An Efficient Implementation for Kernel-based Regularized System Identification with Periodic Input Signals

Shen, Zhuohua; Xu, Yu; Andersen, Martin Skovgaard; Chen, Tianshi

Published in:
Proceedings of 62nd IEEE Conference on Decision and Control

Link to article, DOI:
10.1109/CDC49753.2023.10383860

Publication date:
2023

Document Version
Peer reviewed version

Citation (APA):
Shen, Z., Xu, Y., Andersen, M. S., & Chen, T. (2023). An Efficient Implementation for Kernel-based Regularized System Identification with Periodic Input Signals. In *Proceedings of 62nd IEEE Conference on Decision and Control* (pp. 1480-1485). IEEE. https://doi.org/10.1109/CDC49753.2023.10383860

An Efficient Implementation for Kernel-based Regularized System Identification with Periodic Input Signals

Zhuohua Shen[1], Yu Xu[1], Martin S. Andersen[2], and Tianshi Chen[1]

***Abstract*— Efficient implementation of algorithms for kernel-based regularized system identification is an important issue. The state of art result is based on semiseparable kernels and a class of commonly used test input signals in system identification and automatic control, and with such input signals, the output kernel is semiseparable and exploring this structure gives rise to very efficient implementation. In this paper, we consider instead the periodic input signal, which is another class of commonly used test input signals. Unfortunately, with periodic input signals, the output kernel is NOT semiseparable. Nevertheless, it can be shown that the output kernel matrix is hierarchically semiseparable. Moreover, it is possible to develop efficient implementation of algorithms by exploring the hierarchically semiseparable structure of the output kernel matrix and the periodic and Toeplitz structure of the regression matrix. The efficiency of the proposed implementation of algorithms is demonstrated by Monte Carlo simulations.**

I. Introduction

In the past decade, kernel-based regularized system identification is the major advance in system identification, has achieved many important results and become an emerging new system identification paradigm, see e.g., the survey papers [1], [2] and the book [3]. The key difference between this new paradigm and the classical paradigm based on the maximum likelihood/prediction error methods [4] is two fold. Firstly, the new paradigm finds a systematic way to embed the prior knowledge on the underlying system to be identified in the model structure through a well designed kernel. Secondly, the model complexity can be tuned through the hyper-parameter used to parameterize the kernel in a continuous and more reliable way.

The recent advance on kernel-based regularized system identification includes the kernel design and analysis [5], [6], [7], [8], [9], efficient implementation [10], [11], asymptotic theory [12], [13], [14], and its application in various contexts, e.g., spatial temporal data processing [15] and iterative learning control [16]. In particular, efficient implementation has been an important issue, because it is the key for the engineers to apply this emerging new system identification in the engineering practice. The most widely used implementation [10] has a computational complexity of $\mathcal{O}(Nn^2+n^3)$, where N is the number of data and n is the order of the finite impulse response (FIR) model, and thus is not efficient if n is large, i.e., the underlying system to be identified has a slow dynamics. Recently, an efficient implementation was proposed in [11], and based on semiseparable kernels and a class of commonly used test input signals in system identification and automatic control. With such input signals, the output kernel is semiseparable and exploring this structure gave rise to implementation with computational complexity of $\mathcal{O}(Np^2+p^3)$, where p is the semiseparability rank of the output kernel.

Unfortunately, the implementation proposed in [11] cannot be applied to periodic input signals, which is another class of commonly used test input signals, because the output kernel with general periodic input signals is NOT semiseparable. In this paper, we consider the case with semiseparable kernels and periodic input signals. It will be shown that the output kernel matrix is hierarchically semiseparable. Moreover, it is possible to develop efficient implementation of algorithms by exploring the hierarchically semiseparable structure of the output kernel matrix and the periodic and Toeplitz structure of the regression matrix. The efficiency of the proposed implementation of algorithms is demonstrated by Monte Carlo simulations.

The remaining parts of this paper are organized as follows. In Section II, we introduce some background materials and the problem statement. In Section III, we give the details of our proposed implementations and then in Section IV, we illustrate our implementation by Monte Carlos simulations. Finally, we conclude this paper in Section V.

