Smooth Strongly Convex Regression

Andrea Simonetto
IBM Research Ireland
Dublin, Ireland
andrea.simonetto@ibm.com

Abstract—Convex regression (CR) is the problem of fitting a
convex function to a finite number of noisy observations of an un-
derlying convex function. CR is important in many domains and
one of its workhorses is the non-parametric least square estimator
(LSE). Currently, LSE delivers only non-smooth non-strongly
convex function estimates. In this paper, leveraging recent results
in convex interpolation, we generalize LSE to smooth strongly
convex regression problems. The resulting algorithm relies on
a convex quadratically constrained quadratic program. We also
propose a parallel implementation, which leverages ADMM, that
leaves the overall computational complexity to a tight
\( O(n^2) \) for \( n \) observations. Numerical results support our findings.

I. INTRODUCTION

Convex regression (CR) is concerned with fitting a convex
function to a finite number of observations. In particular,
suppose that we are given \( n \) noisy observations of a convex
function \( \varphi : \mathbb{R}^d \to \mathbb{R} \) as

\[
y_i = \varphi(x_i) + \epsilon_i, \quad i \in I_n := \{1, \ldots, n\},
\]

where \( \epsilon_i \)'s are random variables, while \( x_i \in \mathbb{R}^d \). The objective of CR is then to estimate the true function \( \varphi \), given the
observations \( y_i \)'s, in a way in which the estimated function \( \hat{\varphi} \) is convex.

CR is a particular class of shape-constrained regression
problems, and since its first conception in the 50's, it has
attracted much attention in various domains, such as statistics,
economics, operations research, signal processing and control [1]–[3]. In economics, CR has been motivated by the need
for approximating consumers’ utility functions from empirical
data [4], a task which has been recently re-considered in the
context of personalized optimization with user’s feedback [5].

In this paper, we study least squares estimators (LSEs)
for CR. LSEs have some key advantages over many other
estimators proposed in the literature for CR (e.g., constrained Gaussian processes [6], splines [7], or others [8]). First, LSEs are non-parametric and hence they do not require any tuning and avoid the issue of selecting an appropriate estimation structure. Second, LSEs can be computed by solving a convex
quadratic program. Therefore, at least in theory, they can be solved very efficiently using interior point methods. Third,
being based on the least squares paradigm, they can be naturally extended to time-varying cases (when the function to be
estimated changes continuously in time) by, e.g., exponential
forgetting coefficients; these cases are becoming more and
more important in the current data streaming era [9].

One of the main theoretic drawbacks is however that the
class of functions that can be enforced is limited to the general
convex functions, while in many applications one would like to
be able to impose at least smoothness and/or strong convexity.

In this paper, our contributions are as follows,

- First, we propose a novel smooth strongly convex CR
algorithm. The resulting estimator has at its heart a convex
quadatically constrained quadratic program, with \( n + nd \)
variables and \( n(n - 1) \) constraints. We also report on its
computational complexity as a function of \( n \). The building
blocks for this novel algorithm are the recent results in smooth
strongly convex interpolation [10], [11].

- Second, we propose a decomposition scheme based on
the alternating direction method of multipliers (ADMM) to
leaves the computational complexity and make the method
parallel. The resulting computational complexity per iteration
is then \( O(n^2) \) which is tight (i.e., no LSE can obtain a lower
computational complexity). The ADMM approach is based
on a properly constructed constraint graph, as well as a dual
formulation of the local sub-problems.

The results presented in this paper generalize LSE to smooth
strongly convex functions, and the non-smooth results can be
re-obtained as a special case. \(^1\)

II. DEFINITIONS AND PROBLEM STATEMENT

We start by formally defining the functional class of interest.
Given two parameters \( \mu \) and \( L \) satisfying \( 0 \leq \mu < L \leq +\infty \),
we consider convex functions satisfying both a smoothness
and a strong convexity condition. Given a convex function
\( \varphi : \mathbb{R}^d \to \mathbb{R} \), we say that the function is \( L \)-smooth and \( \mu \)-strongly
convex, which we denote with \( F_{\mu,L} \), iff the following
two conditions are satisfied:

- Inequality \( 1/L \|g_1 - g_2\|_2 \leq \|x_1 - x_2\|_2 \) holds \forall x_1, x_2 \in
\mathbb{R}^d \) and corresponding subgradients \( g_1, g_2 \in \mathbb{R}^d \);

- Function \( \varphi(x) - \mu/2\|x\|_2^2 \) is convex.

