

ON A SCALABLE NONPARAMETRIC DENOISING OF TIME SERIES SIGNALS

LUKÁŠ POSPÍŠIL¹, PATRICK GAGLIARDINI¹, WILLIAM SAWYER²,
AND ILLIA HORENKO¹

ABSTRACT. Denoising and filtering of time series signals is a problem emerging in many areas of computational science. Here we demonstrate how the non-parametric computational methodology called Finite Element Method of time series analysis with H1 regularization can be extended for denoising of very long and noisy time series signals. The main computational bottleneck is induced by the cost of the inner Quadratic programming problem. Analyzing the solvability and utilizing the problem structure, we suggest an adapted version of the Spectral Projected Gradient method (SPG-QP) to resolve the problem. This approach increases the granularity of parallelization, making the proposed methodology highly suitable for Graphics Processing Unit (GPU) computing. We demonstrate the scalability of our open-source implementation based on PETSc for the Piz Daint supercomputer of the Swiss Supercomputing Centre CSCS, by solving large-scaled data denoising problems and comparing their computational scaling and performance to the performance of the standard denoising methods.

Keywords: time series analysis, quadratic programming, SPG-QP, regularization

1. INTRODUCTION

Time series signals (i.e., data measured in intervals over a period of time) are typical for many practical areas such as econometrics (e.g., movement of stock prices [21]), climatology (e.g., temperature changes [42]) or molecular dynamics (e.g., in conformational changes of the molecule [23]). The analysis of time series signals aims to extract meaningful characteristics and understand the process which has generated those data. Such an analysis is the key ingredient in the forecasting the process beyond the observed and measured time. However, one of the main difficulties in the analysis of real measurement is the presence of the measurement/experimental noise. Additionally, almost exponentially growing amount of collected data in many practical applications requires a development of better and faster data-driven denoising, modeling and classification tools suitable for High Performance Computing (HPC).

Suppose we observed time series signal $x_t \in \mathbb{R}^n, t = 1, \dots, T$ (where n is the data dimension and T is the length of time series) and those data are appropriately described by the model function $\mu(t, \Theta)$ with parameters $\Theta \in \mathbb{R}^m$ and an additive

¹ UNIVERSITÀ DELLA SVIZZERA ITALIANA, VIA GIUSEPPE BUFFI 13, 6900 LUGANO, SWITZERLAND

² SWISS NATIONAL SUPERCOMPUTING CENTRE, VIA TREVANO 131, 6900 LUGANO, SWITZERLAND

E-mail addresses: illia.horenko@usi.ch (corresponding author).

Date: June 20, 2017.

noise ε , i.e.

$$(1) \quad x_t = \mu(t, \Theta) + \varepsilon_t, \quad t = 1, \dots, T,$$

where $\{\varepsilon_t\}$ is a family of independent and identically distributed (i.i.d.) random variables with zero expectation. The explicit model function $\mu(t, \Theta)$ is chosen a priori based on our knowledge about the particular application. In general, the aim of modeling process is to determine optimal parameters $\bar{\Theta}$ such that the observed data x_t are described by (1) in the most optimal way, for example using maximum likelihood principle (MLE) or minimizing mean-square error. Finally, the denoised signal can be obtain as an output of (1) with known $\bar{\Theta}$ and without the presence of (eliminated) noise term ε_t .

In the first part of introduction, we shortly review the general classification of time series modeling methodologies based on the choice of μ . In the second part, we investigate methods from the point of computational cost and highlight the importance of developing the optimization algorithms for its effective solution. The final part of introduction presents Finite Element Method of time series analysis with H1 regularization (FEM-H1) methodology used in the approach presented in this paper.

1.1. General model classification. In the simple case, the form of the model function μ is known and, for instance, it can be expressed as some priori defined function depended on time. If the dimension of the underlying parameters Θ is finite, then this method is called *parametric*.

For example, in the case of a linear regression

$$(2) \quad \mu(t, \theta_0, \theta_1) = \theta_1 t + \theta_0$$

the unknown model parameters $\theta_0, \theta_1 \in \mathbb{R}$ can be found using MLE (or minimizing least-square error) as a solution of an appropriate optimization problem

$$(3) \quad [\bar{\theta}_0, \bar{\theta}_1] = \arg \min_{\theta_0, \theta_1} \sum_{t=1}^T \|x_t - \mu(t, \theta_0, \theta_1)\|^2.$$

The recovered signal is obtained as values of $\mu(t, \bar{\theta}_0, \bar{\theta}_1), t = 1, \dots, T$. However, the model used (2) is valid only if the original data was generated by a linear model - and if no non-linear effects had a significant impact on the underlying process.

Another example of parametric methods are Hidden Markov Models (HMM). Hereby, it is a priori assumed that there exist regimes such that data in each regime is distributed according to some explicit parametric distribution from a known and fixed family of parametric distributions (e.g., Gaussian, Poisson, etc.). The aim is then to search for an optimal regime-switching homogeneous Markov process represented by unknown components of transition matrix and initial states [5].

In general, parametric methods are usually based on rather strong explicit assumptions about the problem structure. These assumptions help to create a tractable finite-dimensional formulation of the problem, which can be solved analytically or efficiently numerically. In general, the more model assumptions are imposed, the less general the model is - and the simpler is a numerical optimization problem to be solved. On the other hand, imposing a misspecified parametric structure leads to models that incorrectly describe the problem under consideration.

The way how to avoid the (possibly inappropriate) restrictive a priori explicit parametric assumptions about the dynamics of the model parameters is to use

nonparametric models. In the case of nonparametric models, the dimension of the underlying parameters Θ is infinite and we assume that optimal parameters are represented as functions from a priori restricted class of feasible functions. However, the larger generality and complexity of the used nonparametric models also imposes a much higher computational cost on resulting infinite-dimensional optimization problem. Therefore, the nonparametric models are much more challenging in numerical implementation and execution.

An example of non-parametric method is a Generalized Additive Model (GAM) [29]. In comparison to linear regression (2), (3), the model function μ in GAM can be defined as an arbitrary, nonlinear and nonparametric smooth function from Sobolev space on an interval $[t_1, t_T] = [1, T]$

$$(4) \quad W^2([t_1, t_T]) = \{\mu(\cdot) \in \mathcal{C}^{(2)}([t_1, t_T]) : \int_{t_1}^{t_T} [\mu''(\hat{t})]^2 d\hat{t} < \infty\}.$$

The optimal (i.e. sufficiently smooth) modeling function is then given by solving the optimization problem

$$(5) \quad \bar{\mu} = \arg \min \sum_{t=1}^T (x_t - \mu(t)) + \lambda \int_{t_1}^{t_T} [\mu''(\hat{t})]^2 d\hat{t}.$$

Here $\lambda > 0$ represents the regularization parameter which has to be estimated [51].

1.2. Computational cost. The choice of the "most optimal" tools in every particular application is not a trivial task and is triggered by the counter-play of many factors [39], most of all by the the following two factors: (i) the amount of bias that is introduced by the analysis method (for example, coming from the eventually-wrong a priori assumptions about the linearity, Gaussianity and homogeneity of the underlying processes) and (ii) the computational scalability of different data-driven algorithms - as well as the possibility to deploy these algorithms in a High Performance Computing (HPC) setting - to be able to process the data *en masse*.

Experience shows that the analysis algorithms that introduce the highest-potential bias (e.g., standard linear Fourier filtering methods based on the Fast Fourier Transforms (FFT) or parametric Bayesian methods like HMM with Gaussian or Poisson outputs and a time-homogenous Markov model assumption [39, 40, 49]) demonstrate the best HPC scaling performances whereas the nonlinear and non-Gaussian approaches like Convolutional Neuronal Networks (CNNs) need more communication and scale worse - so only the deployment of massively-parallel GPU architectures helped to reach the scale-up that was necessary to apply these methods to large realistic problems [15, 48].

From the mathematical perspective, essentially all of the data analysis and classification methods currently available in the standard analysis packages can be formulated as the numerical algorithms for solutions of large optimization problems.

To give some examples, the standard Fourier-, wavelet- and kernel-filtering algorithms for denoising of the signals $x_t \in \mathbb{R}^n$ - as well as the parameter identification methods for Support Vector Machine (SVM) classification, linear discriminant analysis and linear autoregressive models (AR) - can be formulated and implemented as

solution algorithms for the same type of the unconstrained quadratic minimization problem (QP)

$$\bar{y} = \arg \min_y \|x - \Phi y\|_2,$$

where $\|\cdot\|_2$ denotes Euclidean norm, $y \in \mathbb{R}^r$ (r is typically much less than n), and $\Phi \in \mathbb{R}^{n \times r}$ is a known filtering matrix. Variational methods (like the regularized kernel filtering, compressed sensing and regularized model inference) [12, 52, 50] can be obtained by adding the regularizing inequality constraints to this convex problem: for example, a very popular compressed sensing algorithm in a dual formulation can be obtained by adding the linear inequality constraint $\|y\|_1 \leq \epsilon$ (where $\|\cdot\|_1$ denotes L1 norm and ϵ is some a priori fixed "sparsity" parameter) to the above unconstrained QP. Neuronal networks can be straightforwardly approached as parametric non-convex optimization problems of the type

$$\bar{y} = \arg \min_y \|x - \Phi(y)\|_2,$$

where Φ is some a priori fixed non-linear operator characterizing the network topology and y are unknown network parameters.

Finally, the nonstationary and nonparametric denoising and modeling methods based on regularized nonconvex clustering algorithms (like the FEM-H1 methodology developed in [24]) can be implemented as the solution of minimization problem of regularized clustering functional with equality and inequality constraints. In following we shortly review this approach.