II. Background and Problem Statement

A. Kernel-based Regularized System Identification with Periodic Inputs and Semiseparable Kernels

In this paper, we consider the identification of linear time-invariant (LTI), discrete-time, causal, and bounded-input bounded-ouptut (BIBO) stable systems described by

$$y(t) = G^0(q)u(t) + v(t), t = 1, \cdots, M, \tag{1}$$

where t is the time index, $M \in \mathbb{N}$ is the number of data, $y(t), u(t)$ and $v(t)$ are the measurement output, input and measurement noise of the system at time t, respectively, and $G^0(q)$ is the unknown transfer function of the system with q being the forward shift operator such that $qu(t) = u(t+1)$, and $v(t)$ is assumed to be white noise with mean zero and variance σ^2.

Since the system is LTI, causal and BIBO stable, its

[1] Zhuohua Shen, Yu Xu, and Tianshi Chen are with the School of Data Science, The Chinese University of Hong Kong, Shenzhen, 518172, Shenzhen, China. {zhuohuashen,yuxu19}@link.cuhk.edu.cn, tschen@cuhk.edu.cn

[2]Martin S. Andersen (mskan@dtu.dk) are with the Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark.

transfer function $G^0(q)$ has the following expansion

$$G^0(q) = \sum_{k=1}^{\infty} g_k^0 q^{-k}, \tag{2}$$

where g_k^0, $k = 1, \cdots, \infty$, are the so-called impulse response coefficients of $G^0(q)$ and absolutely summable, i.e., $\sum_{k=1}^{\infty} |g_k^0| < \infty$. Therefore, the identification of $G^0(q)$ is equivalent to the estimation of the impulse response g_k^0, $k = 1, \cdots, \infty$, which is, however, an intrinsically ill-conditioned problem with finite number of data. One way to overcome this problem is to first propose a parametric model $G(q, \theta)$ with the parameter $\theta \in \mathbb{R}^n$ and then estimate $G(q, \theta)$ based on the data $\{y(t), u(t)\}_{t=1}^N$ with $N \geq n$.

For the kernel-based regularization method (KRM), it chooses $G(q, \theta)$ as the finite impulse response (FIR) model, which is obtained by truncating the infinite impulse response of $G^0(q)$ at a sufficiently high order n as follows

$$G(q, \theta) = \sum_{k=1}^{n} g_k q^{-k}, \theta = [g_1, \cdots, g_n]^T, \tag{3}$$

where g_k, $k = 1, \cdots, n$, are called the finite impulse response coefficients, and then yields

$$y(t) = \sum_{k=1}^{n} g_k u(t-k) + v(t), t = 1, \cdots, N. \tag{4}$$

More specifically, (4) can be rewritten in the following linear regression form

$$Y_N = \Phi_N \theta + V_N \tag{5}$$

where $N = M - n$, $Y_N = [y(n+1), \cdots, y(M)]^T$, $\Phi_N^T = [\phi(n)^T, \cdots, \phi(M)^T]$ with $\phi(k) = [u(k), \cdots, u(k-n+1)]$, $k = n, n+1, \cdots, M$, and $V_N = [v(n+1), \cdots, v(M)]^T$. Then θ can be estimated by minimizing the following kernel-based regularized least squares (RLS) criterion:

$$\begin{aligned} \hat{\theta}_N &= \arg\min_{\theta} \|Y_N - \Phi_N \theta\|^2 + \gamma \theta^T K(\alpha)^{-1} \theta \\ &= \left(\Phi_N^T \Phi_N + \sigma^2 K(\alpha)^{-1}\right)^{-1} \Phi_N^T Y_N \end{aligned} \tag{6}$$

where $\|\cdot\|$ is the Euclidean norm, $\gamma > 0$ is the regularization parameter, $K(\alpha) \in \mathbb{R}^{n \times n}$ is the so-called *kernel* matrix and defined through a positive semidefinite kernel [17] $\kappa(t, s; \alpha) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$, and $\alpha \in \mathbb{R}^p$ is the so-called hyper-parameter used to parameterize the kernel $\kappa(t, s; \alpha)$.