These definitions allow for \( L \) to be equal to \( +\infty \) (i.e., the
non-smooth case). In the case of a finite \( L \), the first condition
implies differentiability of the function. When \( L = +\infty \), this
condition becomes vacuous, and the function can be non-
differentiable. The class of generic convex functions simply
contains a proper subset of \( F_{\mu,L} \), the case \( L = \mu \) can be discarded, as it
only involves simple quadratic functions, that can be estimated
parametrically much more efficiently.

\(^1\) Notation. Vectors are indicated with \( x \in \mathbb{R}^n \), and matrices with
\( A \in \mathbb{R}^{m \times n} \). The Euclidean norm is indicated with \( \|x\|_2 \), the infinity norm as
\( \|x\|_\infty \). \( (\cdot)^T \) is the transpose operator. Symmetric positive (semi)-definite
matrices of dimension \( n \) are indicated as \( A \in \mathbb{S}^n \) (\( A \in \mathbb{S}^n_+ \)). For convex
functions \( \varphi : \mathbb{R}^n \to \mathbb{R} \), we indicate with \( \partial \varphi(x) \) their subgradient at point \( x \)
and with \( \varphi^* \) their convex conjugate. \( O(\cdot) \) is the standard big-O notation.
The problem we are interested in solving can be then formalized by using the empirical \( l_2 \) norm as:

\[
\hat{\varphi}_n \in \arg\min_{\psi \in \mathcal{F}_{\mu,L}} \left\{ \sum_{i \in I_n} (y_i - \psi(x_i))^2 \right\}.
\]  

(2)

As we will see, the solution of (2) will consist of two parts: (i) the solution of a finite-dimensional optimization problem defined on the observation set, and (ii) an interpolating function which maintains the functional class also in all the other points of the domain. In this respect, we define the notion of \( \mathcal{F}_{\mu,L} \)-interpolation as follows.

**Definition 1:** The set \( \{(x_i, g_i, f_i)\}_{i \in I_n} \) where \( x_i, g_i \in \mathbb{R}^d \) and \( f_i \in \mathbb{R} \) for all \( i \in I_n \) is \( \mathcal{F}_{\mu,L} \)-interpolable iff there exists a function \( \varphi \in \mathcal{F}_{\mu,L} \), \( \varphi : \mathbb{R}^d \to \mathbb{R} \), such that \( g_i \in \partial \varphi(x_i) \) and \( f_i = \varphi(x_i) \) for all \( i \in I_n \).

### III. SHAPE-CONSTRAINED LEAST-SQUARES

#### A. State of the art

The infinite dimensional optimization problem in (2) can be reduced to a finite dimensional one for the case \( F_{0,+\varepsilon} \) as follows. Let \( f_i = \hat{\varphi}_n(x_i) \), for \( i \in I_n \) and define the vector \( f = [f_1, \ldots, f_n]^\top \in \mathbb{R}^n \). Let \( g_i = \partial \varphi(x_i) \), and define the vector \( g = [g_1^\top, \ldots, g_n^\top]^\top \in \mathbb{R}^{n d} \). We can now rewrite the optimization problem (2) for \( F_{0,+\varepsilon} \) on the observation set as the following convex quadratic program (QP):

\[
\begin{align*}
\text{minimize} & \quad f \in \mathbb{R}^n, \quad g \in \mathbb{R}^{n d} \sum_{i \in I_n} (y_i - f_i)^2 \\
\text{subject to:} & \quad f_j + g_j^\top (x_i - x_j) \leq f_i, \quad \forall i, j \in I_n.
\end{align*}
\]  

(3a)

Problem (3) is a convex QP with \( n + n d \) variables and \( n(n - 1) \) constraints (after removing the trivial \( i = j \) ones). Constraint (3b) imposes convexity on the observation points.