1.3. Nonparametric nonstationary FEM-H1 methodology. We will consider observed data as time series $x_t \in \mathbb{R}^n, t = 1, \dots, T$. Our aim is to find coefficients $\Theta(t)$ of some model function $\mu(t, \Theta)$ such that this function in some sense fits the data in the best way. This fitting condition (i.e., the distance between observed data and values of the model function) is measured by a metric g , which can be for example defined by means of the function

$$(6) \quad g(x_t, \Theta(t)) = (x_t, \mathbb{E}[\mu(t, \Theta(t))])^2.$$

Therefore, the most appropriate parameters $\Theta(t)$ can be obtained by solving the variational problem

$$(7) \quad \bar{\Theta} = \arg \min_{\Theta(\cdot) \in \Omega_\Theta} L(\Theta), \quad L(\Theta) = \sum_{t=1}^T g(x_t, \Theta(t)),$$

where L refers to a model distance function and Ω_Θ represents the space of all feasible parameters of parameter functions $\Theta(\cdot)$ for the considered model. However, this problem is ill-posed if only one sequence of data $\{x_1, \dots, x_T\}$ is available (this is typical situation for many practical applications, e.g., computational finance or climatology, where only one historical sequence of data is available for each particular time series). One option of regularizing this problem and making it well-posed is based on the clustering of $\Theta(t)$, i.e., one can assume that there exist K different stationary parameters $\Theta = [\Theta_1, \dots, \Theta_K]$ such that the fitness function (6) can be expressed as a convex combination

$$g(x_t, \Theta(t)) = \sum_{i=1}^K \gamma_i(t) g(x_t, \Theta_i),$$

where $\gamma_k(t) \in \{1, \dots, T\} \rightarrow [0, 1], k = 1, \dots, K$ are so-called model indicator functions, see [41] and [33]. These functions define the *activeness* of appropriate i -th cluster at a given time t ; if $\gamma_i(t) = 1$, then the data are modeled by i -th model in time t . These properties can be written in form of constraints

$$(8) \quad \sum_{i=1}^K \gamma_i(t) = 1, \forall t, \quad 0 \leq \gamma_i(t) \leq 1, \forall t, i.$$

Hence, model indicator functions could be considered as switching functions between individual models on clusters. Additionally, one can incorporate additional information about observed process, for example by assuming that switching between clusters is in some sense slower than the changes of the signal caused by the presence of the modeling error or the noise in data. In our notation, this approach means that model indicator functions γ_i are smooth in some appropriately-chosen function space. For example, one can enforce the smoothness in \mathcal{H}_1 -space by introducing the Tikhonov-based penalization term

$$(9) \quad \begin{aligned} [\bar{\Theta}, \bar{\Gamma}] &= \arg \min_{\substack{\Theta \in \Omega_{\Theta} \\ \Gamma \in \Omega_{\Gamma}}} L_{\varepsilon}(\Theta, \Gamma), \\ L_{\varepsilon}(\Theta, \Gamma) &= \sum_{t=1}^T \sum_{i=1}^K \gamma_i(t) g(x_t, \Theta_i) + \varepsilon^2 \sum_{i=1}^K \sum_{t=2}^T (\gamma_i(t-1) - \gamma_i(t))^2, \end{aligned}$$

where Ω_{Γ} is a feasible set defined by conditions (8) and ε^2 denotes the regularization parameter.

This methodology was called FEM-H1 and it was introduced and developed in [31, 32, 35, 33, 34, 36, 37]. For the unified and simplified derivation of the method, as well as its relation to classical methods of unsupervised learning, please see [41]. Moreover, the method was extended for spatial regularization using the network information in the graph-based form of the regularization matrix, see [24]. In this case, the only difference appears in the formulation of the smoothing term.

From a numerical point of view, the problem (9) can be solved as a sequence of split optimization problems, see Algorithm 1.

Algorithm 1. Outer optimization algorithm.

Set feasible initial approximation $\Gamma^0 \in \Omega_{\Gamma}$

while $\|L_{\varepsilon}(\Theta^k, \Gamma^k) - L_{\varepsilon}(\Theta^{k-1}, \Gamma^{k-1})\| \geq \varepsilon$

solve $\Theta^k = \arg \min_{\Theta \in \Omega_{\Theta}} L_{\varepsilon}(\Theta, \Gamma^{k-1})$ (with fixed Γ^{k-1})

solve $\Gamma^k = \arg \min_{\Gamma \in \Omega_{\Gamma}} L_{\varepsilon}(\Theta^k, \Gamma)$ (with fixed Θ^k)

$k = k + 1$

endwhile

Return approximation of model parameters Θ^k and approximation of model indicator functions Γ^k .

Please notice that the first optimization problem in Algorithm 1 is strongly connected to the type of modeling problem and model used. However, if we are able to solve the stationary variant of the problem, then this clustered problem includes only one modification represented by the multiplication by constant coefficients $\gamma_i(t)$. Beyond that, this problem can be reduced into K completely independent

problems; for each cluster we are solving the stationary problem. And also the size of this problem is typically small since we suppose that the number of clusters is reasonably small.

One of the main challenges in applying this framework to the analysis of real time series data (e.g., in computational finance, climatology, or neuroscience) is induced by a high computational cost of the second optimization sub-problem in this algorithm. Due to this limitation, published applications of these methods are confined to relatively-small data sets [31, 32, 35, 33, 34, 36, 37]. This second optimization sub-problem is completely independent of the type of the application, i.e., independent of the choice of fitness function (6). Nevertheless, the size of this problem is given by KT and cannot be separated because of the conditions (8) and the form of regularization term in (9). In optimization theory, the problem of this form (quadratic cost function with a feasible set formed by linear equality and inequality constraints) is called quadratic programming problem (QP), see [44] and [18]. Therefore, if we develop the efficient solver to deal with this main computational bottleneck of the FEM-H1 data analysis framework, then we will be able to apply the framework to very large realistic data sets from different application areas (finance, image processing, bioinformatics, etc.). A central goal of this paper is to provide an algorithmic solution to this fundamental problem of the FEM-H1 framework.

Therefore, we will subsequently concentrate on the HPC solution of the problem of unknown model indicator functions Γ . For practical reasons, we define a column vector with all (unknown) model indicator functions by

$$\gamma := [\gamma_1, \dots, \gamma_K] \in \mathbb{R}^{KT}$$

and problem (9) for constant Θ can be written in the form of block-structured QP problem

$$(10) \quad \begin{aligned} \bar{\gamma} &:= \arg \min_{\gamma \in \Omega_\Gamma} L_\epsilon(\gamma) \\ \gamma &:= [\gamma_1, \dots, \gamma_K] \in \mathbb{R}^{KT}, \\ \gamma_i &:= [\gamma_i(1), \dots, \gamma_i(T)] \in \mathbb{R}^T \\ L_\epsilon(\gamma) &:= \frac{1}{T} b_\Theta^\top \gamma + \frac{\epsilon^2}{T} \gamma^\top H \gamma, \\ \Omega_\Gamma &:= \{\gamma \in \mathbb{R}^{KT} : \gamma \geq 0 \wedge \sum_{k=1}^K \gamma_k(t) = 1, \forall t = 1, \dots, T\} \end{aligned}$$

where $H \in \mathbb{R}^{KT \times KT}$ is a block-diagonal matrix, whose blocks $H_i \in \mathbb{R}^{T \times T}$ are formed by Laplace matrices and

$$b_\Theta := [g(x_t, \Theta_1), \dots, g(x_t, \Theta_K)] \in \mathbb{R}^{KT}$$

denotes the column block-structured vector of modeling errors [33]. Notice that we scaled the cost function by positive coefficient $1/T$ to control the scale of the function values for the cases with large T .

The paper is organized as follows. In Section 2, we examine the solvability and properties of the QP problem (10). Afterwards in Section 3, we present the modification of the spectral projected gradient method for QP problems suitable for a HPC implementation and discuss its advantages in comparison with other standard algorithms. In our project, we are interested in the HPC implementation of FEM-H1 methodology to be able to deal with very long time series. From the beginning, we consider a situation when even the input data cannot be stored

and operated on one computational node, therefore the distributed layout of the vectors and matrices has to be introduced and considered during the whole solution process. In Section 4 we briefly introduce our parallel implementation approach. Section 5 presents the performance of our algorithm on a data denoising problem, which is constructed to mimic the main features (like the very high noise-to-signal ratios, non-Gaussianity of the noise) that are typical for time series from practical applications. In contrast to the analysis of the “real life” practical data (where the underlying “true signal” is hidden in the noise and is not known a priori), analysis of this test data that we propose allows for a direct comparison of the introduced method to different standard denoising algorithms. It also allows the assessment of the denoising performance of the methods for various ratios of signal-to-noise — an assessment that can not be achieved for the “real life” data. We also present the scalability results of our implementation on the Piz Daint supercomputer [1]. In this section, we show the efficiency of FEM-H1 methodology in comparison with other standard denoising approaches. We show that our method outperforms other standard denoising methods in the terms of the denoising quality in the situations when the signal-to-noise ratio of the data becomes small.

Finally, Section 6 concludes the paper and presents some ideas for our future research.

2. SOLVABILITY OF INNER QP

For the simplicity of our analysis, we rewrite the problem (10) using the convenient notation

$$(11) \quad \min_{x \in \Omega} f(x), \quad f(x) := \frac{1}{2} x^\top A x - b^\top x,$$

where $A := \frac{\epsilon^2}{T} H \in \mathbb{R}^{KT \times KT}$ is a symmetric positive semidefinite (SPS) Hessian matrix of a quadratic cost function $f : \mathbb{R}^{KT} \rightarrow \mathbb{R}$ and $b := -\frac{1}{T} b_\Theta^\top$ is the so-called right hand side vector. This name came from the necessary optimality condition for the unconstrained problem $\Omega = \mathbb{R}^{KT}$, which is given by

$$(12) \quad \nabla f(x) = Ax - b = 0 \Leftrightarrow Ax = b,$$

see [44], [11], and [18].

Since Hessian matrix A of the quadratic function f is SPS, this cost function is continuous and (not strictly) convex. Moreover, the null space (kernel) is given by

$$(13) \quad \text{Ker } A = \text{span}\{[\mathbf{1}^\top, \mathbf{0}, \dots, \mathbf{0}]^\top, [\mathbf{0}, \mathbf{1}^\top, \mathbf{0}, \dots, \mathbf{0}]^\top, \dots, [\mathbf{0}, \dots, \mathbf{0}, \mathbf{1}^\top]^\top\} \subset \mathbb{R}^{KT},$$

where we denote $\mathbf{1} := [1, \dots, 1]^\top \in \mathbb{R}^T$ and $\mathbf{0} := [0, \dots, 0]^\top \in \mathbb{R}^T$.

The feasible set $\Omega \subset \mathbb{R}^{KT}$ is a (non-empty) bounded closed convex set and it can be equivalently defined by

$$\Omega = \{\gamma \in \mathbb{R}^{KT} : \gamma \geq 0 \wedge B\gamma = c\},$$

where $B := [I, \dots, I] \in \mathbb{R}^{T \times KT}$, $c := \mathbf{1}$, and $I \in \mathbb{R}^{T \times T}$ denotes the identity matrix. Using this notation, we can easily conclude that the optimization problem (11) is a QP problem with the SPS Hessian matrix, linear equality constraints and bound constraints

$$(14) \quad \min \frac{1}{2} x^\top A x - b^\top x \text{ subject to } Bx = c, x \geq 0.$$

The existence of solution of (14) is implied by the Weierstrass extreme value theorem: the real-valued cost function is continuous, and the non-empty feasible set is bounded.

However, the uniqueness of this solution is not so straightforward. It is given by the relationship between the null space (kernel) of Hessian matrix A , linear term b , and the feasible set Ω , since the differences between solutions lies in this vector space. For instance, if A is symmetric positive definite (SPD), then the cost function is strictly convex and the solution is unique on any non-empty closed convex feasible set, see [18]. Unfortunately, in our case the Hessian matrix is only SPS and the solution is not unique for an arbitrary feasible set. For example in unconstrained case, if $\Omega := \mathbb{R}^n$, then the problem could be possibly non-solvable; if $b \notin \text{Im } A$ then the linear system (12) has no solution. If $b \in \text{Im } A$ then the system of all solutions of the unconstrained problem is given by $\bar{x} = A^+b + d$, where A^+ denotes Moore–Penrose pseudoinverse of the singular matrix A , and vector d represents an arbitrary vector from (in this case non-trivial) $\text{Ker } A$. Therefore, all solutions of the problem differ by the vector from $\text{Ker } A$.