In practice, one needs to first design a suitable kernel $\kappa(t, s; \alpha)$, then estimate both γ and α, and finally, get the RLS estimate $\hat{\theta}_N$. By far the most effective method to estimate α is the so-called empirical Bayes method. This method further assumes that the measurement noise $v(t)$ is Gaussian, $\theta \sim N(0, K(\alpha))$, and moreover, θ is independent of $v(t)$, $t = 1, \cdots, N$, and $\gamma = \sigma^2$. Under these assumptions, it is easy to verify that the RLS estimate $\hat{\theta}_N$ in (6) is also the maximum a posterior estimate of θ and the hyper-parameter α is estimated by maximizing the marginal likelihood of α, i.e., $\hat{\alpha} \triangleq \arg\max_{\alpha} p(Y_N \mid \alpha)$ or equivalently,

$$\begin{aligned} \hat{\alpha} &= \arg\min_{\alpha} Y_N^T H(\alpha)^{-1} Y_N + \log |H(\alpha)| \\ H(\alpha) &= \Phi_N K(\alpha) \Phi_N^T + \sigma^2 I_N, \end{aligned} \tag{7}$$

where $|\cdot|$ is the determinant of a matrix, I_N is the N-dimensional identity matrix, and $H(\alpha)$ is often called the output kernel matrix.

The straightforward computation of $\hat{\theta}_N$ in (6) and $\hat{\alpha}$ in (7) has computational complexity of $\mathcal{O}(N^3)$. In order to apply the KRM in practice with large N, several efficient implementations have been developed over the past decade, e.g., [18], [10], [11]. In this paper, we focus on this topic under the following assumption.

Assumption 1. *Assume that the input signal $u(t)$ is periodical with period $p \in \mathbb{N}$, and the kernel matrix $K(\alpha)$ is extended $\{k\}$-generator representable semiseparable with $k \in \mathbb{N}$, and moreover, $N \geq n \geq p$.*

1) Periodic Input Signals: On the one hand, it is worth to mention that periodic input signals are one class of most widely used test signals in system identification and control, see e.g., [4]. Moreover, due to (4), Φ_N in (5) is a Toeplitz matrix, i.e.,

$$\Phi_N(i, j) = \Phi_N(i+t, j+t) \tag{8}$$

for all $1 \leq i, i+t \leq N$, $1 \leq j, j+t \leq n$, $t \in \mathbb{Z}$, and under Assumption 1, Φ_N in (5) has a periodical structure with period $p \in \mathbb{N}$, i.e.,

$$\Phi_N(i, j) = \Phi_N(i + t_1 p, j + t_2 p) \tag{9}$$

for all $1 \leq i, i+t_1 p \leq N$, $1 \leq j, j+t_2 p \leq n$, and $t_1, t_2 \in \mathbb{Z}$, and Φ_N can be rewritten as follows

$$\Phi_N = \begin{bmatrix} \Phi_b & \Phi_b & \cdots & \Phi_{bc} \\ \Phi_b & \Phi_b & \ddots & \Phi_{bc} \\ \vdots & \vdots & \vdots & \vdots \\ \Phi_{br} & \Phi_{br} & \cdots & \Phi_{bb} \end{bmatrix} \tag{10}$$

where $\Phi_b = \Phi_N(1:p, 1:p)$, $\Phi_{br} = \Phi_N(1:N\%p, 1:n\%p)$, $\Phi_{bc} = \Phi_N(1:p, 1:n\%p)$, $\Phi_{bb} = \Phi_N(1:N\%p, 1:n\%p)$, "%" represents the modulo operation, and $\Phi_N(1:a, 1:b)$ with $a, b \in \mathbb{N}$ the submatrix of Φ_N consisting of the first a rows and b columns of Φ_N.