A solution of (3), that can be labelled as \( \{(f_i^*, g_i^*)\}_{i \in I_n} \), is \( F_{0,+\varepsilon} \)-interpolable by construction. In fact, once the solution is retrieved, an allowed estimator/interpolating function for \( \varphi \) at point \( x \in \mathbb{R}^d \) is given by

\[
\hat{\varphi}_n(x) = \max_{i \in I_n} \left\{ f_i^* + g_i^\top (x - x_i) \right\}.
\]  

(4)

Two comments are in order at this point:

- First, estimator (4) is non-smooth and non-strongly convex in general. While ad-hoc smoothing techniques do exist [12], one incurs a daunting trade-off between smoothing quality (in terms of low Lipschitz constant \( L \)) and estimation quality (in terms of small error \( \varepsilon \) w.r.t. the non-smooth solution). Typically, if one wants to retrieve functions of the class \( F_{\mu,L} \) with a low \( L \), one has to expect poor estimation quality \( \varepsilon \).

- Second, even though problem (3) is a convex QP and one can use off-the-shelf solvers to solve it efficiently (e.g., OSQP [13], or ECOS [14]), the number of shape constraints is still \( O(n^2) \). In addition, its computational complexity grows at least as \( O(n^3(d + 1)^3) \), making practically hard to solve problems with \( n > 200 \) and even small \( d \). However, thanks to the decomposable structure of problem (3) one can resort to first-order methods [3, 12, 15] whose computational complexity scales as \( O(n^2) \) per iteration, to tackle problems up to \( n \sim O(1000) \) in dimensions that can go up to \( d = 200 \) for the very recent [15]. Note that a computational complexity of \( O(n^2) \) is the least one can expect, given the \( O(n^2) \) constraints. Finally, partitioning methods have also been advocated [16] but not explored here.

#### B. Smooth strongly convex regression

In this paper, we propose a new set of constraints (instead of (3b)) together with an interpolation procedure for \( \hat{\varphi}_n(x) \) to enforce smoothness and strong convexity automatically. We use and adapt results from [10, 11] for this purpose. The basic idea is that smoothness and strong convexity interchange via the procedure of convex conjugation. While we leave the technical details to the above mentioned papers, we can cite the following interpolability results.

**Theorem 1:** (\( \mathcal{F}_{\mu,L} \)-interpolability) [10, Theorem 4] The set \( \{(x_i, g_i, f_i)\}_{i \in I_n} \) is \( \mathcal{F}_{\mu,L} \)-interpolable iff the following set of conditions holds for every pair of indices \( i, j \in I_n \):

\[
\begin{align*}
& f_i - f_j - g_i^\top (x_i - x_j) \geq 0, \\
& \frac{1}{L} \left( \frac{1}{L} \left| g_i - g_j \right|^2 + \mu \left| x_i - x_j \right|^2 \right) \geq -2 \mu \left| g_j - g_i \right|^2 (x_j - x_i).
\end{align*}
\]  

(5)

With this in place, we are ready to modify the constraint set (3b) to accommodate \( \varphi \in \mathcal{F}_{\mu,L} \). In particular, to solve for a \( \varphi \in \mathcal{F}_{\mu,L} \), we can leverage the convex problem:

\[
\begin{align*}
\text{minimize} & \quad f \in \mathbb{R}^n, \quad g \in \mathbb{R}^{n d} \sum_{i \in I_n} (y_i - f_i)^2 \\
\text{subject to:} & \quad f_j + g_j^\top (x_i - x_j) \leq f_i, \quad \forall i, j \in I_n.
\end{align*}
\]  

(6a)

This is a convex quadratically constrained quadratic problem (QCQP), a special case of a second-order conic program [17]. Once a solution \( \{(f_i^*, g_i^*)\}_{i \in I_n} \) is found, the following theorem describe an allowed estimation/interpolation strategy.

**Theorem 2:** For any set \( \{(x_i, g_i^*, f_i^*)\}_{i \in I_n} \) that is \( \mathcal{F}_{\mu,L} \)-interpolable, an allowed interpolating function is

\[
\hat{\varphi}_n(x) = \text{conv}(p_i(x)) + \frac{\mu}{2} \left\| x \right\|^2 _2
\]  

(7)

where

\[
p_i(x) := \frac{L - \mu}{2} \left\| x - x_i \right\|^2 _2 + (g_i^* - \mu x_i)^\top x + \left( g_i^\top x_i + f_i^* + \mu / 2 \right) \left\| x \right\|^2 _2,
\]  

(8)

and where \( \text{conv}(\cdot) \) indicates the convex hull.

