top S\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \leq \max_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top A_{11}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \leq \max_{\mathbf{z}=(\mathbf{x}, \mathbf{y}) \neq 0} \frac{\mathbf{z}^\top A\mathbf{z}}{\mathbf{z}^\top \mathbf{z}}, \\ \lambda_{\min}(A) &:= \min_{\mathbf{z}=(\mathbf{x}, \mathbf{y}) \neq 0} \frac{\mathbf{z}^\top A\mathbf{z}}{\mathbf{z}^\top \mathbf{z}} \leq \min_{\mathbf{x}: \mathbf{z}=(\mathbf{x}, -A_{22}^{-1}A_{21}\mathbf{x})} \frac{\mathbf{x}^\top S\mathbf{x}}{\mathbf{x}^\top \mathbf{x} + \mathbf{x}^\top A_{12}A_{22}^{-2}A_{21}\mathbf{x}} \leq \lambda_{\min}(S), \end{aligned} \quad (9)$$

we conclude that  $\kappa_2(S) \leq \kappa_2(A)$ , and we have proved the following proposition.

**Proposition 1** *For an unconstrained quadratic optimization problem, the right PGD method always has a better asymptotic convergence rate than the standard GD method.*

While Proposition 1 guarantees that the right PGD converges asymptotically better than GD applied to the original problem *regardless* of the variable decomposition, the latter still plays a key-role in determining the actual speed-up. To see this, we consider a matrix  $A$  where  $A_{11}$  and  $A_{22}$  are similar through random orthogonal matrices to diagonal matrices with equispaced values between 10 and 1, and  $10^3$  and 1 respectively, and  $A_{12} = A_{21}^\top$  is a small random perturbation such that  $A$  is still s.p.d.. The matrix sizes are  $n_x = 40$  and  $n_y = 60$ . On the left panel of Figure 1, we show the convergence of GD and of PGD where we fully eliminate the variable  $\mathbf{y}$ . It is evident that ill conditioning due to the  $\mathbf{y}$  variable has completely vanished (this is confirmed by  $\kappa_2(S)$  which turns equal to  $\kappa_2(A_{11})$ ). Standard GD applied to  $J(\mathbf{x}, \mathbf{y})$  requires 3004 iterations and 0.12 seconds to reach a tolerance of  $10^{-6}$  on the relative gradient norm. The PGD requires only 56 iterations and 0.03 seconds, thus it leads to a speed up of a factor 4. Note that the  $S$  is not assembled, but handled in a matrix-free way for a fair comparison, so that each evaluation of  $h$  requires a linear solve. On the center panel instead, we only eliminate the last  $n_r = 50$  variables of  $\mathbf{y}$ . In this case, even though the theoretical bound for PGD is slightly better, there is no actual gain from the variable elimination: PGD needs now 2219 iterations, still less than GD, but the computational time rises up to 0.87 seconds. As a general rule,

$h$  should satisfy the same properties of a linear preconditioner: it should be cheap to compute (even more important than in the linear case since  $h$  is evaluated also during the line-search), and lead to a much better conditioned reduced problem. This is heuristically achieved by eliminating all variables associated to small eigenvalues (i.e. flat valleys) of  $\nabla^2 J$  evaluated at the optimal value  $z^*$ .

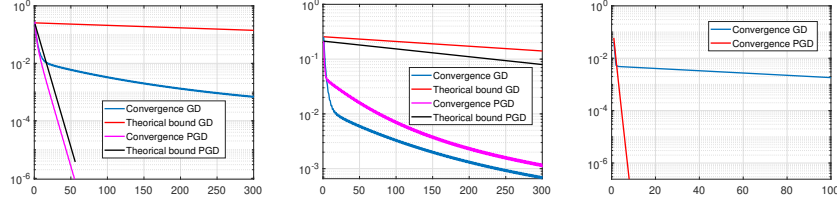


Fig. 1: Comparison between the convergence of GD and PGD for a quadratic problem.  $\kappa_2(A) = 1.001 \cdot 10^3$ ,  $\kappa_2(S) = 10$  (left),  $\kappa_2(S) = 6.08 \cdot 10^2$  (center). Right panel convergence of GD and PGD for the strongly convex optimization problem (11) with  $n = 100$  and Armijo backtracking.