At first, we present the generalization of previous observation from the unconstrained case to solutions of a problem (14).

Lemma 1. *Let \bar{x}_1, \bar{x}_2 be two solutions of problem (14). Then*

$$\bar{x}_1 - \bar{x}_2 \in \text{Ker } A \cap \text{Ker } B.$$

Proof. Let us denote $d := \bar{x}_2 - \bar{x}_1$. Then using the definition of a quadratic cost function f and simple manipulations, we obtain

$$(15) \quad f(\bar{x}_2) = f(\bar{x}_1 + d) = f(\bar{x}_1) + d^\top \nabla f(\bar{x}_1) + \frac{1}{2} d^\top A d.$$

We suppose that both of \bar{x}_1 and \bar{x}_2 are minimizers (both of $f(\bar{x}_1)$ and $f(\bar{x}_2)$ are minimal values of f on the feasible set), therefore $f(\bar{x}_1) = f(\bar{x}_2)$. Using this and comparing sides of equality (15), we can write

$$(16) \quad \frac{1}{2} d^\top A d = -d^\top \nabla f(\bar{x}_1).$$

The left side of this equation is always non-negative because A is SPS. Moreover, the right side is non-positive because \bar{x}_1 is a solution of the convex optimization problem with differentiable f and the necessary optimality condition is given by (see [11])

$$\nabla f(\bar{x}_1)^\top (y - \bar{x}_1) \geq 0 \text{ for all } y \in \Omega.$$

Combining these two inequalities, we obtain

$$(17) \quad d^\top A d = 0 \wedge d^\top \nabla f(\bar{x}_1) = 0.$$

The first equality implies $d \in \text{Ker } A$.

We suppose that both of the solutions belong to the feasible set, therefore they satisfy constraint conditions. From equality conditions for \bar{x}_2 and using $B\bar{x}_1 = c$, we get

$$c = B\bar{x}_2 = B(\bar{x}_1 + d) = B\bar{x}_1 + Bd = c + Bd,$$

therefore $Bd = 0$ or equivalently $d \in \text{Ker } B$. □

In the proof of the previous Lemma, we did not yet use one important property, which appears in (17). Using $d \in \text{Ker } A$ from first equality of (17) we get equivalent condition

$$d^\top \nabla f(\bar{x}_1) = d^\top (A\bar{x}_1 - b) = -d^\top b = 0.$$

or equivalently (using $d \in \text{Ker } A \cap \text{Ker } B$, see Lemma 1)

$$b \perp \text{Ker } A \cap \text{Ker } B.$$

This condition forms the sufficient condition for a possible existence of two different solutions of the general QP problem (14).

However, these solutions could be constrained by additional inequality constraints and the full system of necessary conditions is more complicated. Let us introduce Lagrange function (see [44] and [18]) corresponding to the problem (14)

$$(18) \quad \mathcal{L}(x, \lambda_E, \lambda_I) := \frac{1}{2} x^\top A x - b^\top x + \lambda_E^\top (Bx - c) - \lambda_I^\top x,$$

where λ_I and λ_E are Lagrange multipliers corresponding to the equality and inequality constraints. So-called Karush-Kuhn-Tucker (KKT) optimality conditions are given by

$$(19) \quad \begin{aligned} Ax - b + B^\top \lambda_E - \lambda_I &= 0, \\ Bx - c &= 0, \\ x, \lambda_I &\geq 0, \\ [x]_j [\lambda_I]_j &= 0 \quad \forall j = 1, \dots, KT. \end{aligned}$$

where we use the notation $[v]_j$ to denote the j -th component of vector v .

Additionally, we can utilize the block-diagonal structure of our specific problem (10) given by the decomposition into clusters. Let us denote the block of matrix A by $\hat{A} \in \mathbb{R}^{T \times T}$ and corresponding blocks of vectors $x_k, b_k, \lambda_{Ik} \in \mathbb{R}^T, k = 1, \dots, K$. Then we can write the first KKT system in a form

$$\hat{A}x_k - b_k + \lambda_E + \lambda_{Ik} = 0, \quad k = 1, \dots, K.$$

Now we can sum all these equations to get

$$\hat{A} \left(\sum_{k=1}^K x_k \right) - \sum_{k=1}^K (b_k + \lambda_{Ik}) + K\lambda_E = 0$$

and since from equality constraint we have $\sum_{k=1}^K x_k = \mathbf{1}$ and $\mathbf{1} \in \text{Ker } \hat{A}$ (see (13)), we can write

$$(20) \quad \lambda_E = \frac{1}{K} \sum_{k=1}^K (b_k + \lambda_{Ik})$$

and substitute back into first KKT condition (19). Using the definition of a matrix B , we obtain

$$(21) \quad Ax - Qb - Q\lambda_I = 0, \quad Q := I - \frac{1}{K} B^\top B.$$

Here the orthogonal matrix $Q \in \mathbb{R}^{KT \times KT}$ represents the projector onto $\text{Ker } B$.

Using KKT optimality conditions and the block structure of the problem, we are able to prove the following lemma, which gives the relationship between Lagrange multipliers corresponding to different solutions.

Lemma 2. *Let \bar{x}_1, \bar{x}_2 be two different solutions of the problem (11) and let $\bar{\lambda}_{1I}, \bar{\lambda}_{1E}, \bar{\lambda}_{2I}, \bar{\lambda}_{2E}$ be corresponding Lagrange multipliers in KKT system (19). Then*

$$\bar{\lambda}_{1I} = \bar{\lambda}_{2I} \text{ and } \bar{\lambda}_{1E} = \bar{\lambda}_{2E}.$$

Proof. We have already shown that the Lagrange multipliers corresponding to equality constraints are uniquely given by the values of the Lagrange multipliers corresponding to the inequality constraints (20). Therefore, in the proof we will focus on a relationship between $\bar{\lambda}_{1I}$ and $\bar{\lambda}_{2I}$.

Let us consider two different solutions \bar{x}_1 and \bar{x}_2 . These solutions satisfy all KKT conditions (19), (21)

$$\begin{aligned} A\bar{x}_1 - Qb - Q\bar{\lambda}_{1I} &= 0, & A\bar{x}_2 - Qb - Q\bar{\lambda}_{2I} &= 0, \\ \bar{x}_1, \bar{\lambda}_{1I} &\geq 0, & \bar{x}_2, \bar{\lambda}_{2I} &\geq 0, \\ [\bar{x}_1]_j [\bar{\lambda}_{1I}]_j &= 0, & [\bar{x}_2]_j [\bar{\lambda}_{2I}]_j &= 0. \end{aligned}$$

Let us denote

$$(22) \quad \bar{x}_2 - \bar{x}_1 =: d, \quad \bar{\lambda}_{2I} - \bar{\lambda}_{1I} =: p$$

and substitute this into the first KKT condition for $(\bar{x}_2, \bar{\lambda}_{2I})$

$$(23) \quad A(\bar{x}_1 + d) - Qb - Q(\bar{\lambda}_{1I} + p) = 0.$$

Since $d \in \text{Ker } A$ (see Lemma 1) and using the first KKT condition for $(\bar{x}_1, \bar{\lambda}_{1I})$ we can write (23) in form

$$Qp = 0 \Rightarrow p \in \text{Im } B^\top.$$

Now we focus on the inequality conditions. We use our notation (22) and substitute into KKT conditions ($\forall j = 1, \dots, KT$)

$$\begin{aligned} [\bar{x}_2]_j - [d]_j &\geq 0, & [\bar{x}_1]_j + [d]_j &\geq 0, \\ [\bar{\lambda}_{2I}]_j - [p]_j &\geq 0, & [\bar{\lambda}_{1I}]_j + [p]_j &\geq 0. \end{aligned}$$

We multiply these inequalities by non-negative numbers $[\bar{\lambda}_{2I}]_j, [\bar{\lambda}_{1I}]_j, [\bar{x}_2]_j, [\bar{x}_1]_j$ and use complementarity KKT conditions. We get

$$\begin{aligned} [d]_j [\bar{\lambda}_{2I}]_j &\leq 0, & [d]_j [\bar{\lambda}_{1I}]_j &\geq 0, \\ [p]_j [\bar{x}_2]_j &\leq 0, & [p]_j [\bar{x}_1]_j &\geq 0. \end{aligned}$$

Adding complementarity conditions with substitution (22) and using original complementarity conditions

$$\begin{aligned} -[d]_j [\bar{\lambda}_{2I}]_j - [p]_j [\bar{x}_2]_j + [d]_j [p]_j &= 0 \\ [d]_j [\bar{\lambda}_{1I}]_j + [p]_j [\bar{x}_1]_j + [d]_j [p]_j &= 0 \end{aligned}$$

we end up with

$$(24) \quad \forall j = 1, \dots, KT : [d]_j [p]_j = 0.$$

Let us remark that this condition is much stronger than $d^\top p = 0$ which could be obtained directly from

$$\begin{aligned} d &\in \text{Ker } A \cap \text{Ker } B, \\ p &\in \text{Im } B^\top, \end{aligned}$$

using $\text{Ker } B \perp \text{Im } B^\top$ (see [38]).

Now we are ready to prove that $p = 0$. Since $p \in \text{Im } B^\top$, there exists $\alpha \in \mathbb{R}^T$ such that

$$\forall k = 1, \dots, K : p_k = \alpha.$$

Suppose by contradiction that there exists an index $i \in \{1, \dots, T\}$ such that $[\alpha]_i \neq 0$. Since (24) corresponding components of d have to be zero i.e.,

$$(25) \quad [d]_i = [d]_{i+T} = \dots = [d]_{i+(K-1)T} = 0.$$

However, if we suppose that $d \neq 0$ (solutions \bar{x}_1 and \bar{x}_2 are different), then the vector d with property (25) is not from $\text{Ker } A$ (see (13)), which is a contradiction. Therefore $p = 0$. \square

In general case, we still cannot prove the uniqueness conditions. The following lemma presents the situation when our problem (11) has an infinite number of solutions.

Lemma 3. *Let us consider the problem (11) with*

$$b \in \text{Im } B^\top$$

and $K \geq 2$. Then this problem has an infinite number of solutions. Moreover, one of these solutions is given by

$$(26) \quad \bar{x} = \frac{1}{K} \mathbf{1}.$$

Proof. At first, we prove that (26) is a solution. This point is not on the boundary of a feasible set, therefore we can ignore the inequality constraints — all of them are satisfied and corresponding $\bar{\lambda}_I = 0$. The first KKT condition (21) is given by

$$Ax = Qb.$$

Since Q is orthogonal projector onto $\text{Ker } B$, and we suppose $b \perp \text{Ker } B$, the right hand-side of this equation is equal to 0. Notice that $\bar{x} \in \text{Ker } A$, therefore also the left hand-side is equal to zero and the first KKT condition is satisfied. Moreover, the equality constraint could be also easily checked, therefore \bar{x} is solution.