**Proof:** See Appendix A.

Problem (6) together with the interpolation strategy (7) yield the promised smooth strongly convex estimator for function \( \varphi \). If we set \( \mu = 0 \) and \( L = +\infty \), we retrieve the non-smooth estimator. For \( \mu = 0 \), \( L < +\infty \), we obtain a non-strongly convex smooth estimator, while for \( \mu > 0 \) and \( L = +\infty \) a non-smooth strongly convex one.

Problem (6) is a QCQP in \( n + n d \) variables and \( n(n-1) \) constraints, which can be solved with off-the-shelf convex
solvers, e.g., ECOS [14], or MOSEK. Since the computational complexity grows at least as $O(n^3(d + 1)^3)$, similar practical limitations than the non-smooth estimator apply here. We will show how to overcome them by resorting to ADMM next.

C. Parallel implementation

Since the computational complexity of (6) could be prohibitive for practical applications, we move now to understand how one can decompose the problem into smaller parts and reduce the overall complexity. Strategies to use first-order methods (proximal methods and ADMM) on the quadratic problem (3) for non-smooth convex functions have been reported in [3], [12], [15]. In the case of (3), the problem is separable in both cost and constraints (once dualized) and a decomposition strategy is rather direct. For the case of (6), however, each constraint couples the variables $g_i$ and $g_j$ due to the quadratic term, and a decomposition is not immediate. We consider here a novel edge-based ADMM decomposition.

Consider the set of constraints of type (5) for all $i, j \in I_n$ and define the constraint graph, as a directed graph $G = (V, E)$, whose vertices are the nodes $i \in I_n$, and edges are all the combinations of $i, j; E = \{(i, j)|i \in I_n, j \in I_n, i \neq j\}$. The cardinality of $E$ is $|E| = n(n - 1)$. For each edge $e \in E$, consider its two nodes, say $i$ and $j$, and define the edge variables $\eta_{e,i} = [f_{e,i}^f, g_{e,i}^c]^T$, $\eta_{e,j} = [f_{e,j}^f, g_{e,j}^c]^T$, as well as $\xi_e = [\eta^e_{i,j}, \eta^e_{j,i}]^T \in \mathbb{R}^{2(1+d)}$. In this context, each directed edge has its own functional and derivative variables.

We use the notation $i \sim e$ to indicate that node $i$ is one of the two vertices of edge $e$, while we explicitly write $e.(i \rightarrow j)$ to indicate that edge $e$ is the directed edge with $i$ as source node and $j$ as sink node. For ease of representation, we also define the local constraint,

$$C_{e.(i \rightarrow j)} := \{f^c_e - f^c_j - g^c_{j,e}(x_i - x_j) \geq 0, \frac{1}{2(1-\mu/L)} \left( \frac{1}{L} \left| g^c_{i,e} - g^c_{j,e} \right|^2 + \mu \|x_i - x_j\|^2 \right) - \frac{\mu}{L} (g^c_{i,e} - g^c_{j,e})^T (x_j - x_i) \right\}. \tag{9}$$

We now split problem (6) at the edges, so that each variable $\eta^e_{i,j}$ containing node $i$, is different for each edge $e$ having node $i$ has one of its vertices. Then we enforce equality of all the node variables via the supporting vector $z_i \in \mathbb{R}^1 + d$ for each $i$, $z = [z^1_1, \ldots, z^1_n]^T \in \mathbb{R}^{n + nd}$. With this philosophy, problem (6) can be rewritten in the equivalent form:

\[
\begin{align*}
\text{minimize} & \quad \xi \in \mathbb{R}^{2|E|(1+d)} \quad \text{subject to} \quad C_{e.(i \rightarrow j)} \quad \forall e \in E \quad \eta_{e,i} = z_i \quad \forall e \in E \quad i \sim e \quad (10c) \\
\end{align*}
\]

It is key that the constraint $C_{e.(i \rightarrow j)}$ is present only once for each directed edge.