We now remark that Proposition 1 can be readily extended to strongly convex cost functions that have Lipschitz continuous gradients ([16, Chapter 1]), namely functions that for certain  $\mu, L \in \mathbb{R}$  satisfy

$$\begin{aligned} J(z') &\geq J(z) + \langle \nabla J(z), z' - z \rangle + \frac{\mu}{2} \|z - z'\|_2^2, \quad \forall z, z' \in \mathbb{R}^n, \\ \|\nabla J(z) - \nabla J(z')\|_2 &\leq L \|z - z'\|_2, \quad \forall z, z' \in \mathbb{R}^n. \end{aligned} \quad (10)$$

For such functions, it is possible to prove ([16, Theorem 2.1.15]) that (6) holds with  $\kappa_2(A)$  replaced by  $\frac{L}{\mu}$ . Since then 1) for twice continuously differentiable functions  $\mu$  and  $L$  can be taken equal to the minimum and maximum eigenvalues of  $\nabla^2 J(z)$  over all  $z \in \mathbb{R}^n$ , 2)  $\nabla_{xx}^2 \tilde{J}(x)$  corresponds to the Schur complement of  $\nabla^2 J(z)$  for all  $z = (x, h(x))$  and thus the strongly convex parameters  $\tilde{\mu}, \tilde{L}$  of  $\tilde{J}$  satisfy (repeating the arguments of (9) for all  $x$ )  $\mu \leq \tilde{\mu}$  and  $\tilde{L} \leq L$ , we conclude that the PGD applied to  $\tilde{J}(x)$  has a better asymptotic convergence rate than standard GD applied to  $J(z)$ .

To conclude this section, we present on the right of Fig. (1) a numerical study of the convergence of GD and PGD, with Armijo backtracking, for the strongly convex function

$$J(z) := \text{Log} \left( \sum_{i=1}^n a_i e^{b_i x_i} \right) + \frac{1}{2} z^\top D z, \quad (11)$$

where  $n = 10^3$ ,  $n_{el} = 20$ ,  $a_i = i \forall i$ ,  $b_i = 10$ ,  $1 \leq i \leq n_{el}$ ,  $b_i = 1$ ,  $i > n_{el}$ , and  $D$  is a diagonal matrix, with the first  $n_{el}$  entries equal to  $10^{-4}$  and the remaining ones equal to  $10^{-2}$ . The function is ill conditioned with respect to the first  $n_{el}$  variables. By nonlinear eliminating them, the improvement of PGD over GD is impressive: GD requires 693 iterations and 3.74 seconds, while PGD needs 9 iterations and

0.25 seconds with a speed up of order 15. Notice that in this setting an Armijo backtracking line search is employed which, compared to the quadratic case, leads to further evaluations of the implicit map  $h$  that is computed with an inner Newton method.

### 3 Inexact variable reduction

The gain due to the reduced number of PGD iterations may be null when the evaluation of  $h(\cdot)$  is expensive. In such cases, one could consider an inexact variable reduction in which the set of nonlinear equations  $\nabla_{\mathbf{y}}J(\mathbf{x}, \mathbf{y}) = 0$  are solved approximately with an iterative scheme. An example is the iterative procedure

$$\begin{aligned} \widehat{h}_N(\mathbf{x}; \mathbf{y}^0) &:= \mathbf{y}^N, \\ \mathbf{y}^{\ell+1} &= \mathbf{y}^{\ell} - t_{\ell} D_{\ell}(\mathbf{x}, \mathbf{y}^{\ell}) \nabla_{\mathbf{y}}J(\mathbf{x}, \mathbf{y}^{\ell}), \text{ for } \ell = 0, \dots, N-1, \end{aligned} \quad (12)$$

which performs  $N$  steps of a (left) preconditioned GD (possibly Newton steps if  $D_{\ell}(\mathbf{x}, \mathbf{y}^{\ell}) = \left( \nabla_{\mathbf{y}\mathbf{y}}^2 J(\mathbf{x}, \mathbf{y}^{\ell}) \right)^{-1}$ ) on the optimization problem  $\min_{\mathbf{y}} J(\mathbf{x}, \mathbf{y})$  starting from  $\mathbf{y}^0$  with  $\mathbf{x}$  fixed. Once a new iterate  $\mathbf{x}^{k+1}$  is computed, one can update the initial guess of (12) by setting  $\mathbf{y}^0 \leftarrow \widehat{h}_N(\mathbf{x}^{k+1}; \mathbf{y}^0)$ .

