ALDISCR: an algorithm for infinite-dimensional constrained optimization (Rapport de recherche 2008/14)

Dujol, Romain; Sartenaer, Annick

Publication date:
2008

Document Version
Early version, also known as pre-print

Link to publication

Citation for pulished version (HARVARD):
Dujol, R & Sartenaer, A 2008, *ALDISCR: an algorithm for infinite-dimensional constrained optimization (Rapport de recherche 2008/14)*. FUNDP, Faculté des Sciences. Département de Mathématique., Namur.

`ALDISCR`: an algorithm for infinite-dimensional constrained optimization

Romain DUJOL[*] Annick SARTENAER[*]

August 28, 2008

Abstract

This document is a report on the achievements so far concerning the implementation of the `ALDISCR` algorithm designed by SACHS and SARTENAER [3]. This algorithm aims at solving constrained optimization problems in an infinite-dimensional setting. It may be implementable directly but still be a bit rigid: we propose a few practicalities to soften the process. The current implementation is tested on a few examples and conclusions so far are given.

1 Introduction

As introduced in the abstract, we consider the framework developed in [3], namely an equality-constrained optimization problem:

$$\min f(x) \quad \text{s.t.} \quad c(x) = 0 \tag{1}$$

where $f : X \to \mathbf{R}$ and $c : X \to Y$ with X and Y as Hilbert spaces.

The algorithm `ALDISCR` is an extension of the classical augmented Lagrangian algorithm (see bibliography of [3] for detailed references on the topic). We assume that we can discretize (1):

$$\min f_n(x) \quad \text{s.t.} \quad c_n(x) = 0 \tag{2}$$

where $f_n : X_n \to \mathbf{R}$ and $c_n : X_n \to Y_n$ with X_n and Y_n as finite-dimensional spaces.

All considered functions are at least twice continuously Fréchet-differentiable. The discretization scheme is nested: $(X_n)_n$ and $(Y_n)_n$ are both non-decreasing (in

---

[*] University of Namur, Belgium

the sense of set inclusion) sequences of finite-dimensional subspaces of X and Y respectively. Thus the structure considered in these euclidian subspaces is the one induced by the Hilbert space: $\langle \cdot, \cdot \rangle_{X_n} = \langle \cdot, \cdot \rangle_X$ and $\langle \cdot, \cdot \rangle_{Y_n} = \langle \cdot, \cdot \rangle_Y$.

We finally make some convergence hypothesis by assuming the existence of an integer n^* such that:

(a) $\forall x \in X_{n^*}$, $(\|c_n(x) - c(x)\|_Y)_{n \geq n^*} \xrightarrow[n \to \infty]{} 0$

(b) $\forall x \in X_{n^*}, \forall \lambda \in Y_{n^*}, \forall \mu > 0$, $(\|\nabla_x \Phi_n(x, \lambda, \mu) - \nabla_x \Phi(x, \lambda, \mu)\|_X)_{n \geq n^*} \xrightarrow[n \to \infty]{} 0$ where $\Phi(x, \lambda, \mu) \overset{\Delta}{=} f(x) + \langle \lambda, c(x) \rangle_X + \|c(x)\|_Y^2/(2\mu)$ and $\Phi_n(x, \lambda, \mu) \overset{\Delta}{=} f_n(x) + \langle \lambda, c_n(x) \rangle_X + \|c_n(x)\|_Y^2/(2\mu)$ are the augmented Lagrangians associated to problems (1) and (2) respectively.

We now give the algorithm as defined in [3]. In the following, $\varepsilon_{c,n}(x)$ and $\varepsilon_{\Phi,n}(x, \lambda, \mu)$ are upper bounds for $\|c_n(x) - c(x)\|_Y$ and $\|\nabla_x \Phi_n(x, \lambda, \mu) - \nabla_x \Phi(x, \lambda, \mu)\|_X$ respectively. We will assume that both $\varepsilon_{c,n}(x)$ and $\varepsilon_{\Phi,n}(x, \lambda, \mu)$ converge to 0 for fixed x, λ and $\mu > 0$.

Algorithm `ALDISCR`

Step 0: Initialization We are given an initial discretization level n_0, a initial Lagrange multiplier estimate $\lambda_0 \in Y_{n_0}$ and a penalty parameter $\mu_0 \in]0, 1[$.

We choose some constants ω_*, η_*, τ, α, α_η and β_η all in $]0, 1[$.

Set $k = 0$, $\omega_0 = \mu_0$ and $\eta_0 = \mu_0^{\alpha_\eta}$.

Step 1: Inner iteration Find $x_k \in X_{n_k}$ such that $\|\nabla_x \Phi_{n_k}(x_k, \lambda_k, \mu_k)\|_X \leq \omega_k/2$

Step 2: Test for convergence If $\omega_k \leq \omega_*$, $\|c_{n_k}(x_k)\|_Y \leq \eta_*/2$ and $\varepsilon_{c,n_k}(x_k) \leq \eta_*/2$, stop

Step 3: Updates If $\|c_{n_k}(x_k)\|_Y \leq \eta_k$, execute Step 3a. Otherwise, execute Step 3b.

Step 3a: Update Lagrange mutliplier Set $\lambda_{k+1} = \lambda_k + c_{n_k}(x_k)/\mu_k$, $\mu_{k+1} = \mu_k$, $\omega_{k+1} = \omega_k \mu_{k+1}$ and $\eta_{k+1} = \eta_k \mu_{k+1}^{\beta_\eta}$

Step 3b: Update penalty parameter Set $\lambda_{k+1} = \lambda_k$, $\mu_{k+1} = \tau \mu_k$, $\omega_{k+1} = \mu_{k+1}$, $\eta_{k+1} = \mu_{k+1}^{\alpha_\eta}$

Step 4: Refinement Choose $n_k \geq n_{k+1}$ such that $\varepsilon_{c,n_{k+1}}(x_k) < \min\{\alpha \eta_{k+1}, \mu_{k+1} \omega_{k+1}\}$ and $\varepsilon_{\Phi,n_{k+1}}(x_k, \lambda_{k+1}, \mu_{k+1}) \leq \omega_{k+1}/2$.

Increment k by one and go to Step 1.

Apparently, we should be able to implement this algorithm as is, provided we can compute both X and Y inner products and norms (at least for elements of X_n and Y_n) and we know how an element of Y_n is injected in $Y_{n'}$ for $n' \geq n$.