Now it remains to show that there exists at least one additional solution, i.e., that there exists a non-zero vector $d \in \text{Ker } A \cap \text{Ker } B$ such that

$$\bar{x} + d \in \Omega.$$

Since we suppose $K \geq 2$, the vector space $\text{Ker } A \cap \text{Ker } B$ is nontrivial and it is possible to choose a non-zero vector from this space. Additionally, \bar{x} does not belong to the boundary of Ω , therefore there exists a nonzero vector in any *direction*. \square

3. SPECTRAL PROJECTED GRADIENT METHOD FOR QP PROBLEMS

There exist several types of algorithms for solving a general QP problem (14). Since the manipulation with both constraint types is usually difficult, these algorithms are based on elimination of one type of constraints in the outer loop, and then they solve the sequence of inner problems with the remaining type of constraints.

To be more specific, one can use popular Interior-Point (IP) methods, see [44] and [53]. These methods enforce the inequality constraints using a barrier function

$$\min_{Bx=c, x \geq 0} f(x) \approx \min_{Bx=c} f(x) + \mu \sum_{i=1}^T \log x_i.$$

Here $\mu > 0$ represents the barrier parameter. Using this approach, the algorithm transforms the original KKT system with inequalities (19) to the system of non-linear equations with so-called duality gap μ . The new problem is not equivalent to the original, but as the duality gap approaches zero, it becomes a better and better approximation. The barrier function increases all function values near the boundary of the feasible set to create a nonpenetrable barrier for a step-based algorithm which solves the inner problem with the remaining equality constraints. Usually μ is not implemented as a constant, but is a sequence $\mu_k \rightarrow 0$, and the solution of a previous inner problem is used as an initial approximation of the inner algorithm for a new μ_{k+1} . Since the non-quadratic term (logarithm) is added to the original quadratic function f , the corresponding KKT system of this inner problem is non-linear. To solve this system, one can use Newton-type methods with step-size control to be sure that new step will not jump through the barrier. Newton-type methods for solving non-linear systems introduce the sequence of linear equations which have to be solved. For more details see [44] and [53].

Another approach is to deal with equality constraints first. This can be performed using Augmented Lagrangian methods, see [44] and [18]. The algorithm enforces the equality constraints using a penalty term

$$\min_{Bx=c, x \geq 0} f(x) \approx \min_{x \geq 0} f(x) + \rho \|Bx - c\|^2.$$

Again, the new problem is not equivalent to the original one, although if $\rho \rightarrow \infty$ then $\|Bx - c\| \rightarrow 0$. Therefore, the penalty parameter ρ is usually implemented as the increasing sequence. The main advantage of this approach is the structure of inner problem with inequalities - this new problem is again a QP. However, the Hessian matrix of the new problem is given by

$$\nabla^2 (f(x) + \rho \|Bx - c\|^2) = A + \rho B^T B,$$

therefore, since the condition number depends on the value ρ and while $\rho \rightarrow \infty$, the problem becomes harder and harder to solve. Similar to the interior-point methods, there exists extensive theory about the connection between the stopping criteria for solving ill-conditioned inner problems and the value of a penalty parameter; see Semimonotonic Augmented Lagrangian algorithm [18] or [19]. The inner QP problem can be solved using the interior-point method, active-set algorithms or projected gradient methods. In the case of bound constraints, the projection onto a feasible set is trivial. Therefore, in the case of the active-set algorithm and projected gradient methods, the inequality constraints are satisfied accurately, which is not the case for the interior-point methods. The barrier function makes it impossible to find the solution on the boundary of a feasible set, because in that case the value of a barrier term is equal to infinity.

This property brings us to the main disadvantage of both approaches. In the case of interior-point methods, it is impossible to satisfy inequality constraints accurately. In the case of penalization technique, it is impossible to satisfy equality constraints exactly.

Fortunately, our QP problem (10) is not a general QP (14). To be more specific, the feasible set in our case is a separable set composed from T simplexes of size K . There exists an efficient algorithm for computing the projection of a general point

onto simplex

$$\begin{aligned}
 P_{\Omega_t}(y) &:= \arg \min_{x \in \Omega_t} \|y - x\|, \\
 \Omega_t &:= \{\gamma_t \in \mathbb{R}^K : \gamma_t \geq 0 \wedge \sum_{k=1}^K [\gamma_t]_k = 1\}, \quad t = 1, \dots, T, \\
 \gamma_t &:= [\gamma_1(t), \dots, \gamma_K(t)]^\top.
 \end{aligned}$$

The algorithm was presented in [14] and it computes the projection onto simplex of dimension K in at most K steps, see Algorithm 2.

Algorithm 2. Projection onto simplex [14].

Given arbitrary $y \in \mathbb{R}^K$

if $K = 1$ **then set** $P(y) := 1$ **and stop.**

sort y **in ascending order and set** $k := K - 1$

while $k > 0$

$\alpha := \left(\sum_{j=k+1}^K [y]_j - 1 \right) / (K - k)$

if $\alpha > [y]_k$ **then** $\hat{\alpha} := \alpha$ **and** $k := -1$

else $k := k - 1$

endwhile

if $k = 0$ **then** $\hat{\alpha} := \left(\sum_{j=1}^K [y]_j - 1 \right) / K$

set $[P(y)]_k := \max\{[y]_k - \hat{\alpha}, 0\}$ **for all** $k = 1, \dots, K$

Return $P(y)$.

If we use these projections in our algorithm, all constraint conditions will be satisfied accurately. Moreover, computation of the projection can be performed as T independent processes and since T is the largest parameter of problem dimension, this approach increases the granularity of the overall solution process - making it is suitable for GPU computation. We can also use the value of original cost function to stop our algorithm with respect to sufficient decrease given by demands from outer Algorithm 1.

Therefore, we are mainly interested in the projected gradient descent methods, i.e. in the algorithms which use $x^0 \in \Omega$ as an initial approximation and suitable step-lengths $\alpha_k > 0$ to generate the approximations of the solution by

$$(27) \quad x^{k+1} = P_{\Omega}(x^k - \alpha_k \nabla f(x^k)), \quad k = 0, 1, \dots$$

In this case, the feasibility of generated approximations is enforced by using the projections, and the descent of object function is induced by using $-\nabla f(x)$, which is generally the best local decrease direction. One of the most efficient projection gradient methods is a spectral projected gradient method (SPG) [9]. The first part of a one SPG iteration is based on generating the point using (27) with step-size defined by Barzilai-Borwein algorithm (BB), [7]. However, the projected variant of BB is not convergent in general and the original proof of convergence cannot be applied, see [17]) and additional line-search technique has to be implemented.

In this Section, we shortly review both of components of SPG method, i.e. projected BB method and additional generalized Armijo condition. The method was developed to solve general optimization problems and in this paper, we add our own modification for solving QP problems using the properties of the quadratic objective function.

Let us follow [47]. BB is the non-monotone gradient descent method for solving unconstrained convex optimization problems. These methods are based on a construction of a sequence of solution approximations using the recursive formula

$$(28) \quad x^{k+1} = x^k - \alpha_k g^k, \quad k = 0, 1, \dots$$

with a step size $\alpha_k \in \mathbb{R}^+$ and a vector of the steepest descent $-g^k := -\nabla f(x^k)$. The most popular gradient descent method is the Steepest Descent method (SD, firstly presented by Cauchy [13]). This method uses the step-length, which minimizes the function $f(x^{k+1})$ using locally optimal step-size

$$(29) \quad \alpha_k = \arg \min_{\alpha \in \mathbb{R}} f(x^k - \alpha \nabla f(x^k)) = \frac{\langle g^k, g^k \rangle}{\langle A g^k, g^k \rangle},$$

which leads to the monotone descent of the objective function.

However, the step-size of BB method is based on the different idea. To briefly review the relation of the BB method for the solution of unconstrained problems to the Newton's method for solving a scalar non-linear equation

$$g(x) = 0, \quad g : \mathbb{R} \rightarrow \mathbb{R},$$

let us replace the derivative $g'(x^k)$ in the Newton's method by its secant approximation to get

$$(30) \quad x^{k+1} = x^k - \frac{1}{g'(x^k)} g(x^k) \approx x^k - \frac{x^k - x^{k-1}}{g(x^k) - g(x^{k-1})} g(x^k).$$

Denoting $g^k = g(x^k) = f'(x^k) = \nabla f(x^k)$ and

$$(31) \quad \alpha_k = \frac{x^k - x^{k-1}}{g^k - g^{k-1}},$$

we can see that the secant method (30) can be considered as a gradient descent method (28). If $g(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n$, then we cannot evaluate α_k by (31), but we can assemble the *secant equation*

$$(32) \quad \frac{1}{\alpha_k} (x^k - x^{k-1}) = g^k - g^{k-1}.$$

After denoting

$$s^k = x^k - x^{k-1}, \quad g^k - g^{k-1} = A s^k,$$

and solving (32) in the least-squares sense

$$\alpha_k = 1 / \arg \min_{\beta \in \mathbb{R}} (\langle s^k, s^k \rangle \beta^2 - 2 \langle A s^k, s^k \rangle \beta + \langle A s^k, A s^k \rangle)$$

and some simplifications, we get

$$(33) \quad \alpha_k = \frac{\langle s^k, s^k \rangle}{\langle A s^k, s^k \rangle}.$$

This is the step-size of the BB method. The proof of convergence with estimates for solving the unconstrained QP problem (i.e., (14) with $\Omega = \mathbb{R}^n$) was presented in [16].

However, the projected variant of BB (i.e., (27) with (33)) is not convergent in general and the original proof of convergence cannot be applied, see [17].

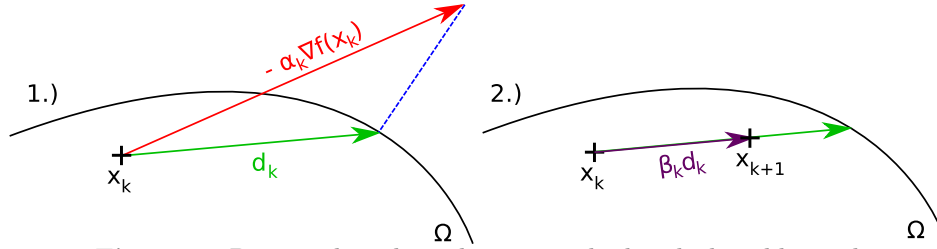


Figure 1. Projected gradient descent method with the additional line-search in two steps, 1.) computation of projected gradient and 2.) an additional line-search.