The above problem can now be tackled with ADMM. We leave the derivations out, since standard, and report only the final result. Start with some initialization for the auxiliary variable $z$ and initialize the scaled dual variables of the constraints (10c) as $\lambda = [\lambda^1_1, \lambda^2_1, \ldots] \in \mathbb{R}^{|E|(1+d)} = 0$. Set the penalization $\rho > 0$. Then at each step,

1) Solve the edge QCQP for each edge $e \in E$:

$$\begin{align*}
\left[ \eta^e_{i,j}, \eta^e_{j,i} \right]^T = & \quad \xi^e = \arg\min_{\xi \in \mathbb{R}^{2|E|(1+d)}} \left\{ \frac{1}{2n} \sum_{i \in e} (y_i - f^c_{i,e})^2 + \\
& \quad + \sum_{i \sim e} \frac{\rho}{2} \| \eta_{e,i} - z_i + \lambda_{e,i} \|_2^2 \right\} \quad (11)
\end{align*}$$

2) Update the $z_i$ variables for each node $i \in V$:

$$z^+ = \frac{1}{2n} \sum_{e \sim i} \eta^+_{e,i} \quad (12)$$

3) Update the $\lambda_e$ variables $\forall e \in E$, and node $i \sim e$:

$$\lambda^+_{e,i} = \lambda_{e,i} + (\eta^+_{e,i} - z^-_i) \quad (13)$$

4) Set $\xi_e = \xi^+_{e,i}$ for all $e \in E$, $z_i = z^+_i$ for all $i \in V$, and $\lambda_e = \lambda^+_e$ for all $e \in E$, $i \sim e$, and go to 1.

Convergence of the above procedure to a minimizer of the original problem (6) is assured by the standard ADMM convergence results [15]. In practice, one stops the ADMM iterations after a specified error criterion has been met, or a number of iterations has been reached. In this case, we consider as approximate solution to our problem, the final $z$ vector that ADMM yields (we let $\hat{f}_i$ be the near-optimal $f$ values and $\hat{g}_i$ be the near-optimal $g$ values) and we use the approximate interpolating function [see Th. 2].

$$\hat{\varphi}_n(x) = \text{conv}(\hat{p}_i(x)) + \frac{\mu}{2} \| x \|^2, \quad (14)$$

$$\hat{p}_i(x) := L - \eta_{e,i} + \frac{\mu}{2} \left( x - x_i \right) + (\hat{f}_i - \mu x_i)^T x +$$

$$- \hat{g}_i^T x + \hat{f}_i + \mu / 2 \| x \|^2. \quad (15)$$

Note that function (14) is still a $L$-smooth $\mu$-strongly convex function by construction, just the ADMM approximate solution $z$ is only approximately $F_{\mu, L}$-interpolatable, which means that $\hat{f}_i \approx \vec{\varphi}_n(x_i)$ and $\hat{g}_i \approx \vec{\varphi}_n(x_i)$ instead of equality (meaning that function (14) is only approximately minimizing the LS metric). In practice, we notice that this approximation delivers good feasible results for small errors.

D. Solving the local dual

Solving (11) in the primal form is still too computational complex (especially if it has to be done for each edge for each iteration of the ADMM algorithm). However, since there is only one scalar constraint, the dual is mono-dimensional and easier to solve with a few iterations of a projected Newton’s method. To do this, we write (11) in the standard form:

$$\begin{align*}
\text{minimize} & \quad \xi \in \mathbb{R}^{2|E|(1+d)} \quad \text{subject to} \quad \xi^e P_0 \xi + q_0^e \xi + r_0 \leq 0 \quad (16a) \\
\end{align*}$$

\[
\begin{align*}
\text{where} & \quad P_0 \in \mathbb{S}^{2(1+d)}_+ \quad P_1 \in \mathbb{S}^{2(1+d)}_+ \quad q_0 \in \mathbb{R}^{2(1+d)} \quad q_1 \in \mathbb{R}^{2(1+d)} \quad r_0 \in \mathbb{R} \quad \text{and} \quad r_1 \in \mathbb{R} \quad \text{are properly defined matrices,}
\end{align*}
\]
vectors, and scalars to match [16] to the original [11]. Define the Lagrangian function
\[
\mathcal{L}(\nu) := \xi_1^T P_0 \xi_e + q_0^T \xi_e + r_0 + \nu(\xi_1^T P_1 \xi_e + q_1^T \xi_e + r_1),
\]
for the dual variable \( \nu \geq 0 \). Note that \( P_0 \) is positive definite by construction (due to \( \rho > 0 \)), and therefore setting \( P_\nu := P_0 + \nu P_1 \), the inverse \( P_\nu^{-1} \) exists and it is well-defined. Set also \( q_\nu := q_0 + \nu q_1 \). The dual problem of [16] is:
\[
\text{maximize } \phi(\nu) := -\frac{1}{4} q_\nu^T P_\nu^{-1} q_\nu + \nu r_1 + r_0
\]
whose solution can be found with standard projected Newton’s method.\(^2\) Once the dual problem is solved and the unique dual solution \( \nu^* \) is found, the unique primal solution of [16] can be retrieved as
\[
\xi_e^* = -\frac{1}{2} P_\nu^{-1} q_\nu^*.
\]