Nevertheless, some issues may arise when looking more carefully. To this end, some practical additions to the algorithm will be addressed in Section 2. Finally, we will consider some examples in Section 3 and give the current conclusions in Section 4.

2 Practical additions to `ALDISCR`

2.1 Convergence condition (Step 2)

First, we can replace the convergence condition (Step 2):

$$\|c_{n_k}(x_k)\|_Y \leq \eta_*/2 \text{ and } \varepsilon_{c,n_k}(x_k) \leq \eta_*/2 \tag{3}$$

by the following weaker condition:

$$\|c_{n_k}(x_k)\|_Y + \varepsilon_{c,n_k}(x_k) \leq \eta_* \tag{4}$$

since the condition we aim at is $\|c(x_k)\|_Y \leq \eta_*$ and we always have $\|c(x_k)\|_Y \leq \|c_{n_k}(x_k)\|_Y + \varepsilon_{c,n_k}(x_k)$.

2.2 Using finiteness within infiniteness

All conditions are expressed within the structure of the initial problem (1), that is with $\|\cdot\|_X$ and $\|\cdot\|_Y$. We would like to use the already existing powerful finite-dimensional optimization codes. One of the main features of such codes is that they use fully the Euclidian structure of the finite-dimensional space $\mathbf{R}^n$. Therefore we shall prefer the usual Euclidian norm $\|\cdot\|_2$ on each discretized space X_n and Y_n, since there is norm equivalence at each iteration.

But as n grows, this equivalence is gradually lost and the norm equivalence ratio between the Euclidian norm on X_n (or Y_n) and $\|\cdot\|_{X_n} = \|\cdot\|_X$ (or $\|\cdot\|_{Y_n} = \|\cdot\|_Y$) tends to diverge: Euclidian norm becomes less and less relevant to use as it is less "representative" of the problem. Indeed, replacing directly $\|\cdot\|_X$ (or $\|\cdot\|_Y$) by the Euclidian norm yields too strong conditions, that we do not want as the efficiency of the algoritm would be lost.

A straightforward conclusion would be to keep formulation as is. As a consequence, the structure is to be propagated in the inner iteration (Step 1), namely into the finite-dimensional solving process. This yields in altering the finite-dimensional solver to be suitable to the structure of X and Y. Many solvers work for any kind of inner product, but such modifications require to go deep in the solver code. Moreover, performance could be lost in the process. Another drawback is that using the inner product $\langle\cdot,\cdot\rangle_X$ (or $\langle\cdot,\cdot\rangle_Y$) directly affects the gradient definition and the adjoint operator definition (when computing $\nabla_x\Phi_n$ for instance) and yields complicated computations.

A workaround is to consider a representation point of view in an analytical way. Of course, this point of view is already used in the implementation, since we "represent" discretized functions by, say, an ordered list of their values at discretization points. From the rank theorem, we know that there exists a bijective linear map σ_n^X from X_n to $\mathbf{R}^{\dim X_n}$: we define in the same way the bijective linear map σ_n^Y from Y_n to $\mathbf{R}^{\dim Y_n}$. We say that $\sigma_n^X(x)$ is the *representation* of x. We shall stress out that the fact that the representation process used for implementation is ***linear*** is an crucial assumption: otherwise we will not be able to write the algorithm in a simple way.

We now precise how things go when implemented. As a matter of fact, we never have access the true object and can only work with its representation. Thus what is really implemented is not f_n itself, but $\overline{f_n} = f_n \circ \left(\sigma_n^X\right)^{-1}$. We do not implement c_n as well, but $\overline{c_n} = \sigma_n^Y \circ c_n \circ \left(\sigma_n^X\right)^{-1}$. In the same way, we do not implement $\langle\cdot,\cdot\rangle_Y$, but[1] $\langle\overline{\lambda_1},\overline{\lambda_2}\rangle_{\overline{Y}} = \left\langle \left(\sigma_n^Y\right)^{-1}\overline{\lambda_1}, \left(\sigma_n^Y\right)^{-1}\overline{\lambda_2}\right\rangle_Y$. In this section, x denotes an element of X_n and $\overline{x}$ an element of $\mathbf{R}^{\dim X_n}$ (same for $\lambda \in Y_n$ and $\overline{\lambda} \in \mathbf{R}^{\dim Y_n}$). We can now compute the representation $\overline{\Phi_n}$ of the augmented Lagrangian and its derivative:

$$\begin{aligned}\overline{\Phi_n}(\overline{x},\overline{\lambda},\mu) &\triangleq \Phi_n\left(\left(\sigma_n^X\right)^{-1}\overline{x}, \left(\sigma_n^Y\right)^{-1}\overline{\lambda}, \mu\right)\\ &= \overline{f_n}(\overline{x}) + \left\langle\overline{\lambda}, \overline{c}(\overline{x})\right\rangle_{\overline{Y}} + \frac{\langle\overline{c}(\overline{x}),\overline{c}(\overline{x})\rangle_{\overline{Y}}}{2\mu}\\ \left(\nabla_{\overline{x}}\overline{\Phi_n}(\overline{x},\overline{\lambda},\mu)\middle|\overline{h}\right)_2 &= \left(\nabla\overline{f_n}(\overline{x})\middle|\overline{h}\right)_2 + \left\langle\overline{\lambda}+\overline{c}(\overline{x})/\mu, \overline{c}'(\overline{x})\cdot\overline{h}\right\rangle_{\overline{Y}}\end{aligned}$$

We also have the following lemma:

Lemma 1. *Let $\varphi : X_n \to \mathbf{R}$ be a $\mathscr{C}^2$ map. We denote $\overline{\varphi} = \varphi \circ \left(\sigma_n^X\right)^{-1}$. We also assume that the following relation holds:*

$$\forall x \in X_n,\ \kappa_n^X \|x\|_{X_n} \leq \|\sigma_n^X x\|_2 \leq K_n^X \|x\|_{X_n}$$

where $\|\cdot\|_2$ is the usual Euclidian norm on $\mathbf{R}^{\dim X_n}$ and κ_n^X and K_n^X are positive scalars. Then, we have the following relation:

$$\forall x \in X_n,\ \kappa_n^X \left\|\nabla\overline{\varphi}\left(\sigma_n^X x\right)\right\|_2 \leq \|\nabla\varphi(x)\|_{X_n} \leq K_n^X \left\|\nabla\overline{\varphi}\left(\sigma_n^X x\right)\right\|_2$$

---

[1] $\langle\cdot,\cdot\rangle_{\overline{X}}$ is not required in the implementation, except in some cases for the computation of f_n and/or c_n. This inner product is defined exactly in the same way.