2) Semiseparable Kernels: On the other hand, most of the existing kernels proposed in KRM are semiseparable, see e.g., [11]. As a result, the kernel matrix $K(\alpha)$ is extended $\{k\}$-generator representable semiseparable for some $k \in \mathbb{N}$. Recall that a symmetric matrix $K \in \mathbb{R}^{n \times n}$ is said to be extended $\{k\}$-generator representable semiseparable [19, p.304] if there exists $k \in \mathbb{N}, k \leq n$ such that

$$K = S(U, V) = \text{tril}(UV^T) + \text{triu}(VU^T, 1) \tag{11}$$

where $U, V \in \mathbb{R}^{n \times k}$ are called *generators* of K, $\text{tril}(UV^T)$ denotes the lower-triangular matrix generated by UV^T by zeroing all its (i, j) entries with $j > i$, and $\text{triu}(VU^T, 1)$ the upper-triangular matrix generated by VU^T by zeroing all its (i, j) entries with $i > j - 1$.

B. Problem Statement

In this paper, we study the problem of how to develop more efficient implementation algorithms for the computation of $\hat{\alpha}$ in (7) (and also $\hat{\theta}_N$ in (6), but it is skipped, because the space is limited and its computation is a byproduct of the computation of $\hat{\alpha}$ in (7)) with lower computational complexity than the existing ones, e.g., [18], [10], [11], under Assumption 1 by making use of the structural properties of Φ_N and $K(\alpha)$ as sketched in Sections II-A.1 and II-A.2, and exploring the structural properties of $H(\alpha)$ defined in (7).

Remark II.1. *As can be seen from [11], the computation of the cost function of the empirical Bayes method* (7) *shares some common elements as that of the other hyper-parameter estimation methods, such as the Stein's unbiased risk estimation (SURE) method and the generalized cross validation (GCV) method. Therefore, the proposed implementation algorithms can be used to develop efficient implementation algorithms also for the SURE and GCV methods.*

C. Fundamentals of Efficient Implementation by Exploring Hierarchically Semiseparable Matrix and Toeplitz Matrix

In this subsection, we introduce some fundamentals of efficient implementation by exploring the structure of hierarchically semiseparable matrix and Toeplitz matrix.

1) Hierarchically Semiseparable (HSS) Matrix: As will be shown shortly in Proposition 2, $H(\alpha)$ in (7) is actually an HSS matrix. HSS matrices are a generalization of semiseparable matrices, and also a special case of hierarchically off-diagonal low-rank (HODLR) matrices[20], [21], [22]. HODLR matrices have low-rank off-diagonal blocks, while the diagonal blocks can be subdivided into the same form recursively, i.e., they also have low-rank off-diagonal blocks. The HSS matrices have the additional property that the off-diagonal blocks can depend on the off-diagonal blocks in the deeper level's diagonal blocks [22]. More specifically, if $A \in \mathbb{R}^{N\times N}$ is a two-level HODLR matrix, then A can be represented as

$$A = \begin{bmatrix} A_1^{(1)} & U_1^{(1)} K_{12}^{(1)} {V_2^{(1)}}^T \\ U_2^{(1)} K_{21}^{(1)} {V_1^{(1)}}^T & A_2^{(1)} \end{bmatrix} \tag{12}$$

where the off-diagonal blocks $U_1^{(1)} K_{12}^{(1)} {V_2^{(1)}}^T$ and $U_2^{(1)} K_{21}^{(1)} {V_1^{(1)}}^T$ are of low-rank, and $A_1^{(1)}, A_2^{(1)}$ can be further divided into

$$\begin{aligned} A_1^{(1)} &= \begin{bmatrix} A_1^{(2)} & U_1^{(2)} K_{12}^{(2)} {V_2^{(2)}}^T \\ U_2^{(2)} K_{21}^{(2)} {V_1^{(2)}}^T & A_2^{(2)} \end{bmatrix} \\ A_2^{(1)} &= \begin{bmatrix} A_3^{(2)} & U_3^{(2)} K_{34}^{(2)} {V_4^{(2)}}^T \\ U_4^{(2)} K_{43}^{(2)} {V_3^{(2)}}^T & A_4^{(2)} \end{bmatrix} \end{aligned} \tag{13}$$