One option how to enforce the convergence of the method is to use an additional line-search. Let us denote the difference

$$(34) \quad g_{\alpha_k}^P(x^k) = P_{\Omega}(x^k - \alpha_k \nabla f(x^k)) - x^k$$

as a *projected gradient* in point $x^k \in \Omega$ with the step-length $\alpha_k > 0$. To enforce the convergence, the SPG algorithm uses an additional line-search step

$$(35) \quad x^{k+1} = x^k + \beta_k d_k$$

with $d_k := g_{\alpha_k}^P(x^k)$ and an *appropriate* choice of the step-size $\beta_k \in (0, 1]$. The method with these two steps (computation of the projected gradient and an additional line-search) is demonstrated in Fig. 1.

The next Lemma demonstrates the reason for using the projected gradient as a search direction in (35): it is a descent direction.

Lemma 4. *Let $x \in \Omega$, $\alpha > 0$ and $g_{\alpha}^P(x) = P_{\Omega}(x - \alpha \nabla f(x)) - x \neq 0$. Then*

$$(36) \quad \langle g_{\alpha}^P(x), \nabla f(x) \rangle < 0.$$

Proof. We suppose $\Omega \subset \mathbb{R}^n$ is a closed convex set, therefore (see [8])

$$\forall y, z \in \mathbb{R}^n : \langle P_{\Omega}(y) - P_{\Omega}(z), y - z \rangle \geq \|P_{\Omega}(y) - P_{\Omega}(z)\|^2.$$

If we choose $y = x - \alpha \nabla f(x)$ and $z = x = P_{\Omega}(x)$, then we can estimate

$$\begin{aligned} -\alpha \langle g_{\alpha}^P(x), \nabla f(x) \rangle &= \langle g_{\alpha}^P(x), (x - \alpha \nabla f(x)) - x \rangle \\ &\geq \|g_{\alpha}^P(x)\|^2 > 0. \end{aligned}$$

□

The SPG algorithm uses Grippo, Lampariello, and Lucidi method (GLL) [25] to find appropriate step-size β_k . This algorithm is based on a bisection method to satisfy the so-called *generalized Armijo condition*

$$(37) \quad f(x^k + \beta_k d^k) < f_{\max} + \tau \beta_k \langle \nabla f(x^k), d^k \rangle.$$

Here $\tau \in (0, 1)$ represents a safeguarding parameter and

$$f_{\max} := \max\{f(x^{k-j}) : 0 \leq j \leq \min\{k, m-1\}\}.$$

The main difference between this generalized version and the original Armijo conditions [44] is in the utilization of function values in m previous approximations instead of using only the previous $f(x^{k-1})$. This approach supplies the non-monotonic behavior of the BB method and at the same time controls the descent. The proof of convergence of SPG is based on satisfying the generalized Armijo condition in

every step, see [9]. We present the SPG method by Algorithm 3 and GLL method by Algorithm (4).

Algorithm 3. the original SPG method [10].

Given cost function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, initial approximation $x^0 \in \Omega$, projection onto feasible set $P_\Omega(x)$, safeguarding parameters $0 < \alpha_{\min} \ll \alpha_{\max}$, precision $\varepsilon > 0$, and initial step-size $\alpha_0 > 0$.

```

k := 0
while  $\|P_\Omega(x^k - \nabla f(x^k)) - x^k\| > \varepsilon$ 
   $d^k := P_\Omega(x^k - \alpha_k \nabla f(x^k)) - x^k$ 
  compute step-size  $\beta_k$  using GLL
   $x^{k+1} := x^k + \beta_k d^k$ 
   $s^k := x^{k+1} - x^k$ 
   $y^k := \nabla f(x^{k+1}) - \nabla f(x^k)$ 
  if  $\langle s^k, y^k \rangle \leq 0$ 
     $\alpha_{k+1} := \alpha_{\max}$ 
  else
     $\alpha_{k+1} := \min\{\alpha_{\max}, \max\{\alpha_{\min}, \langle s^k, s^k \rangle / \langle s^k, y^k \rangle\}\}$ 
  endif
  k := k + 1
endwhile

```

Return approximation of solution x^{k+1} .

Algorithm 4. GLL line-search [25].

Given cost function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, parameter $m \in \mathbb{N}$, approximation and direction $x^k, d^k \in \mathbb{R}^n$, parameter $\tau \in (0, 1)$, safeguarding parameters $\sigma_1, \sigma_2 \in \mathbb{R} : 0 < \sigma_1 < \sigma_2 < 1$.

```

 $f_{\max} := \max\{f(x^{k-j}) : 0 \leq j \leq \min\{k, m-1\}\}$ 
 $x_{temp} := x^k + d^k$ 
 $\delta := \langle \nabla f(x^k), d^k \rangle$ 
 $\beta := 1$ 
while  $f(x_{temp}) > f_{\max} + \delta\beta\delta$ 
   $\beta_{temp} := -\frac{1}{2}\beta^2\delta / (f(x_{temp}) - f(x^k) - \beta\delta)$ 
  if  $\beta_{temp} \in (\sigma_1, \sigma_2\beta)$ 
     $\beta := \beta_{temp}$ 
  else
     $\beta := \beta/2$ 
  endif
   $x_{temp} := x^k + \beta d^k$ 
endwhile

```

Return step-size β .

The main bottleneck of the GLL method is the computational complexity, which cannot be estimated in general. To be more specific, it is hard to say when the bisection will be finished.

The SPG was developed to solve more general optimization problems on convex sets. In our problems, the cost function is a quadratic function. We can use its particular form and its properties to simplify the GLL algorithm, obtaining an algorithm with fewer cost function evaluations, i.e., with a smaller number of the most time-consuming operations - multiplications by the Hessian matrix A . The motivation came from the other well-known algorithms for solving QP, like steepest descent method and the conjugate gradient method, see [30].

This modification was initially presented in [45] and it reduces the bisection in GLL to simple formula.

At first, let us present the basic equality in QP, see [18] ($\forall x, d \in \mathbb{R}^n, \forall b \in \mathbb{R}$)

$$(38) \quad f(x + \beta d) = f(x) + \beta \langle Ax - b, d \rangle + \frac{1}{2} \beta^2 \langle Ad, d \rangle.$$

We start the simplification of SPG for solving QP problems with the most obvious simplifications. Notice that in the Algorithm 3 we can write

$$\begin{aligned} y^k &= \nabla f(x^{k+1}) - \nabla f(x^k) = (Ax^{k+1} - b) - (Ax^k - b) \\ &= A(x^{k+1} - x^k) = As^k. \end{aligned}$$

Since matrix A is SPS, we can write for any $s^k \in \mathbb{R}^n \setminus \text{Ker } A$

$$\langle s^k, y^k \rangle = \langle s^k, As^k \rangle > 0,$$

and the condition in SPG Algorithm 3 is always satisfied.

Moreover, the BB step-length (33) is the inverse Rayleigh quotient (with $s^k \notin \text{Ker } A$) and it can be bounded by

$$\frac{1}{\lambda_{\max}} \leq \alpha_{k+1} \leq \frac{1}{\hat{\lambda}_{\min}},$$

where $\hat{\lambda}_{\min}$ is the smallest non-zero eigenvalue and λ_{\max} is the largest eigenvalue of the matrix A (please notice that in our case, the Hessian matrix is SPS (9), (13)). Therefore, we can omit safeguarding parameters α_{\min} and α_{\max} in the Algorithm 3.

Let us take a better look into GLL line-search Algorithm 4. The computation of β_{temp} can be simplified using (38). We obtain

$$\begin{aligned} \beta_{\text{temp}} &:= -\frac{\beta^2 \delta}{2(f(x^k + \beta d^k) - f(x^k) - \beta \delta)} \\ &= -\frac{\beta^2 \delta}{2\beta \delta + \beta^2 \langle Ad^k, d^k \rangle - 2\beta \delta} \\ &= -\frac{\langle \nabla f(x^k), d^k \rangle}{\langle Ad^k, d^k \rangle} := \bar{\beta}. \end{aligned}$$

This is a simple Cauchy step-size (29). Since the vector d^k is the descent direction (36) and A is SPS, our optimal $\bar{\beta}$ is positive.

Obviously, the computation of a new β_{temp} is independent of the previous value and the original GLL method performs solely the bisection method, i.e., it tries to halve the coefficient β and to verify the generalized Armijo condition. Furthermore, the value of a step-size β has to be from the interval $[\sigma_1, \sigma_2] \subseteq [0, 1]$, because the smaller or larger value may cause a departure from the feasible set.

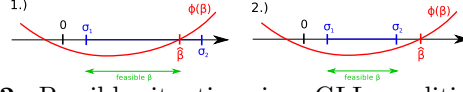


Figure 2. Possible situations in a GLL condition for QP.

The division of step-size β by two now modifies only the generalized Armijo condition. This condition can be also simplified

$$\begin{aligned}
0 &> f(x^k + \beta d^k) - f_{\max} - \tau \beta \delta \\
&= f(x^k) + \beta \langle \nabla f(x^k), d^k \rangle + \frac{1}{2} \beta^2 \langle Ad^k, d^k \rangle \\
&\quad - f_{\max} - \tau \beta \langle \nabla f(x^k), d^k \rangle \\
&= \frac{1}{2} \beta^2 \langle Ad^k, d^k \rangle + (1 - \tau) \beta \langle \nabla f(x^k), d^k \rangle \\
&\quad + f(x^k) - f_{\max} \\
0 &> \frac{1}{2} \beta^2 + (1 - \tau) \beta \frac{\langle \nabla f(x^k), d^k \rangle}{\langle Ad^k, d^k \rangle} \\
&\quad + \frac{1}{\langle Ad^k, d^k \rangle} (f(x^k) - f_{\max}).
\end{aligned}$$

Afterwards, we denote the function on the right hand-side and the constant term by

$$\Phi(\beta) := \frac{1}{2} \beta^2 - (1 - \tau) \bar{\beta} \beta - \xi, \quad \xi := \frac{1}{\langle Ad^k, d^k \rangle} (f_{\max} - f(x^k)).$$

We are interested in β such that the generalized Armijo condition in a form

$$(39) \quad \Phi(\beta) < 0$$

is satisfied. The positive root of $\Phi(\beta)$ is given by

$$\hat{\beta} := (1 - \tau) \bar{\beta} + \sqrt{(1 - \tau)^2 \bar{\beta}^2 + 2\xi}.$$

There exist only two possible situations, see Fig. 2. Therefore, we can conclude that the feasible step-size in the second step of SPG could be defined as

$$\beta_k \in \left[\sigma_1, \min\{\sigma_2, \hat{\beta}\} \right].$$

This simple interval can replace GLL, i.e., any β_k from this interval satisfies the generalized Armijo condition.

The computation of the function values can be also simplified using (38)

$$f(x^{k+1}) = f(x^k) + \beta_k \langle g^k, d^k \rangle + \frac{1}{2} \beta_k^2 \langle Ad^k, d^k \rangle.$$

Finally, we can simplify the computation of the BB step-length using (35)

$$\alpha_{k+1} = \frac{\langle s^k, s^k \rangle}{\langle s^k, y^k \rangle} = \frac{\langle s^k, s^k \rangle}{\langle s^k, As^k \rangle} = \frac{\langle \beta_k d^k, \beta_k d^k \rangle}{\langle \beta_k d^k, \beta_k Ad^k \rangle} = \frac{\langle d^k, d^k \rangle}{\langle d^k, Ad^k \rangle}$$

and the recursive formula for the computation of a new gradient

$$g^{k+1} := Ax^{k+1} - b = A(x^k + \beta_k d^k) - b = g^k + \beta_k Ad^k.$$

We use all these simplifications to form the Algorithm 5. For the sake of simplicity, we relabel the coefficient $\tau := 1 - \tau \in (0, 1)$.