Both relations are similar except that roles of $\|\cdot\|_2$ and $\|\cdot\|$ are exchanged. A proof of this lemma can be found at the end of this paper in Appendix A page 16. This result does not hold if σ_n^X is not linear.

An important remark has to be done: we cannot carelessly exchange the representation operation σ_n^X and the gradient operation ∇. Indeed in general, $\nabla\overline{\varphi}\left(\sigma_n^X x\right) \neq \sigma_n^X \nabla\varphi(x)$.

The trick is now to solve at each inner iteration the subproblem using $\overline{\Phi_n}$ instead of Φ_n. By doing this, we reduce each inner iteration to a classical finite-dimensional optimization problem on some $\mathbf{R}^p$ using the canonic Euclidian structure. We can now reformulate the whole algorithm in the representation point of view:

```
Algorithm ALDISCR (Representation point of view)

Step 0: Initialization  We are given an initial discretization level $n_0$, a initial Lagrange multiplier estimate $\lambda_0 \in Y_{n_0}$ and a penalty parameter $\mu_0 \in [0, 1[$.
    We choose some constants $\omega_*$, $\eta_*$, $\tau$, $\alpha$, $\alpha_\eta$ and $\beta_\eta$ all in $[0, 1[$.
    Set $k = 0$, $\omega_0 = \mu_0$ and $\eta_0 = \mu_0^{\alpha_\eta}$. Set $\overline{\lambda}_0 = \sigma_{Y_{n_0}}\lambda_0$.

Step 1: Inner iteration  Find $\overline{x}_k \in \mathbf{R}^{\dim X_{n_k}}$ such that
    $\|\nabla_x \overline{\Phi_{n_k}}(\overline{x}_k, \overline{\lambda}_k, \mu_k)\|_2 \le \omega_k/(2K_{n_k}^X)$ with $K_{n_k}^X$ defined as in Lemma 1

Step 2: Test for convergence  If $\omega_k \le \omega_*$, $\|\overline{c_{n_k}}(\overline{x}_k)\|_{\overline{Y}} + \varepsilon_{c,n_k}(\overline{x}_k) \le \eta_*/2$, stop

Step 3: Updates  If $\|\overline{c_{n_k}}(\overline{x}_k)\|_{\overline{Y}} \le \eta_k$, execute Step 3a. Otherwise, execute Step 3b.
    Step 3a: Update Lagrange mutliplier  Set $\overline{\lambda}_{k+1} = \overline{\lambda}_k + \overline{c_{n_k}}(\overline{x}_k)/\mu_k$,
        $\mu_{k+1} = \mu_k$, $\omega_{k+1} = \omega_k\mu_{k+1}$ and $\eta_{k+1} = \eta_k\mu_{k+1}^{\beta_\eta}$
    Step 3b: Update penalty parameter  Set $\overline{\lambda}_{k+1} = \overline{\lambda}_k$, $\mu_{k+1} = \tau\mu_k$,
        $\omega_{k+1} = \mu_{k+1}$, $\eta_{k+1} = \mu_{k+1}^{\alpha_\eta}$

Step 4: Refinement  Choose $n_{k+1} \ge n_k$ such that:
        $\varepsilon_{c,n_{k+1}}(\overline{x}_k) < \min\{\alpha\eta_{k+1}, \mu_{k+1}\omega_{k+1}\}$ and
        $\varepsilon_{\Phi,n_{k+1}}\left(\overline{x}_k, \sigma_{n_{k+1}}^X \circ \left(\sigma_{n_k}^X\right)^{-1}\overline{\lambda}_{k+1}, \mu_{k+1}\right) \le \omega_{k+1}/2$.
    Set $\overline{\lambda}_{k+1} = \sigma_{n_{k+1}}^X \circ \left(\sigma_{n_k}^X\right)^{-1}\overline{\lambda}_{k+1}$. Increment $k$ by one and go to Step 1.
```

The assumption on the linear property of representation is fully used, in particular in Step 1 when using Lemma 1 and in Step 3a when updating the Lagrange multiplier. The instruction “Set $\overline{\lambda}_{k+1} = \sigma^X_{n_{k+1}} \circ \left(\sigma^X_{n_k}\right)^{-1} \overline{\lambda}_{k+1}$” in Step 4 is the injection from X_{n_k} to $X_{n_{k+1}}$ in the representation formulation.

The most important change occurs in Step 1 where we have changed the initial Lagrangian subproblem into a typical finite-dimensional Lagrangian subproblem with the usual Euclidian structure. Therefore we only need to compute $\nabla\overline{\Phi_n}$ instead of $\overline{\nabla\Phi}_n$. Indeed computing the latter is more complex since it requires the full knowledge of the Hilbertian structure of X (inner product, adjoint definition, ...).

The computation of the derivative of $\overline{\Phi_n}$ can be done easily by hand, and can even be performed by automatic differentiation for instance. The only needed additional element is an upper bound K^X_n of the norm of $\sigma^X_n \in \mathscr{L}(X_n, \mathbf{R}^{\dim X_n})$ to adjust to the termination condition for the inner iteration solving.

3 Numerical examples

Algorithm parameters Unless explicitly precised, we used the following set of parameters: $\omega_* = 10^{-6}$, $\eta_* = 10^{-6}$, $\tau = 0.1$, $\alpha = 0.9$, $\alpha_\eta = 0.1$ and $\beta_\eta = 0.9$. We choose $\mu_0 = 0.1$ as well.

Inner iteration solver The solver used for inner iteration is a trust-region based solver[1]. Unless specified, we use the quadratic submodel at each inner iteration and compute the step with STEIHAUG-TOINT method (see [1] page 205). The approximate Hessian is evaluated by finite differences.

Machine, OS and technologies Simulations were run on a machine with two 2.4GHz processors under Linux system. Implementation is currently made in MatLAB (Release 14 Service Pack 3), since we do not focus on efficiency for the moment, but on feasibility.

3.1 (OP0) An finite-dimensional optimization problem

We consider the following problem :

$$\begin{cases} \min\limits_{(x,y)\in\mathbf{R}^2} x^2 + y^2 \\ x + y = 1 \end{cases} \tag{OP0}$$

whose solution is $(x^*, y^*, \lambda^*) = (1/2, 1/2, -1)$.