To preserve consistency between the full and reduced optimization problems, the inexact elimination must satisfy  $\widehat{h}_N(\mathbf{x}^*; \mathbf{y}^*) = \mathbf{y}^*$ , that is, the iterative procedure, when starts from the optimal  $\mathbf{y}^*$  associates to  $\mathbf{x}^*$  again  $\mathbf{y}^*$ . This is trivially satisfied by (12) since  $\nabla_{\mathbf{y}}J(\mathbf{x}^*, \mathbf{y}^*) = 0$ . Incidentally, we emphasize that a special choice is  $N = 0$  leading to the constant map  $\widehat{h}_0(\mathbf{x}; \mathbf{y}_0) = \mathbf{y}_0$ . We then have

$$\min_{\mathbf{x}} \widetilde{J}(\mathbf{x}) = \min_{\mathbf{x}} J(\mathbf{x}, \widehat{h}_0(\mathbf{x}; \mathbf{y}_0)) = \min_{\mathbf{x}} J(\mathbf{x}, \mathbf{y}_0),$$

and hence the minimization of  $\widetilde{J}$  is equivalent to the minimization of  $J$  with the variable  $\mathbf{y}$  frozen. To recover convergence to  $(\mathbf{x}^*, \mathbf{y}^*)$ , one may alternate the minimization of  $J$  with respect to  $\mathbf{x}$  and  $\mathbf{y}$ ,

$$\mathbf{x}^{k+1} = \operatorname{argmin}_{\mathbf{x}} J(\mathbf{x}, \mathbf{y}^k) \quad \text{and} \quad \mathbf{y}^{k+1} = \operatorname{argmin}_{\mathbf{y}} J(\mathbf{x}^{k+1}; \mathbf{y}), \quad (13)$$

recovering the popular alternating minimization methods ([2, 3, 1]), that can thus be seen as particular instances of a reduced variable approach by choosing a specific inexact function  $\widehat{h}(\cdot)$ .

Notice further that an inexact evaluation of  $h$  complicates the computation of the gradient of the  $\widetilde{J}$  since

$$\nabla \widetilde{J}(\mathbf{x}^k) = \nabla_{\mathbf{x}} J(\mathbf{x}^k, \widehat{h}_N(\mathbf{x}^k, \mathbf{y}^0)) + \nabla_{\mathbf{y}} J(\mathbf{x}^k, \widehat{h}_N(\mathbf{x}^k, \mathbf{y}^0)) \widehat{h}'_N(\mathbf{x}^k, \mathbf{y}^0),$$

where the second term now does not cancel due to the inexactness of  $\widehat{h}_N$ , and  $\widehat{h}'_N$  may involve high order derivatives of  $J$ . As a consequence, in our numerical implementation we perform inexact elimination using a variant of (12) consisting of an inexact Newton method [9]. The iterative procedure is stopped when  $\|\nabla_{\mathbf{y}} J(\mathbf{x}, \mathbf{y})\| < \text{Tol}$ . The number of inner iteration depends then on the current tolerance, i.e.,  $N = N(\text{Tol})$ . The initial tolerance is  $10^{-3}$  and after each gradient step on  $\widetilde{J}(\mathbf{x})$ , Tol is decreased by a factor  $\rho := 0.5$ . We then use the inexact gradient  $\nabla \widetilde{J}(\mathbf{x}^k) \approx \nabla_{\mathbf{x}} J(\mathbf{x}^k, \widehat{h}_N(\mathbf{x}^k, \mathbf{y}^{0,k}))$  as a descent direction, whose inexactness though is controlled by that of  $\widehat{h}_N(\cdot)$  and vanishes in the limit for  $k \rightarrow \infty$ . We briefly mention that we have also implemented the inexact elimination (12) using  $N$ -fixed steps of gradient descent, and used the full gradient  $\nabla J$  as descent direction. The results though were not satisfactory compared with the former procedure.