Putting things together, the ADMM procedure (1-4) with a Newton’s method to solve the mono-dimensional local dual problems [18] has computational complexity of \( n(n-1) \times O((d+1)^3) \) (where the \( O((d+1)^3) \) term comes from the computation of the inverse \( P_\nu^{-1} \)) and for \( d \ll n \), the leading error is \( O(n^2) \).

IV. NUMERICAL EVALUATION

We evaluate the presented algorithms in terms of computational time and mean square error on a toy problem. The aim is to evaluate scalability and quality in a simple one-dimensional setting. A more detailed numerical investigation is deferred to future research efforts. All the computations are performed using Python 3.6, on a 2.7 GHz Dual-Core Intel Core i5 laptop with 8GB of RAM. We use CVXPY [19] for solving the non-smooth and smooth problems. Internally, the QPs are solved with OSQP, while the QCQPs with ECOS.

The true generating function is \( \varphi(x) = x^2 \) over \( x \in [-1, 1] \). Function \( \varphi \) is 2-smooth and 2-convexly. The observations are generated by adding noise drawn from \( \mathcal{N}(0, \sigma^2) \).

We run (i) the non-smooth problem with QP [5] and estimator [4]; (ii) the \( L \)-smooth \( \mu \)-convexly strongly convex problem with QCQP [6] and estimator [7]; (iii) the edge-based ADMM to solve the smooth problem with local problems [11] and approximate estimator [14], with \( \rho = 1/n \) and initial \( z \) specified running a Gaussian process estimator [20], which offers a computationally cheap smooth (but in general non-convex) first approximation. For the Gaussian process we use a square exponential kernel. For the ADMM we solve the local problems resorting to their duals and employing a projected

\(^2\) For the sake of completeness, the gradient of the dual function is
\[
\frac{d\phi}{d\nu} = -\frac{1}{4} q_\nu^T P_\nu^{-1} q_\nu + q_\nu^T P_\nu^{-1} P_1 P_\nu^{-1} q_\nu + r_1,
\]
while the Hessian is
\[
\frac{d^2\phi}{d\nu^2} = -\frac{1}{2} q_\nu^T P_\nu^{-1} q_\nu + q_\nu^T P_\nu^{-1} P_1 P_\nu^{-1} q_\nu - \frac{1}{2} q_\nu^T P_\nu^{-1} P_1 P_\nu^{-1} P_1 P_\nu^{-1} q_\nu.
\]

Newton’s method with back-tracking. The stopping criterion for the ADMM has been selected as
\[
\varepsilon = \max\{\|n_{\text{true}}^+_e z^+_s\}_{e \in E, i=1}^s, \|z^+ - z\|_X\}. \tag{20}
\]

In Figure 1 we report the three methods (non-smooth, smooth, and ADMM) for the task of estimating the given function \( \varphi(x) \) with an increasing amount of observations. Here the observation noise standard deviation is set to \( \sigma = 0.2 \), while \( \varepsilon = 0.01 \). We also set \( \mu = 1 \) and \( L = 5 \). We display the estimated function \( \hat{\varphi}(x) \) with a continuous blue line, while the true function is a dashed orange line. Using grey dots, we represent the observations. As we notice, non-smooth and smooth estimators achieve the desired estimation task with higher accuracy the more observations are present. ADMM also does well (even with a moderate error \( \varepsilon \)).

We run simulations for different \( n \)’s, considering \( \sigma = 0.1 \), and two versions of ADMM, one with \( \varepsilon_1 = 0.03 \) and the other with \( \varepsilon_2 = 0.01 \). (All the other parameters have been left the same as the ones in Figure 1) All the results are averaged over 10 realizations of the observations for the centralized problems, and 25 for the parallel methods (which are more dependent on the realization for the number of iterations).