3.1.1 One-level discretization

Here, we have $X = \mathbf{R}^2$ and $Y = \mathbf{R}$. For starters, we can choose the sequences X_n and Y_n to be constant and equal to X and Y respectively. Hence[2] $f_n = f$ and $c_n = c$ for every n, so there is no discretization error : $\varepsilon_{c,n}(x, y) = \varepsilon_{\Phi,n}(x, y, \lambda, \mu) = 0$.

So running the algorithm will be equivalent to run the original augmented-Lagrangian on (OP0), performing iterations with increasing criticality.

Numerical result The solution is found from the initial guess $x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$ and $\lambda_0 = 0$ with the required precision after seven outer iterations in around 0.1 second.

3.1.2 Two-level discretization

We can go a bit further and refine our discretization scheme. Nothing is changed for Y: $Y_n = Y = \mathbf{R}$ for every n. But let us suppose we have two levels of discretization for X : $X_1 = \mathbf{R}$ and $X_n = X = \mathbf{R}^2$ for $n > 1$ with the canonical embedding $X_1 \sim \mathbf{R} \times \{0_{\mathbf{R}}\} \subset \mathbf{R}^2$.

Discretization error on constraints c Let us consider $f_1(x) = f(x, 0) = x^2$ and $c_1(x) = c(x, 0) = x - 1$. If we consider $(x, 0) \in X_1$, applying c_1 or c_n with $n > 1$ is the same. If we consider $(x, y) \in X_n$ with $n > 1$, we go back to the one-level case. Therefore we have $\varepsilon_{c,n}(x, y) = 0$ for every n and $(x, y) \in X_n$.

Discretization error on augmented Lagragian gradient $\nabla\Phi$ For $(x, 0) \in X_1$, $c_1'(x) = \begin{pmatrix} 1 & 0 \end{pmatrix}$ — once embedded back in $\mathbf{R}^2$ — and $c_n'(x, 0) = c'(x, 0) = \begin{pmatrix} 1 & 1 \end{pmatrix}$ for $n > 1$. Then we have:

$$\begin{aligned}
&\nabla_x\Phi((x, 0), \lambda, \mu) - \nabla_x\Phi_1(x, \lambda, \mu) \\
&\quad = \nabla f(x, 0) - \nabla f_1(x) + [c'(x, 0) - c_1'(x)]^*\lambda + \frac{1}{\mu}\left[c'(x, 0)^* c(x, 0) - c_1'(x)^* c_1(x)\right] \\
&\quad = 0 + \lambda \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \frac{1}{\mu}[c'(x, 0) - c_1'(x)]^* c_1(x) \qquad \text{since } c_1(x) = c(x, 0) \\
&\quad = \left(\lambda + \frac{x-1}{\mu}\right) \begin{pmatrix} 0 \\ 1 \end{pmatrix}
\end{aligned}$$

so we get the following result:

$$\|\nabla_x\Phi((x, 0), \lambda, \mu) - \nabla_x\Phi_1(x, \lambda, \mu)\| = \left|\lambda + \frac{x-1}{\mu}\right|.$$

---

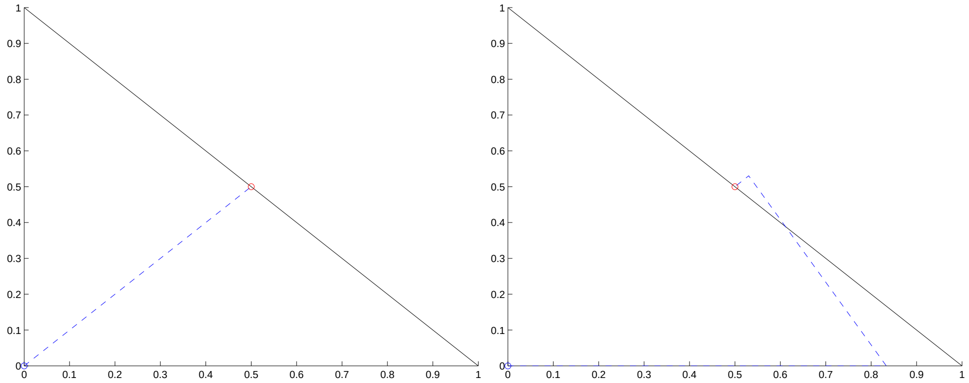
[2] In this section, we trivially have $\sigma_n^X = \mathrm{id}_X$ and $\sigma_Y = \mathrm{id}_Y$. Therefore we decide to drop the notation $\bar{\cdot}$ in this section.

If $(x, y) \in X_n$ with $n > 1$, we go back to the one-level case. Hence we have

$$\varepsilon_{\Phi,n}(x, y, \lambda, \mu) = \left|\lambda + \frac{x-1}{\mu}\right| \delta_{1n}.$$

where δ_{ij} is the KRONECKER symbol.

Numerical result The solution is found from the initial guess $x_0 = 0$ and $\lambda_0 = 0$ with the required precision after seven outer iterations in around 0.1 second.



(a) One-level discretization

(b) Two-level discretization

Figure 1: Resolution for problem (OP0). The blue circle is $x_0 = (0.0, 0.0)$. The dotted broken line is the path of iterates x_n and the circle is the exact solution $x^* = (0.5, 0.5)$. The black solid line is the set of points (x, y) such that $x + y = 1$.

3.2 (POISSON-1D) Infinite-dimensional optimization problem without constraints

We consider the one-dimensional POISSON equation on the interval $[0, 1]$:

$$\begin{cases} -\Delta u = g & \text{almost everywhere on } [0, 1] \\ u(0) = u(1) = 0 \end{cases}$$

Considering the weak formulation of this PDE, one can use the LAX-MILGRAM theorem. Roughly, this theorem states that the PDE is equivalent to the following optimisation problem:

$$\begin{cases} \min \frac{1}{2} \int_0^1 \dot{u}(t)^2 \, dt - \int_0^1 g(t) u(t) \, dt \\ u \in X \overset{\Delta}{=} \mathrm{H}_0^1([0, 1], \mathbf{R}) \end{cases} \tag{POISSON-1D}$$

where $\mathrm{H}^1_0([0,1],\mathbf{R}) = \{u \in \mathrm{L}^2([0,1],\mathbf{R}) \,|\, u' \in \mathrm{L}^2([0,1],\mathbf{R}), u(0) = u(1) = 0\}$ is an infinite-dimensional Hilbert space with inner product $(v_1 \,|\, v_2)_X = \int_0^1 \dot{v_1}(t)\dot{v_2}(t)\,dt$.