which is in the same structure as A until it reaches the size or rank threshold [23]. For HSS matrices, the matrices $U_1^{(1)}, U_2^{(1)}, V_1^{(1)}, V_2^{(1)}$ can further depend on $U_i^{(2)}$ and $V_j^{(2)}$ recursively through the so-called *translation operators*. For example, there are translation operators $R_k^{(2)}, W_k^{(2)}, k = 1, 2, 3, 4$ such that

$$U_i^{(1)} = \begin{bmatrix} U_{2i-1}^{(2)} R_{2i-1}^{(2)} \\ U_{2i}^{(2)} R_{2i}^{(2)} \end{bmatrix}, \; V_i^{(i)} = \begin{bmatrix} V_{2i-1}^{(2)} W_{2i-1}^{(2)} \\ V_{2i}^{(2)} W_{2i}^{(2)} \end{bmatrix}, \; i = 1, 2 \tag{14}$$

It is similar for other U_i, V_i's at each level [20].

Both HODLR and HSS structures have been widely applied in many fields to design fast algorithms, including Gaussian Process regression [24], [25], radial basis function interpolation [26], and sparse linear system solvers [27]. Concerning the specific linear algebra operations, exploring the structure of HODLR and HSS matrices allows efficient implementation of algorithms for basic arithmetical operations [28], e.g., addition, matrix-matrix multiplication, inversion, and decompositions, e.g., QR, LU, Cholesky.

2) Periodic Toeplitz Matrices: The regression matrix Φ_N in (5) is not only Toeplitz but also periodic. As well known, some operations of Toeplitz matrices can be calculated efficiently. For example,

- the Toeplitz matrix-vector product can be calculated efficiently by using Fast Fourier Transform (FFT). Acutally, for a Toeplitz matrix $A \in \mathbb{R}^{n\times n}$ and a vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$, direct calculation of $A\mathbf{x}$ has computational complexity of $\mathcal{O}(n^2)$, but by using the convolution rule and FFT, the computation cost is reduced to $\frac{9}{2} n \log n = \mathcal{O}(n \log n)$ flops, see e.g., [29].
- the thin QR decomposition of a Toeplitz matrix can also be calculated efficiently. Let $A \in \mathbb{R}^{m\times n}$ be a Toeplitz matrix with $m \geq n$, and its thin QR decomposition denoted by $A = QR$, where $Q \in \mathbb{R}^{m\times n}$ with $Q^T Q = I_n$ and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. Then the direct calculation of the thin QR decomposition of A with both Q and R requires $2mn^2$ flops, but by exploring the structure of the Toeliptz matrix, Qiao proposed an algorithm in [30] based on the one in [31], which requires $mn + 6.5n^2$ flops for R and $13mn + 6.5n^2$ flops for both Q and R.

Moreover, as will be shown shortly in Sections III-B and III-C, it is possible to develop more efficient implementations of algorithms for both the full and thin QR decompositions of Φ_N by further exploring its periodic structure.

III. How to explore the structure

We show how to develop efficient implementation algorithms by exploring the structure of (7) and in particular, the structure of the output kernel matrix $H(\alpha)$ to compute $Y_N^T H(\alpha)^{-1} Y_N$ and $\log |H(\alpha)|$ efficiently.

A. Preamble

We will introduce two different routes, whose key ideas are sketched below.

1) Route 1: First, we can construct the HSS structure for $H(\alpha)$ by selecting an appropriate level and the order of the deepest level according to the period p. Then, followed by the general framework of efficient algorithms to factorize the HSS $H(\alpha)$, we propose modified Householder and Given full QR decomposition applying the periodic structure. At the last (l^{th}) step, for changing α, the semi-separable structure of $K(\alpha)$ is used to compute the explicit $D(\alpha)$. The framework and precise costs of computing $Y_N^T H(\alpha)^{-1} Y_N$ and $\log|H(\alpha)|$ are discussed.