Figure 2 represents the computational times of the various methods. As we see, the smooth method is the slowest one (as \( n \) increases), as expected. The ADMM methods run slower than the other methods at the beginning due to the non-accurate initialization (and therefore due to the need for more iterations to reach the specified stopping criterion), while as \( n \) increases, they run the fastest (since the initialization becomes better and better).

Figure 3 represents an error metric defined as
\[
E_{\hat{\varphi}_n} = \frac{1}{n_s} \sum_{s \in I_{n_s}} (\hat{\varphi}_n(y_s) - \varphi(y_s))^2 \tag{21}
\]
This metric is defined over a finer equi-spaced sampling \( y_s, s = \{1, \ldots, n_s\} := I_{n_s}, \text{ over } [-1, 1] \) with \( n_s = 1000 \). The
idea is to capture the error in the estimator \( \hat{\varphi}_n \), rather than just the mean square error on the observation points. As we see and expected, the smooth estimator is better than the non-smooth one. ADMM delivers estimates of comparable accuracy than the smooth one. In the figure, we also display an \( O(n^{-2/5}) \) line, which is the typical convergence rate of non-parametric LSE for \( d = 1 \) \([12], [21]–[23]\), in the mean-square-error-on-the-observation-points sense.

What is interesting here is that the metric \( E_{\hat{\varphi}_n} \) is built on the estimated \( \hat{\varphi}_n \) which for ADMM is the approximate (14). Because of this, the ADMM’s estimator can be better than the smooth one (especially in low observation settings) This is perhaps quite interesting and will be further explored in the future.

V. CONCLUSION

We have proposed a method to perform smooth strongly convex regression in a non-parametric least squares sense. The method relies on the solution of a convex quadratically constrained quadratic program. We have discussed computational complexity and offered a first-order alternative based on ADMM to lessen the computational load to a tight \( O(n^2) \) for \( n \) noisy observations of the true function.

APPENDIX

A. Proof of Theorem 2

The theorem follows from the discussion in \([10]\) Remark 2] with some extra computations. For the sake of completeness, we report it here. In particular, take any set \( \{(\hat{x}_i, \hat{g}_i, \hat{f}_i)\}_{i \in I_n} \). If this set is \( \mathcal{F}_{1/(L-\mu)} \)-interpolable, then an allowed interpolating function is \([10]\) Remark 2):

\[
h(\hat{x}) = \max_i \{h_i(\hat{x})\},
\]

with

\[
h_i(\hat{x}) = \hat{f}_i + \hat{g}_i^T(\hat{x} - \hat{x}_i) + \frac{1}{2(L - \mu)}\|\hat{x} - \hat{x}_i\|^2_2.
\]

As in \([10]\) Remark 2] by convex conjugation and curvature addition, an \( \mathcal{F}_{\mu,L} \) interpolating function of \( \{(x_i, g_i, f_i)\}_{i \in I_n} \) is:

\[
\hat{\varphi}_n(x) = \text{conv}(h^*_i(x)) + \frac{\mu}{2}\|x\|^2_2.
\]

Computing \( h^*_i(x) \) can be done in the standard way,

\[
h^*_i(x) = \sup_{\hat{x}} \{x^T \hat{x} - h_i(\hat{x})\} = 
\sup_{\hat{x}} \{x^T \hat{x} - (\hat{f}_i + \hat{g}_i^T(\hat{x} - \hat{x}_i)) + \frac{1}{2(L - \mu)}\|\hat{x} - \hat{x}_i\|^2_2\}. \quad (24)
\]

For the inner optimization problem (sup), by first-order optimality conditions,

\[
-x + \hat{g}_i + \frac{1}{L-\mu}(\hat{x} - \hat{x}_i) = 0 \iff \hat{x} = \hat{x}_i + (L-\mu)(x - \hat{g}_i),
\]

which substituted back into (24) yields

\[
h^*_i(x) = \frac{L - \mu}{2} \|x - \hat{g}_i\|^2_2 + x^T \hat{x}_i - \hat{f}_i. \quad (26)
\]

From \([10]\) Theorem 4(c)] follows that

\[
\{(\hat{x}_i, \hat{g}_i, \hat{f}_i)\}_{i \in I_n} = \{(\hat{g}_i - \mu \hat{x}_i, \hat{x}_i, \hat{x}_i^T \hat{g}_i - \hat{f}_i - \mu/2\|\hat{x}_i\|^2_2)\}_{i \in I_n}.
\]

Operating the substitutions in (26) and calling \( p_i(x) := h^*_i(x) \), the thesis follows.