There is no constraint in this problem, so we can consider $Y \overset{\Delta}{=} \mathbf{R}$ for instance and c the zero function. We will choose c_n (hence $\overline{c_n}$) to be zero as well, so $\varepsilon_{c,n}(\overline{x}) = 0$ and $\|\nabla_x \Phi - \nabla_x \Phi_n\| = \|\nabla f - \nabla f_n\|$.

3.2.1 Discretization of space X

This part will be general, as this discretization scheme may occur in further examples (typically, the ones involving functional objects). Let us have $I = [a,b]$ an bounded interval of $\mathbf{R}$.

When considering a subdivision $\Sigma = \{t_i\}_{0 \leq i \leq N} \subset I$ as an increasing sequence of $N+1$ values of I with $t_0 = a$ and $t_N = b$, we will denote $|\Sigma| \overset{\Delta}{=} N$. (*Be warned that it is different from the usual writing since* $\operatorname{card}\Sigma = N+1$.)

Let us consider a subdivision of I. We will denote by $\mathrm{L}^2_\Sigma(I,\mathbf{R}^p)$ the subset of square-integrable functions constant on each subinterval[3] $[t_{i-1}, t_i]$ defined by Σ. For any given subdivision Σ, $\mathrm{L}^2_\Sigma(I,\mathbf{R}^p)$ is a finite-dimensional Hilbert subspace of $\mathrm{L}^2(I,\mathbf{R}^p)$.

Still considering Σ as subdivision of I, we define $\mathrm{H}^1_\Sigma(I,\mathbf{R}^p)$ as the subset of square-integrable functions that are continuous and affine on each subinterval $[t_{i-1}, t_i[$. For any given subdivision Σ, $\mathrm{H}^1_\Sigma(I,\mathbf{R}^p)$ is a finite-dimensional Hilbert subspace of $\mathrm{H}^1(I,\mathbf{R}^p)$.

In the following, we will note $h_i \overset{\Delta}{=} t_i - t_{i-1}$ and $[u]_{i-1} \overset{\Delta}{=} [u(t_i) - u(t_{i-1})]/h_i$. Note that $[u]_{i-1}$ is linear with respect to u.

With this definition, the image of the differentiation operator on $\mathrm{H}^1_\Sigma(I,\mathbf{R}^p)$ is exactly $\mathrm{L}^2_\Sigma(I,\mathbf{R}^p)$ and if $u \in \mathrm{H}^1_\Sigma(I,\mathbf{R}^p)$, we have

$$u' = \sum_{i=1}^{|\Sigma|} [u]_{i-1} \cdot \mathbf{1}_{[t_{i-1}, t_i[} \in \mathrm{L}^2_\Sigma(I,\mathbf{R}^p)$$

If $u \notin \mathrm{H}^1_\Sigma(I,\mathbf{R}^p)$, we cannot exchange the discretization and the derivation operations in general. Moreover the inclusion $\mathrm{H}^1 \subset \mathrm{L}^2$ is not conserved by the discretisation, i.e. we have $\mathrm{H}^1_\Sigma(I,\mathbf{R}^p) \not\subset \mathrm{L}^2_\Sigma(I,\mathbf{R}^p)$.

We straightforwardly consider the discretized space $\mathrm{H}^1_{0,\Sigma}(I,\mathbf{R}^p)$ as the subset of $\mathrm{H}^1_\Sigma(I,\mathbf{R}^p)$ such that $u(t_0) = u(t_{|\Sigma|}) = 0$, i.e.

$$\mathrm{H}^1_{0,\Sigma}(I,\mathbf{R}^p) = \mathrm{H}^1_\Sigma(I,\mathbf{R}^p) \cap \mathrm{H}^1_0(I,\mathbf{R}^p).$$

---

[3] A subdivision with $N+1$ points defines N subintervals, so $|\Sigma|$ is the number of subintervals.

The discretization operation is merely the restriction of the $\mathrm{H}^1_\Sigma(I, \mathbf{R}^p)$ discretization operation on $\mathrm{H}^1_{0,\Sigma}(I, \mathbf{R}^p)$.

Nested discretized spaces are linked to nested subdivisions. Indeed, if $\Sigma \subset \Sigma'$ (ie. Σ' is finer than Σ), then $\mathrm{L}^2_\Sigma(I, \mathbf{R}^p) \subset \mathrm{L}^2_{\Sigma'}(I, \mathbf{R}^p)$ and the same goes for H^1- and H^1_0-spaces.

3.2.2 Return to (POISSON-1D)

We go back to our example. Here $I = [0, 1]$. As the algorithm required nested discretized spaces, we need nested subdivisions. For starters, we are going to use uniform subdivisions obtained by dichotomy:

$$\begin{aligned}
\Sigma_0 &= \{0, 1\} \\
\Sigma_1 &= \{0, 1/2, 1\} \\
\Sigma_2 &= \{0, 1/4, 1/2, 3/4, 1\} \\
\vdots \quad & \quad \vdots \\
\Sigma_{n+1} &= \Sigma_n \cup \{(t_{i-1} + t_i)/2 \text{ where } \Sigma_n = \{t_i\}_{0 \le i \le |\Sigma_n|}\}
\end{aligned}$$

That is each subdivision is obtained by considered all points from the previous with all their midpoints in addition. With our notation, $|\Sigma_n| = 2^n$ and each subinterval $[t_{i-1}, t_i]$ of Σ_n is of constant length $h_i = h = 2^{-n}$. All subdvisions are nested ($\Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_n \subset \Sigma_{n+1} \subset \cdots \subset I$), so the associated discretized spaces are nested. For the sake of simplicity, we will drop the subscript Σ and only keep the level n of discretization : $\mathrm{L}^2_n([0, 1], \mathbf{R}) \triangleq \mathrm{L}^2_{\Sigma_n}([0, 1], \mathbf{R})$, the same being done for H^1- and H^1_0-spaces.

Since $X = \mathrm{H}^1_0([0, 1], \mathbf{R})$, we choose the sequence X_n to be $X_n \triangleq \mathrm{H}^1_{0,n}([0, 1], \mathbf{R})$. We can now define the discretized versions f_n of the criterion f:

$$f_n(u) \triangleq \frac{1}{2} \sum_{i=1}^{N} h_i [u]_{i-1}^2 - \sum_{i=1}^{N} h_i g(t_{i-1}) u_{i-1}$$

with $N = |\Sigma_n| = 2^n$, $u_i = u(t_i)$, $h_i = t_i - t_{i-1}$ and $[u]_{i-1} = (u_i - u_{i-1})/h_i$.