Notice that the most time-consuming operation - multiplication by Hessian matrix A - is performed only once per iteration. Moreover, all scalar products in every

Algorithm 5. SPG for QP problems (SPG-QP).

Given cost function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, initial approximation $x^0 \in \Omega$, projection onto feasible set $P_\Omega(x)$, parameters $m \in \mathbb{N}, \tau \in (0, 1)$, safeguarding parameters $\sigma_1, \sigma_2 \in \mathbb{R} : 0 < \sigma_1 < \sigma_2 < 1$, precision $\varepsilon > 0$, and initial step-size $\alpha_0 > 0$.

```

k := 0
g0 := Ax0 - b
f0 := 1/2⟨g0 - b, x0⟩
for k = 0, 1, ...
    dk := PΩ(xk - αkgk) - xk
    compute matrix-vector multiplication Adk
    compute multiple dot-product ⟨dk, {dk, Adk, gk}⟩
    if √⟨dk, dk⟩ ≤ ε then stop.
    fmax := max{f(xk-j) : 0 ≤ j ≤ min{k, m-1}}
    ξ := (fmax - fk) / ⟨dk, Adk⟩
    β̄ := -⟨gk, dk⟩ / ⟨dk, Adk⟩
    β̂ := τβ̄ + √τ2β̄2 + 2ξ
    choose βk ∈ ⟨σ1, min{σ2, β̂}⟩
    xk+1 := xk + βkdk
    gk+1 := gk + βkAdk
    fk+1 := fk + βk⟨dk, gk⟩ + 1/2βk2⟨dk, Adk⟩
    αk+1 := ⟨dk, dk⟩ / ⟨dk, Adk⟩
    k := k + 1
endwhile
    
```

Return approximation of solution x^k .

iteration can be performed as a single operation. This feature decreases the amount of global communication during the solution process.

4. HPC IMPLEMENTATION

We are developing and maintaining a new HPC library [46] for nonstationary time series analysis in C++ using PETSc [6]. This library supports the manipulation of vectors distributed on multiple nodes. These data can be used during computation on CPU or GPU. Therefore, the user of our library can decide which architecture will be used for computation.

The solution of our problem consists of two different types of parallelization. The first type is straightforward: the problem (9) has to be solved for several values of regularization parameters ϵ^2 and various numbers of clusters K . Moreover, the solution obtained by iterative process depends on an initial approximation. Each combination of these parameters can be used to run a completely independent instance of the Algorithm 1. The parallelization in this case is *embarrassingly-parallel*.

A more complicated parallelization scheme has to be used in a case where one instance of the Algorithm 1 cannot be run on a single node, because of the size of the input data and/or the size of the unknowns, especially a size of the vector γ . To deal with this problem, we naturally distribute the given long time series data $X \in \mathbb{R}^{T \times n}$ into nodes as successive disjoint time subintervals with approximately the same size, see Fig. 3.

Decomposition in time plays a key role in the effective computation of projections used in the SPG-QP, see Algorithm 5. Using our approach, all data of one simplex are stored in one particular node, therefore the projection is computed on each node

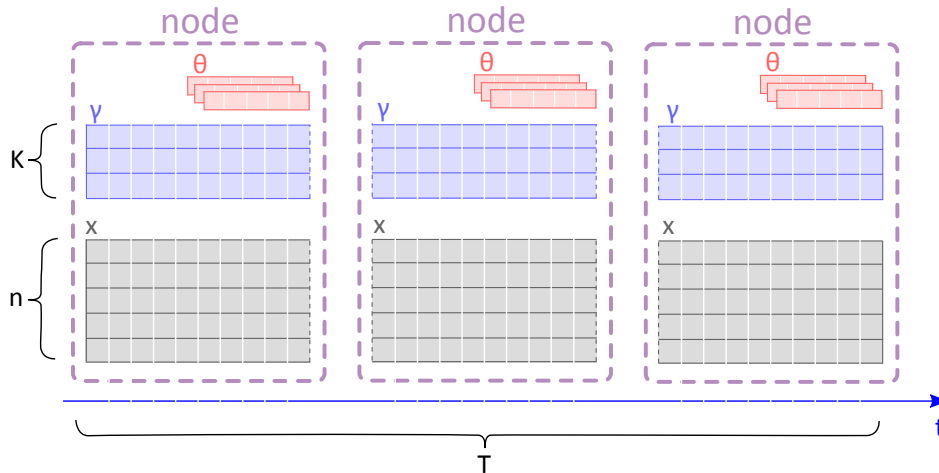


Figure 3. Large global vector of time series data X is distributed into nodes in a natural time-splitting way. Moreover, each node owns the part of global vector γ , which corresponds to the time part of local data. Since the number of model parameters Θ is small, each node owns its own local copy.

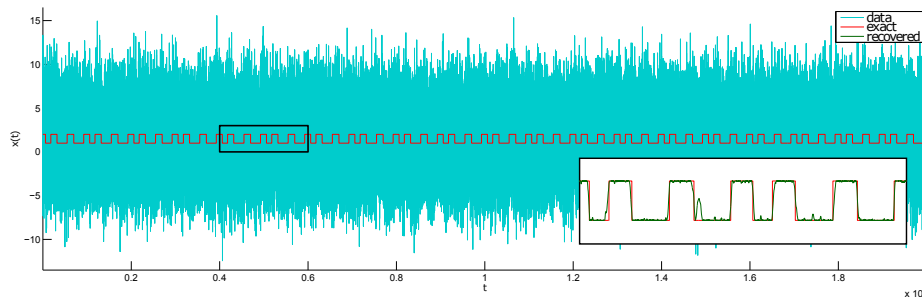


Figure 4. The beginning part of the benchmark signal of total length $T = 10^7$. Here, black box represents a part of the signal which is further enlarged to present the difference between original signal (red) and the solution, namely the denoised signal (green). This denoised signal was obtained using optimal penalty parameter $\varepsilon^2 = 3000$ (obtained with the standard L-curve method [26]) with norm of absolute error 0.04.

fully independently. However, this decomposition in time brings new difficulties like disruption of diagonal-block structure of A defined in analysis in form (10), therefore we had to implement an additional local-to-global index recomputation.

5. NUMERICAL EXPERIMENTS

To demonstrate the scalability of our QP solver, we consider a time series K-means clustering problem. This problem is characterized by the most basic modeling functions: the piecewise-constant functions. We are trying to model the given

data using the constant mean value in every cluster in least-square sense with the FEM-H1 regularization penalty in time:

$$\begin{aligned} \forall t \in T_k : x_t &= \theta_k + \epsilon_t \\ L_\varepsilon(\theta_1, \dots, \theta_K, \Gamma) &= \sum_{t=0}^T \sum_{k=1}^K \gamma_k(t)(x_t - \theta_k)^2 \\ &+ \varepsilon^2 \sum_{k=1}^K \sum_{t=0}^{T-1} (\gamma_{k,t+1} - \gamma_{k,t})^2 \end{aligned}$$

As a benchmark, we take a short signal of length 10^4 and repeat this short signal to obtain long time series $X_{\text{exact}}(t)$. This long signal is considered to be an *exact* benchmark solution of our denoising algorithm. As an input of our problem we consider the signal with a variable noise ε

$$(40) \quad X(t) = X_{\text{exact}} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 10).$$

Fig. 4 presents a beginning part of the considered long signal of length $T = 10^7$.

We provide the exact parameter solution $K = 2, \theta_1 = 1, \theta_2 = 2$ and solve the pure QP problem that represents the computational bottleneck of this time series denoising procedure. In SPG-QP, we choose $\tau = 0.9$ and $m = 20$ with respect to original SPG recommendations [9]. As a stopping criteria we choose the Euclidean norm of a projected gradient

$$\|g^P(x)\| := \|P(x - \alpha \nabla f(x)) - x\| < 10^{-6}.$$

We implement Algorithm 1 and Algorithm 5 in the PETSc framework and solve this problem for several values of penalty parameter ε , see Fig. 5. Standard methods like L-curve [26] are then used to identify the optimal values for ε . Based on these results, we choose value $\varepsilon = 3000$ for the following scalability tests.

To demonstrate the scalability of our implementation, we solve the above-mentioned problem of parameters $T = 10^7, K = 2, \varepsilon^2 = 3000$ with $\theta_1 = 1$ and $\theta_2 = 2$ on CSCS Piz Daint using $N \in \{1, 2, 4, 8, 16, 32, 64\}$ nodes. For complete specification of this machine see [1]. For GPU computation, we run one MPI process per hybrid node (Intel Xeon E5-2690 v3 with NVIDIA Tesla P100), which uses the GPU for computation. In the case of CPU, we use pure CPU-nodes (2x Intel Xeon E5-2695, each with 18 cores) and run 36 MPI processes per node. PETSc deliberately chose not to support a multi-threaded model, but rather only a multi-task model (i.e., multiple MPI processes). Since we are using PETSc, we are left with no choice and we do not use any CPU-thread parallelization (e.g., OpenMP) in our implementation.

We generated a time series signal of the length 10^7 and denoise it on different numbers of nodes. For statistical reasons, we decided to focus on the average numbers of QP iterations - where averaging was performed over different numbers of involved nodes. Please see the computation time of one iteration and number of iterations per second provided in Fig. 6. Here we can observe good scalability of CPU for a small number of nodes. However, the problem is too small for larger number of CPUs or GPUs, therefore the speedup is rapidly decreasing due to MPI communication. In all cases, the GPU computation is faster, but in this case, we should also consider the energy consumption, see Fig 7. These values were computed using the technique presented in [22].