Table 1 reports the number of iterations and computational times of GD and PGD with both exact and inexact elimination to minimize (11) for increasing values of  $n_{el}$ , that is the number of variables that are responsible for the ill-conditioning. While PGD with exact elimination becomes inefficient in terms of computational times as  $n_{el}$  grows, PGD with inexact elimination results very robust both in terms of iterations and computational time, and significantly outperforms GD in all test cases.

| $n_{el}$ | 10         | 50         | 200       | 400        |
|----------|------------|------------|-----------|------------|
| GD       | 713 (3.73) | 657 (3.73) | 558 (296) | 487 (2.52) |
| PGD-Ex   | 9 (0.30)   | 9 (0.30)   | 9 (0.67)  | 10 (3.19)  |
| PGD-In   | 9 (0.25)   | 9 (0.29)   | 9 (0.29)  | 10 (0.30)  |

Table 1: Number of iterations and computational time in seconds for a standard GD, a right PGD with exact elimination and a right PGD with inexact elimination to minimize (11) up to a tolerance of  $10^{-6}$  on the relative gradient norm.

## 4 Conclusions

In this contribution, we argued that nonlinear elimination can be used to reduce the number of variables in optimization problems, resulting in reduced objective functions that can potentially be easier to minimize with classical optimization algorithms. Future efforts will focus on analyzing theoretically the convergence of inexact elimination procedures, and the connections with the popular alternating minimization methods. It is also relevant to develop general criteria to identify efficient decomposition of the optimization variables. Concerning computational aspects, we plan to investigate our approach in more realistic problems, such those appearing in risk-adverse PDE-constrained optimization [8].

## Acknowledgements

The two authors have been partially supported by the INdAM-GNCS project *GNCS 2024 - CUP E53C23001670001*

## References

1. Beck, A., Tretushvili, L.: On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization* **23** (2023)
2. Bertsekas, D.P.: *Nonlinear Programming*. Athena Scientific (2016)
3. Bezdek, J.C., Hathaway, R.J.: Some notes on alternating optimization. In: *Advances in Soft Computing—AFSS 2002*. Springer (2002)
4. Brenner, K.: On global and monotone convergence of the preconditioned Newton’s method for some mildly nonlinear systems. In: *International Conference on Domain Decomposition Methods*, pp. 85–92. Springer (2022)
5. Cai, X.C., Keyes, D.E.: Nonlinearly preconditioned inexact Newton algorithms. *SIAM Journal on Scientific Computing* **24**(1), 183–200 (2002)
6. Cai, X.C., Li, X.: Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity. *SIAM journal on scientific computing* **33**(2), 746–762 (2011)
7. Chauqui, F., Gander, M.J., Kumbhar, P.M., Vanzan, T.: Linear and nonlinear substructured restricted additive Schwarz iterations and preconditioning. *Numerical Algorithms* **91**(1), 81–107 (2022)
8. Ciaramella, G., Nobile, F., Vanzan, T.: A multigrid solver for PDE-constrained optimization with uncertain inputs. *Journal of Scientific Computing* **101**(1), 13 (2024)
9. Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact Newton methods. *SIAM Journal on Numerical Analysis* **19** (1982)
10. Dolean, V., Gander, M.J., Kheriji, W., Kwok, F., Masson, R.: Nonlinear preconditioning: How to use a nonlinear Schwarz method to precondition Newton’s method. *SIAM Journal on Scientific Computing* **38**(6), A3357–A3380 (2016)
11. Kohler, S., Rheinbach, O.: Composing two different nonlinear FETI–DP methods. In: *International Conference on Domain Decomposition Methods*, pp. 479–486. Springer (2022)
12. Kothari, H.: Nonlinear Schwarz preconditioning for quasi-Newton methods. In: *International Conference on Domain Decomposition Methods*, pp. 311–318. Springer (2022)
13. Kothari, H., Kopanicáková, A., Krause, R.: Nonlinear Schwarz preconditioning for nonlinear optimization problems with bound constraints. In: *International Conference on Domain Decomposition Methods*, pp. 319–326. Springer (2022)
14. Lanzkron, P.J., Rose, D.J., Wilkes, J.T.: An analysis of approximate nonlinear elimination. *SIAM Journal on Scientific Computing* **17**(2), 538–559 (1996)
15. Liu, L., Gao, W., Yu, H., Keyes, D.E.: Overlapping multiplicative Schwarz preconditioning for linear and nonlinear systems. *Journal of Computational Physics* **496**, 112548 (2024)
16. Nesterov, Y.: *Lectures on Convex Optimization*. Springer (2018)