We use the straightforward representation $\overline{u} = \sigma_n^X u = (u_k)_{0<k<N}$ and we note $\overline{u}_k = u_k$ and $[\overline{u}]_{i-1} = (\overline{u}_i - \overline{u}_{i-1})/h_i$. We now compute $\partial \overline{f_n} / \partial \overline{u}_k$ for $0 < k < N$ (with $\overline{u}_0 = \overline{u}_N = 0$ by convention):

$$\begin{aligned}
\frac{\partial \overline{f_n}}{\partial \overline{u}_k}(\overline{u}) &= \frac{1}{2} \sum_{i=1}^{N} h_i \frac{\partial [\overline{u}]_{i-1}^2}{\partial \overline{u}_k} - \sum_{i=1}^{N} h_i g(t_{i-1}) \frac{\partial \overline{u}_{i-1}}{\partial \overline{u}_k} \\
&= -\frac{\overline{u}_{k+1} - 2\overline{u}_k + \overline{u}_{k-1}}{h} - h g(t_k)
\end{aligned}$$

Going on with the computations, we consider $f - f_n$. If $u \in X_n$, we get:

$$\begin{aligned}(f - f_n)(u) &= -\left(\int_0^1 g(t)u(t)\,dt - \sum_{i=1}^{N} h_i g(t_{i-1})u_{i-1}\right). \\ &= \sum_{i=1}^{N}\int_{t_{i-1}}^{t_i} [g(t_{i-1})u_{i-1} - g(t)u(t)]\,dt \\ &= \sum_{i=1}^{N}\int_{t_{i-1}}^{t_i} [g(t_{i-1})\{u_{i-1} - u(t)\} + \{g(t_{i-1}) - g(t)\}u(t)]\,dt\end{aligned}$$

Since $u \in X_n$, u is continuous piecewise-affine. Then, on every subinterval $[t_{i-1}, t_i]$, we have $u(t) = u_{i-1} + [u]_{i-1}(t - t_{i-1})$, so we can write:

$$(f - f_n)(u) = \sum_{i=1}^{N} -\frac{h_i^2}{2} g(t_{i-1})[u]_{i-1} + \int_{t_{i-1}}^{t_i} \{g(t_{i-1}) - g(t)\}u(t)\,dt.$$

Since $f - f_n$ is linear, for $v \in X_n$, we have $\nabla(f - f_n)(u) \cdot v = (f - f_n)(v)$ and:

$$|\nabla(f - f_n)(u) \cdot v| \leq \sum_{i=1}^{N} \frac{h_i^2}{2} |g(t_{i-1})| \cdot |[v]_{i-1}| + \int_{t_{i-1}}^{t_i} |g(t_{i-1}) - g(t)| \cdot |v(t)|\,dt$$

In the rest of the computation, we assume that g is γ-Lipschitzian on [0, 1], so we get, using the POINCARÉ inequality:

$$|\nabla(f - f_n)(u) \cdot v| \leq \left(\frac{1}{2}\max_{[0,1]}|g| + \frac{\gamma}{\sqrt{6}}\right)\frac{1}{N}\|v\|_X$$

We can now conclude that, for $u \in X_{n^*}$, $\lambda \in Y_{n^*}$ and $\mu > 0$, we have:

$$\|\nabla_x\Phi(u, \lambda, \mu) - \nabla_x\Phi_n(u, \lambda, \mu)\| = \|\nabla f(u) - \nabla f_n(u)\| \leq \left(\frac{1}{2}\max_{[0,1]}|g| + \frac{\gamma}{\sqrt{6}}\right)\frac{1}{N}$$

so we choose $\varepsilon_{\Phi,n}(\overline{u}, \overline{\lambda}, \mu) = \left(\frac{1}{2}\max_{[0,1]}|g| + \frac{\gamma}{\sqrt{6}}\right)\frac{1}{N}$.

We now link Euclidian and induced norms on X_n:

$$\|\overline{u}\|_{\mathbf{R}^{N-1}}^2 = \frac{1}{h}\|u\|_{\mathrm{L}^2}^2 \leq \frac{1}{2h}\|u\|_X^2 = \frac{1}{2h}\|u\|_{X_n}^2$$

so we get $\|\overline{u}\|_{\mathbf{R}^{N-1}} \leq \sqrt{N/2}\|u\|_{X_n}$ with $N = 2^n$. Thus we choose $K_n^X = \sqrt{N/2}$.

Numerical result We consider $g : t \mapsto 1 - 3(2x-1)^2$ so the solution of the Poisson equation should be $u : t \mapsto x^2(x-1)^2$. We have $\max_{[0,\,1]} |g| = |g(0)| = |g(1)| = 2$ and $\gamma = 12$.

Using the refinement designed in Step 4 in this example seems counter-productive. Indeed, if $n_0 = 1$, we get $n_1 = 11$ (that is 2049 discretization points directly infered from 3 discretization points) and computation is very costly. We use instead the straightforward update $n_{k+1} = n_k + 1$. The solution is found from the initial guess $x_0 = 0 \in \mathrm{H}^1_{0,1}([0,\,1], \mathbf{R})$ and $\lambda_0 = 0$ with the required precision after seven outer iterations in around 30 seconds.