Next, we would like to compare the introduced SPG-QP method for solving inner QP problem with other existing HPC open-source implementations. The

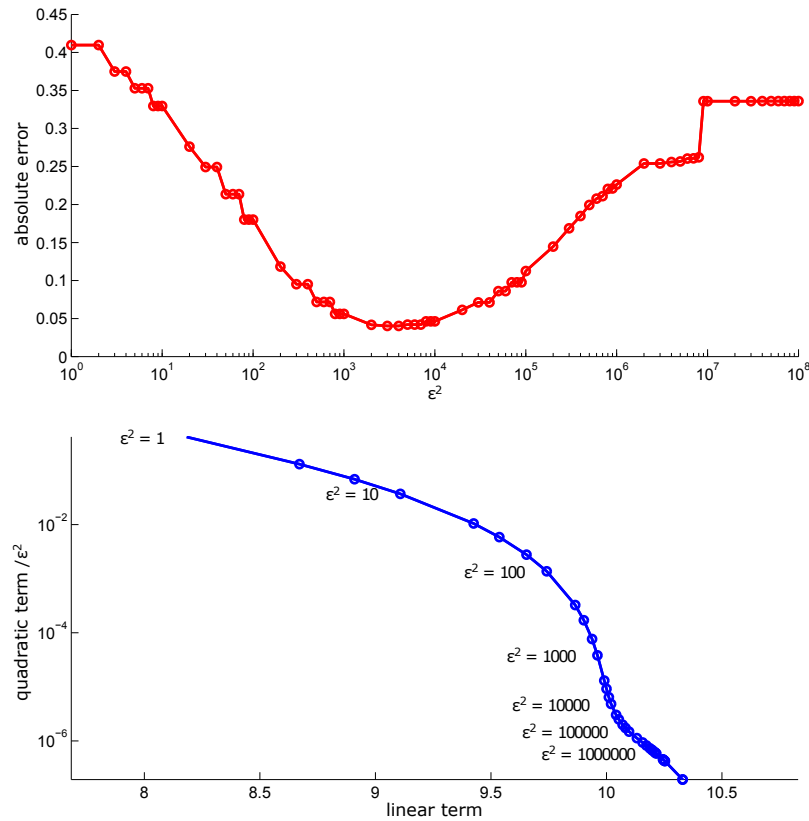


Figure 5. The mean value of L1-norm of absolute error for several values of penalty parameter (top). The left part of graph represents the error which arises due to insufficient regularization and the right part is caused by too much regularization, for the beginning part of benchmark signal of total length $T = 10^7$ (top). The L-curve (bottom) presents the relation between values of the linear term (modeling error) and quadratic term (regularization) which depends on the choice of regularization parameter.

most straightforward choice is to use methods already implemented in PETSc, for instance the Toolkit for Advance Optimization (TAO [4]). However, for the combination of linear equalities and bound constraints (8), one can only use the Interior-Point method. Unfortunately, the actual manual pages say that the state-of-the-art implementation of the Interior-Point method in TAO is more of a place-holder for future constrained optimization algorithms and should not yet be used for large problems or production code. The most promising implementation of QP solvers in PETSc is PERMON [3]. The results for solving QP problems arising in linear elasticity contact problems [28] suggest good scalability performance and efficiency in the case of massively parallel CPU computation. However, our problem (10) has particular properties, especially the null spaces of the Hessian matrix and the matrix of linear equality constraints being not disjoint, see Lemma 1. Therefore, in

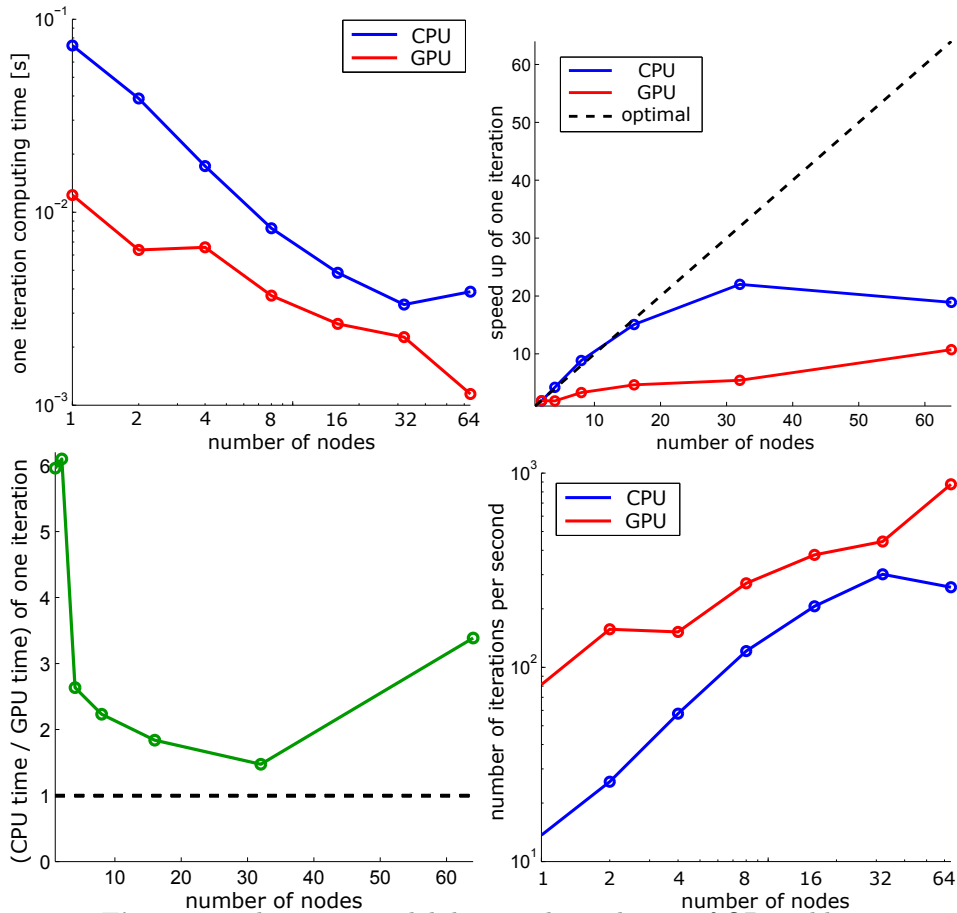


Figure 6. The strong scalability results: solution of QP problem of size $2 \cdot 10^7$ on Piz Daint machine using different architectures — the computation time of one iteration (upper right), relative speed-up of computation time of one iteration (upper left), the number of iterations per second (lower left), and computation times on CPU and GPU (lower right).

our case the QP problem could have more than one solution. Besides that, we have tried to solve our problem with PERMON, but the algorithm is not able to solve problems with larger dimensions $T > 10^4$. Also the parameters of the algorithm have to be more precisely investigated and their tuning is non-trivial. Luckily, authors are working on generalization of the inner solver for solving the problems with singular Hessian matrices [20] and this approach should be implemented soon. Moreover, the GPU implementation is also future work. Summarizing these experiences, we were not able to find any QP-solver that would be applicable to the particularly-structured problem (10) of time series analysis - when the time series data is big (i.e., when $T \gg 10^4$). For complete survey of existing HPC QP libraries see [27].

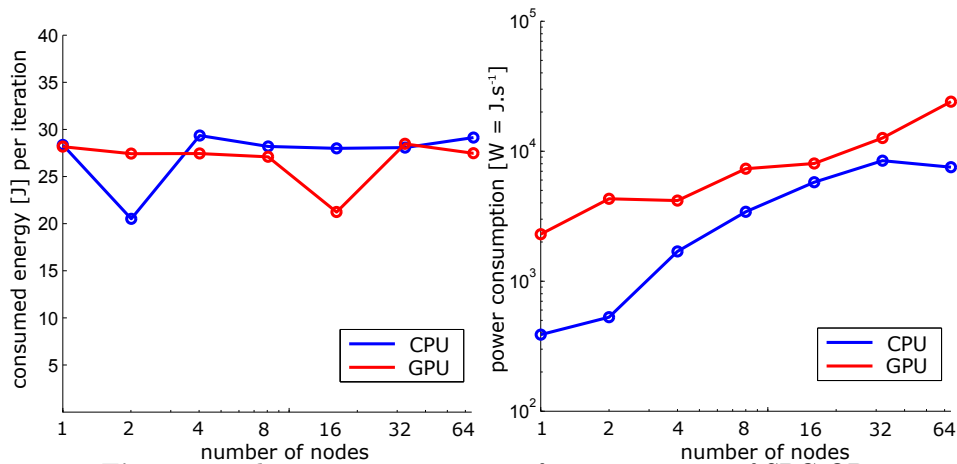


Figure 7. The energy consumption for one iteration of SPG-QP (left) and the power consumption (right).

Next, we also compare the introduced FEM-H1 with implementation to standard denoising methods. We were not able to find any HPC open-source library which includes the HPC implementation of the denoising methods for time series on hybrid architectures. In our implementation, we decided to use PETSc for basic vector/matrix operations, therefore we are directly able to switch between CPU/GPU computations. Existing denoising libraries (e.g. Dlib C++ [2]) usually implement these operations using own code from the scratch, including basic vector/matrix operations or (in the best case) use only sequential external BLAS libraries and cannot run on distributed memory architectures. Therefore, we decided to compare the denoising efficiency only for relatively short signals using the uniform sequential implementations in Matlab [43]. To be more specific, we compare the following denoising algorithms

- *Fourier* uses Matlab implementation of fft/iff and has one parameter s , which defines the size of the window.
- *Fourier L2* is Fourier filtering with an additional L2 regularization term. There are two parameters: s as a size of the window and λ as a regularization parameter.
- *Fourier Sobolev* is Fourier filtering with additional Sobolev prior penalty. Algorithm uses two parameters: size of the window s and regularization parameter λ .
- *Fourier TVR* is Fourier transformation with Total Variation Regularization. The method leads to the system of non-linear equations, which is typically solved using gradient method with constant step-size and monotone descent of the solution error. The method uses three parameters: size of the window s and regularization parameters λ, ε .
- *Bayesian HMM* is Bayesian Hidden Markov Model method [43]. This machine-learning algorithm uses random initial guesses, therefore we run it 10 times and take the best solution with respect to the absolute error.

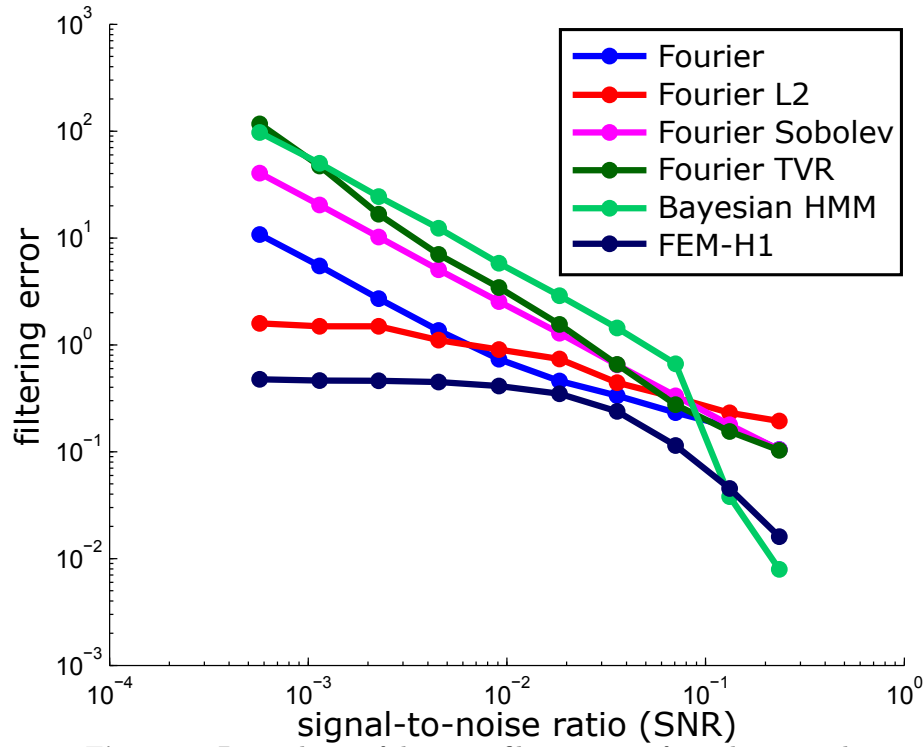


Figure 8. Dependence of the mean filtering error from the original signal-to-noise ratio (SNR). Results were obtained for the signal with Gaussian noise by averaging 100 independent realizations of the noisy signal data for each of the methods.