REFERENCES

[1] R. J. Samworth and B. Sen, “Editorial: Nonparametric Inference Under Shape Constraints,” *Statistical Science*, vol. 33, no. 4, 2018.
[2] P. Groeneboom, G. Jongbloed, and J. A. Wellner, “Estimation of a Convex Function: Characterizations and Asymptotic Theory,” *The Annals of Statistics*, vol. 29, no. 6, pp. 1653 – 1698, 2001.
[3] N. S. Aybat and Z. Wang, “A Parallel Method for Large Scale Convex Regression Problems,” in *Proceedings of the IEEE Conference in Decision and Control*, 2014.
[4] R. F. Meyer and J. W. Pratt, “The Consistent Assessment and Fairing of Preference Functions,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 3, pp. 270 – 278, 1968.
[5] A. Simonetto, E. Dall’Anese, J. Monteil, and A. Bernstein, “Personalized Optimization with User’s Feedback,” *arXiv*: 1905.00775, 2019.
[6] X. Wang and J. O. Berger, “Estimating Shape Constrained Functions Using Gaussian Processes,” SIAM/ASA Journal on Uncertainty Quantification, vol. 4, no. 1, pp. 1 – 25, 2016.

[7] A. L. Dontchev, H. Qi, and L. Qi, “Quadratic Convergence of Newton’s Method for Convex Interpolation and Smoothing,” Constructive Approximation, vol. 19, pp. 123 – 143, 2003.

[8] M. Birke and H. Dette, “Estimating a Convex Function in Nonparametric Regression,” Scandinavian Journal of Statistics, vol. 34, no. 2, pp. 384 – 404, 2007.

[9] E. Dall’Anese, A. Simonetto, S. Becker, and L. Madden, “Optimization and Learning with Information Streams: Time-varying Algorithms and Applications,” Signal Processing Magazine (to appear), May 2020.

[10] A. Taylor, J. Hendrickx, and F. Glineur, “Smooth Strongly Convex Interpolation and Exact Worst-case Performance of First-order Methods,” Mathematical Programming, vol. 161, no. 1, pp. 307 – 345, 2017.

[11] A. Taylor, “Convex Interpolation and Performance Estimation of First-order Methods for Convex Optimization,” Ph.D. dissertation, Université catholique Louvain, Belgium, January 2017.

[12] R. Mazumder, A. Choudhury, G. Iyengar, and B. Sen, “A Computational Framework for Multivariate Convex Regression and Its Variants,” Journal of the American Statistical Association, vol. 114, no. 525, pp. 318–331, 2019.

[13] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: An operator splitting solver for quadratic programs,” arXiv: 1711.08013, 2017.

[14] A. Domahidi, E. Chu, and S. Boyd, “ECOS: An SOCP Solver for Embedded Systems,” in Proceedings of the ECC, 2013.

[15] M. Lin, D. Sun, and K.-C. Toh, “Efficient algorithms for multivariate shape-constrained convex regression problems,” arXiv: 2002.11410, 2020.

[16] L. A. Hannah and D. B. Dunson, “Multivariate Convex Regression with Adaptive Partitioning,” JMLR, vol. 14, pp. 3261 – 3294, 2013.

[17] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” Foundations and Trends® in Machine Learning, vol. 3, no. 1, pp. 1 – 122, 2011.

[19] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” JMLR, vol. 17, no. 83, 2016.

[20] C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning. Cambridge, MA, US: The MIT Press, 2006.

[21] E. Seijo and B. Sen, “Nonparametric Least Squares Estimation of a Multivariate Convex Regression Function,” The Annals of Statistics, vol. 39, no. 3, pp. 1633 – 1657, 2011.

[22] E. Lim and P. W. Glynn, “Consistency of Multidimensional Convex Regression,” Operation Research, vol. 60, no. 1, pp. 196 – 208, 2012.

[23] J. Blanchet, P. W. Glynn, J. Yan, and Z. Zhou, “Multivariate Distributionally Robust Convex Regression under Absolute Error Loss,” in Proceedings of NeurIPS, 2019.