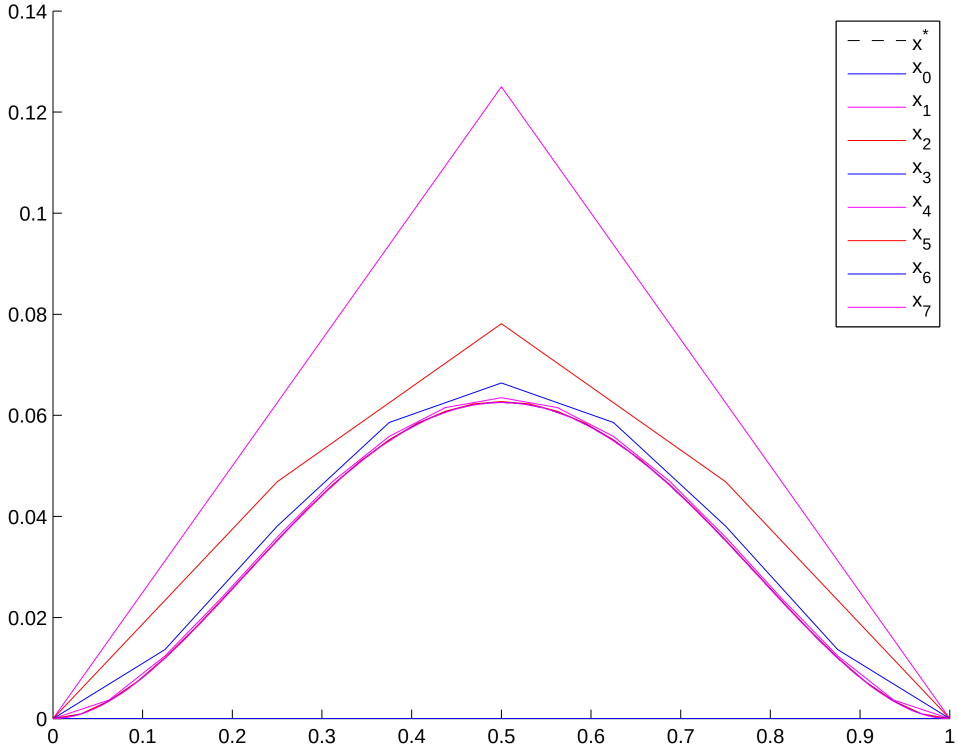


Figure 2: Solution x^* and `ALDISCR` iterates x_n for problem (POISSON-1D)

3.3 (DIDO) Infinite-dimensional problem with scalar constraints

We consider the isoperimetric problem also known as the *Dido problem*:

$$\begin{cases} \min -\displaystyle\int_0^1 u(t)\,dt \\ u \in X \overset{\Delta}{=} \mathrm{H}_0^1([0,\,1],\mathbf{R}) \\ \qquad \displaystyle\int_0^1 \sqrt{1+\dot{u}(t)^2}\,dt = \frac{\pi}{2} \end{cases} \tag{DIDO}$$

We are using the framework defined in the previous section: we set $X = \mathrm{H}_0^1([0,\,1],\mathbf{R})$ and $X_n = \mathrm{H}_{0,n}^1([0,\,1],\mathbf{R})$ as before with the same definition for the representation σ_n^X. Since c is a scalar function, we get $Y_n = Y = \mathbf{R}$. We can now define the discretized versions f_n and c_n of the criterion f and the constraints c respectively:

$$\overline{f_n}(\overline{u}) \overset{\Delta}{=} -\sum_{i=1}^{N} h_i \overline{u}_{i-1}$$
$$\overline{c_n}(\overline{u}) \overset{\Delta}{=} \sum_{i=1}^{N} h_i \sqrt{1+[\overline{u}]_{i-1}^2} - \frac{\pi}{2}$$

so we have for $0 < k < N$ (with $u_0 = u_N = 0$):

$$\begin{aligned} \frac{\partial \overline{f_n}}{\partial \overline{u}_k}(\overline{u}) &= -\sum_{i=1}^{N} h_i \frac{\partial \overline{u}_{i-1}}{\partial \overline{u}_k} = -h_{k+1} = -h \\ \frac{\partial \overline{c_n}}{\partial \overline{u}_k}(\overline{u}) &= \sum_{i=1}^{N} h_i \frac{\partial}{\partial \overline{u}_k}\left(\sqrt{1+[\overline{u}]_{i-1}^2}\right) \\ &= -\frac{[\overline{u}]_k}{\sqrt{1+[\overline{u}]_k^2}} + \frac{[\overline{u}]_{k-1}}{\sqrt{1+[\overline{u}]_{k-1}^2}} \end{aligned}$$

Since c_n coincide on X_n with c, we have $\varepsilon_{c,n}(u) = 0$ and $\|\nabla_x\Phi - \nabla_x\Phi_n\| = \|\nabla f - \nabla f_n\|$.

f and f_n are linear, so $f - f_n$ is linear too. If $v \in X_n$, we get:

$$\begin{aligned} \nabla(f-f_n)(u)\cdot v = (f-f_n)(v) &= -\int_0^1 v(t)\,dt + \sum_{i=1}^{N} h_i v_{i-1} \\ &= -\frac{1}{2N}\sum_{i=1}^{N} h_i [v]_{i-1} = -\frac{1}{2N}\int_0^1 v'(t)\,dt \end{aligned}$$
$$|\nabla(f-f_n)(u)\cdot v| \leq \frac{1}{2N}\int_0^1 |v'(t)|\,dt \leq \frac{1}{2N}\|v\|_X$$

Therefore, if $u \in X_{n^*}$, $\lambda \in Y_{n^*}$ and $\mu > 0$, we have:

$$\|\nabla_x \Phi(u, \lambda, \mu) - \nabla_x \Phi_n(u, \lambda, \mu)\| = \|\nabla f(u) - \nabla f_n(u)\| \leq \frac{1}{2N}$$

so we choose $\varepsilon_{\Phi,n}(\overline{u}, \overline{\lambda}, \mu) = \dfrac{1}{2N}$.

Numerical result The solution cannot be found from the initial guess $x_0 = 0 \in \mathrm{H}^1_{0,1}([0, 1], \mathbf{R})$ and $\lambda_0 = 0$ with the required precision since too much inner iterations are required for solving the Lagrangian subproblem.