We have generated large set of above-mentioned testing signals of length $T = 1000$. For every standard deviation

$$\sigma \in 0.1 \cdot [2, 8, 32, 128, 512, 2048, 8192, 32768, 131072, 524288]$$

we generated 100 signals with random noise using formula (40) and then denoised the signal using proposed algorithms. We set various algorithm parameters $s \in [5, 20, 30, 40, 60, 80]$, $\lambda = [0.111050100]$, $\varepsilon = [0.001, 0.01, 0.1]$ and consider only the best solution with respect to the absolute norm computed as a difference between the denoised signal and the exact signal X_{exact} . Similarly for FEM-H1, we solved the problem for various values of penalty parameter.

In the end of solution process, we computed the average absolute error value through all random noises. The results are presented in Fig. 8. Here, the signal-to-noise ratio (SNR) was computed as the ratio of the maximum variation of the true signal to the maximum variation of the noisy signal.

As can be seen from Fig. 8, FEM-H1 methodology outperforms the standard methods when the signal-to-noise ratio becomes smaller (i.e., when the noise is becoming larger as compared to the true underlying signal).

6. CONCLUSION

In this paper, we introduced an extension of the nonparametric FEM-H1 framework allowing its application to denoising of very large time series data sets. We investigated basic properties of the inner large-scaled QP sub-problem - being the most expensive part of the FEM-H1 non-stationary time series analysis methodology. To solve this problem with HPC, we presented a modification of the Spectral projected gradient method for QP problems. This method is based on projections, enjoys high granularity of parallelization and it is suitable to run on GPU clusters, such as Piz Daint at the Swiss Supercomputing Centre CSCS. We presented numerical results for solving a large-scaled time series denoising problem.

In the future work, we will compare SPG-QP with state-of-the-art parallel implementations of popular optimization methods for solving not only benchmarks, but also solving problems in practical applications, such as the inference of causality networks from multiscale economical data. Additionally, our code will be extended by spatial regularization to increase the number of data analysis applications which could be practically solved by our emerging open-source HPC-library.

7. ACKNOWLEDGMENTS

This work is supported by Platform for Advanced Scientific Computing (PASC) and financed from PASC project *Horenko I., Gagliardini P., Sawyer W.: Towards the HPC-inference of causality networks from multiscale economical data*. The work of I. Horenko was partially supported by the Mercator Fellowship off the DFG SPP 1114 "Scaling Cascades in Complex Systems". We would like to gratefully thank Olga Kaiser, Dimitri Igdalov, Anna Marchenko, Benjamin Cumming, Patrick Sanan, Karl Rupp, Susanne Gerber, and PERMON team (David Horák, Václav Hapla et al.) for their collaboration in the project.

REFERENCES

- [1] CSCS Piz Daint supercomputer specification. http://www.cscs.ch/computers/piz_daint/, 2017.
- [2] Dlib C++ toolkit with machine learning algorithms and tools for creating complex software to solve real world problems. <http://dlib.net/>, 2017.
- [3] PERMON - Parallel, Efficient, Robust, Modular, Object-oriented, Numerical software toolbox for solving QP problems. <http://permon.it4i.cz/>, 2017.
- [4] Toolkit for Advance Optimization (TAO) Solver Manual pages. <http://www.mcs.anl.gov/petsc/petsc-3.7/docs/manualpages/Tao/>, 2017.
- [5] N. Agmon, Y. Alhassid, and R. D. Levine. An algorithm for finding the distribution of maximal entropy. *Journal of Computational Physics*, 30:250–258, 1979.
- [6] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. <http://www.mcs.anl.gov/petsc>, 2016.
- [7] J. Barzilai and J. M. Borwein. Two point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.
- [8] D. P. Bertsekas. *Nonlinear Optimization*. Belmont, second edition, April 2004.
- [9] E. G. Birgin, J. M. Martínez, and M. M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10:1196–1211, 2000.
- [10] E. G. Birgin, J. M. Martínez, and M. M. Raydan. SPG: Software for convex-constrained optimization. *ACM Transactions on Mathematical Software*, 27:340–349, 2001.
- [11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 1st edition, 2004.

- [12] E. Candes and M. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25:21–30, 2008.
- [13] A. L. Cauchy. Compte Rendu des Séances de L'Académie des Sciences. *Comptes Rendus Hebd. Seances Acad. Sci.*, 21(25):536–538, 1847.
- [14] Y. Chen and X. Ye. Projection onto a simplex. *Unpublished manuscript, arXiv:1101.6081*, 2011.
- [15] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew. Deep learning with COTS HPC systems. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1337–1345. JMLR Workshop and Conference Proceedings, May 2013.
- [16] Y. Dai and L. Liao. R-linear convergence of the Barzilai and Borwein gradient method. *The Institute of Mathematics and its Applications*, 22(1), 2002.
- [17] Y.-H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100:21–47, 2005.
- [18] Z. Dostál. *Optimal Quadratic Programming Algorithms, with Applications to Variational Inequalities*, volume 23. SOIA, Springer, New York, US, 2009.
- [19] Z. Dostál and L. Pospíšil. Optimal iterative QP and QPQC algorithms. *Annals of Operations Research*, pages 1–14, 2013.
- [20] Z. Dostál and L. Pospíšil. Minimizing quadratic functions with semidefinite Hessian subject to bound constraints. *Computers & Mathematics with Applications*, 70(8):2014–2028, 2015.
- [21] W. Enders. *Applied Econometric Time Series*. John Wiley & Sons Inc, 4th edition, 2014.
- [22] G. Fourestey, B. Cumming, L. Gilly, and T. C. Schulthes. First experiences with validating and using the cray power management database tool. *Cray User Group (CUG) user meeting, arXiv:1408.2657*, 2014.
- [23] S. Gerber and I. Horenko. On inference of causality for discrete state models in a multiscale context. *Proceedings of the National Academy of Sciences*, 111(41):14651–14656, 2014.
- [24] S. Gerber and I. Horenko. Improving clustering by imposing network information. *Science advances*, 1(7), 2015.
- [25] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.
- [26] P. C. Hansen and D. P. O'Leary. The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM Journal on Scientific Computing*, 14:1487–1503, 1993.
- [27] V. Hapla. *Massively Parallel Quadratic Programming Solvers with Applications in Mechanics*. PhD thesis, VSB-TU Ostrava, online: <http://am.vsb.cz/export/sites/am/cs/theses/phd/VaclavHapla.pdf>, 2016.
- [28] V. Hapla, D. Horák, L. Pospíšil, M. Čermák, A. Vašatová, and R. Sojka. Solving contact mechanics problems with PERMON. *High Performance Computing in Science and Engineering: Second International Conference, HPCSE 2015, Solán*, pages 101–115, 2015.
- [29] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1990.
- [30] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.
- [31] I. Horenko. On simultaneous data-based dimension reduction and hidden phase identification. *Journal of the Atmospheric Sciences*, 65(6):1941–1954, 2008.
- [32] I. Horenko. On robust estimation of low-frequency variability trends in discrete markovian sequences of atmospheric circulation patterns. *Journal of the Atmospheric Sciences*, 66(7):2059–2072, 2009.
- [33] I. Horenko. Finite element approach to clustering of multidimensional time series. *SIAM Journal on Scientific Computing*, 32:62–83, 2010.
- [34] I. Horenko. On clustering of non-stationary meteorological time series. *Dynamics of Atmospheres and Oceans*, 49(2–3):164–187, 2010.
- [35] I. Horenko. On the identification of nonstationary factor models and their application to atmospheric data analysis. *Journal of the Atmospheric Sciences*, 67(5):1559–1574, 2010.
- [36] I. Horenko. Nonstationarity in multifactor models of discrete jump processes, memory, and application to cloud modeling. *Journal of the Atmospheric Sciences*, 68(7):1493–1506, 2011.
- [37] I. Horenko. On analysis of nonstationary categorical data time series: Dynamical dimension reduction, model selection, and applications to computational sociology. *Multiscale Modeling & Simulation*, 9(4):1700–1726, 2011.

- [38] A. J. Laub. *Matrix Analysis For Scientists And Engineers*. Society for Industrial and Applied Mathematics, 2014.
- [39] F. Lotte, M. Congedo, A. Lecuyer, F. Lamarche, and B. Arnaldi. A review of classification algorithms for eeg-based brain-computer interfaces. *Journal of Neural Engineering*, 4(2):R1, 2007.
- [40] J. Macke, M. Opper, and M. Bethge. Common input explains higher-order correlations and entropy in a simple model of neural population activity. *Physical Review Letters*, 106(20):208102, 2011.
- [41] P. Metzner, L. Putzig, and I. Horenko. Analysis of persistent non-stationary time series and applications. *Communications in Applied Mathematics and Computational Science*, 7(2):175–229, 2012.
- [42] M. Mudelsee. *Climate time series analysis: Classical statistical and bootstrap methods*. Springer International Publishing, second edition, 2014.
- [43] K. Murphy. Hidden Markov Model (HMM) Toolbox for Matlab. <http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html>, 1998.
- [44] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2003.
- [45] L. Pospíšil. *Development of Algorithms for Solving Minimizing Problems with Convex Quadratic Function on Special Convex Sets and Applications*. PhD thesis, VSB-TU Ostrava, online: http://am.vsb.cz/export/sites/am/cs/theses/pospisil_phd.pdf.
- [46] L. Pospíšil, I. Horenko, P. Gagliardini, and W. Sawyer. PASC-inference, open-source HPC library of nonparametric Bayesian causality inference methods. http://github.com/eth-cscs/PASC_inference, 2017.
- [47] M. M. Raydan. *Convergence properties of the Barzilai and Borwein gradient method*. PhD thesis, Rice University, 1991.
- [48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [49] F. Stimberg, M. Opper, G. Sanguinetti, and A. Ruttor. Inference in continuous-time change-point models. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, page 27172725. 2011.
- [50] L. A. Vese and C. L. Guyader. *Variational Methods in Image Processing*. Chapman & Hall/CRC, 2015.
- [51] G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, 1990.
- [52] Y. Wiaux, L. Jacques, G. Puy, A. M. M. Scaife, and P. Vanderghelynst. Compressed sensing imaging techniques for radio interferometry. *Monthly Notices of the Royal Astronomical Society*, 395(3):1733–1742, 2009.
- [53] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.