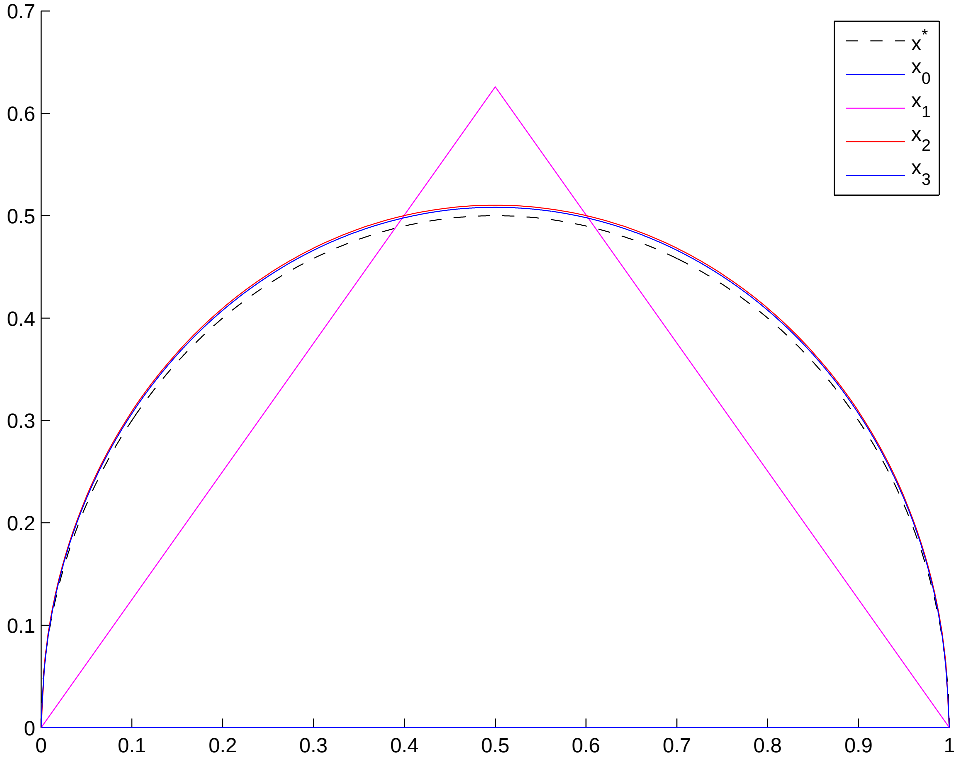


Figure 3: Solution x^* and `ALDISCR` iterates x_n for problem (DIDO)

4 Conclusion

The current implementation of algorithm `ALDISCR` seems to work on simple examples, but still have problems for problems a little more complex: since we are in a preliminary testing stage, some errors are likely to be found and corrected.

Aside from the implementation point of view, we can see this algorithm as an — enhanced — optimization solving process coupled with a classical mesh refinement. Recent works [2] have shown that multi-grid frameworks (in particular, recursive multigrid with trust-region methods) are more efficient than mere mesh refinement. Thus the algorithm may be extended with a recursive multi-grid framework.

References

[1] Andrew CONN, Nicholas GOULD, and Philippe TOINT, *Trust-region methods*, MPS-SIAM Series on Optimization, vol. 1, SIAM, Philadelphia, USA, 2000.

[2] Serge GRATTON, Annick SARTENAER, and Philippe TOINT, *Recursive trust-region methods for multiscale nonlinear optimization*, SIAM Journal on Optimization **19** (2008), no. 1, 414–444.

[3] Ekkehard SACHS and Annick SARTENAER, *A class of augmented lagrangian algorithms for infinite-dimensional optimization with equality constraints*, Tech. report, University of Namur, Belgium, 2005.

A Proof of Lemma 1

Let us write the lemma once again:

Lemma 1. *Let* $\varphi : X_n \to \mathbf{R}$ *be a* $\mathscr{C}^2$ *map. We denote* $\overline{\varphi} = \varphi \circ \left(\sigma_n^X\right)^{-1}$. *We also assume that the following relation holds:*

$$\forall x \in X_n,\ \kappa_n^X \|x\|_{X_n} \leq \|\sigma_n^X x\|_2 \leq K_n^X \|x\|_{X_n} \tag{5}$$

where $\|\cdot\|_2$ *is the usual Euclidian norm on* $\mathbf{R}^{\dim X_n}$. *Then, we have the following relation:*

$$\forall x \in X_n,\ \kappa_n^X \left\|\nabla\overline{\varphi}\left(\sigma_n^X x\right)\right\|_2 \leq \|\nabla\varphi(x)\|_{X_n} \leq K_n^X \left\|\nabla\overline{\varphi}\left(\sigma_n^X x\right)\right\|_2 \tag{6}$$

Proof. Let x and h be in X_n. We note $\overline{x} \overset{\Delta}{=} \sigma_n^X x \in \mathbf{R}^{\dim X_n}$ and $\overline{h} \overset{\Delta}{=} \sigma_n^X h \in \mathbf{R}^{\dim X_n}$ the representations of x and h respectively. Then we have:

$$\begin{aligned}
\overline{\varphi}'(\overline{x}) \cdot \overline{h} &= \left(\varphi \circ \left(\sigma_n^X\right)^{-1}\right)'(\overline{x}) \cdot \overline{h} \\
&= \varphi'\left(\left(\sigma_n^X\right)^{-1}(\overline{x})\right) \cdot \left[\left(\left(\sigma_n^X\right)^{-1}\right)'(\overline{x}) \cdot \overline{h}\right] \\
&= \varphi'\left(\left(\sigma_n^X\right)^{-1}(\overline{x})\right) \cdot \left[\left(\sigma_n^X\right)^{-1}(\overline{h})\right] \text{ since } \left(\sigma_n^X\right)^{-1} \text{ is linear} \\
&= \varphi'(x) \cdot h
\end{aligned}$$

By using gradient definition, we get the following relation for all x and h in X_n:

$$\langle \nabla\varphi(x)\,,\, h\rangle_X = \left(\nabla\overline{\varphi}(\overline{x})\,|\,\overline{h}\right)_2. \tag{7}$$

Let us compute $\|\nabla\varphi(x)\|_X$:

$$\begin{aligned}
\|\nabla\varphi(x)\|_X^2 &= \langle \nabla\varphi(x)\,,\, \nabla\varphi(x)\rangle_X \\
&= \left(\nabla\overline{\varphi}(\overline{x})\,\middle|\,\sigma_n^X \nabla\varphi(x)\right)_2 \text{ by using (7) with } h = \nabla\varphi(x) \in X_n \\
&\leq \|\nabla\overline{\varphi}(\overline{x})\|_2 \left\|\sigma_n^X \nabla\varphi(x)\right\|_2 \text{ (CAUCHY-SCHWARTZ inequality)} \\
&\leq K_n^X \|\nabla\overline{\varphi}(\overline{x})\|_2 \|\nabla\varphi(x)\|_X \text{ by using (5) with } x = \nabla\varphi(x) \in X_n
\end{aligned}$$

so we get $\|\nabla\varphi(x)\|_X \leq K_n^X \|\nabla\overline{\varphi}(\overline{x})\|_2$. The remaining inequality is obtained in the same way by using (7) and (5) on the vector $\left(\sigma_n^X\right)^{-1} \nabla\overline{\varphi}(\overline{x}) \in X_n$. □