First, we show that when Φ_N is periodic, the output kernel matrix $H(\alpha)$ has a HSS structure. Then by referring to [20], [28], we modify a fast full QR decomposition to factorize $H(\alpha)$ into different levels such that for $k = l-1, l-2, \ldots, 2, 1$, we have

$$H^{(k)}(\alpha) = W^{(k)} P^{(k)} \begin{bmatrix} H^{(k-1)} & \\ & \sigma^2 I_{2^k p} \end{bmatrix} P^{(k)^T} W^{(k)^T}, \tag{15}$$

where k is the level index, l is the deepest level chosen for optimal computational complexity, $W^{(k)}$ is an orthogonal transformation, and $P^{(k)}$ is a permutation matrix. Here we try two methods for orthogonalization: Householder and Givens QR decompositions. When the factorization is finished at the l^{th} step,

$$H^{(0)} = W^{(0)} P^{(0)} \begin{bmatrix} D(\alpha) & \\ & \sigma^2 I_p \end{bmatrix} P^{(0)^T} W^{(0)^T} \in \mathbb{R}^{2p \times 2p} \tag{16}$$

where $D(\alpha) = R^{(0)} K(\alpha) R^{(0)^T} + \sigma^2 I_p \in \mathbb{R}^{p\times p}$, for which the explicit form is computed efficiently using the semi-separable structure of $K(\alpha)$. In $H(\alpha)$, only $D(\alpha)$ changes as the parameter α changes. Since Y_N is fixed, all the computations in $Y_N^T H(\alpha)^{-1} Y_N$ except the one related to $D(\alpha)$ are only needed to be done once; since the determinant of orthogonal matrices is zero, $\log|H(\alpha)|$ is almost about $|D(\alpha)|$. The cost of computations related to the changing part $D(\alpha)$ is free of N, which is much smaller when $N \gg n, p$.

2) Route 2: In the second route, we present the following steps to compute $Y_N^T H(\alpha)^{-1} Y_N$ and $\log|H(\alpha)|$:

i) Adapt the thin QR decomposition algorithm [30] to compute $R \in \mathbb{R}^{p\times n}$ with $\Phi_N = QR, Q \in \mathbb{R}^{N\times p}$.
ii) Apply the Matrix Inversion Lemma to obtain
$$\begin{aligned} H^{-1} &= \sigma^{-2}(I_N - \Phi_N \bar{H} \Phi_N^T) \quad (17)\\ \bar{H} &= \sigma^{-2}(K - KR^T(\sigma^2 I_p + RKR^T)^{-1} RK^T), \end{aligned}$$
where for simplicity, we omit the dependence of $H, K \bar{H}$ of α.
iii) Compute $(\sigma^2 I_p + RKR^T)^{-1}$.
iv) $Y_N^T H(\alpha)^{-1} Y_N$: first multiply vector Y_N and Y_N^T from the right and left of (17) and use FFT to obtain $\Phi_N^T Y_N$, and then use semi-separable structure of K to finish the computation of the other parts.
v) $\log|H|$: rewrite it as
$$(N-p)\log(\sigma^2) + \log\left|\sigma^2 I_p + RKR^T\right|. \tag{18}$$
The second term is computed from step iii).

B. Route 1: Efficient implementation by exploring HSS structure of $H(\alpha)$ and periodic structure of Φ_N

1) HSS Structure: We first show that $H(\alpha)$ is an HSS matrix. Since Φ_N is periodic, let the *deepest level* be l (the choice for l will be discussed later), divide it into 2^l parts according to rows:

$$\Phi_N = \begin{bmatrix} \Phi_1^{(l)^T} & \Phi_2^{(l)^T} & \ldots & \Phi_{2^l}^{(l)^T} \end{bmatrix}^T \tag{19}$$

where for $i = 1, \ldots, 2^l - 1$,

$$\Phi_i^{(l)} = \begin{bmatrix} \Phi_b & \cdots & \Phi_{bc} \\ \vdots & \vdots & \vdots \\ \Phi_b & \cdots & \Phi_{bc} \end{bmatrix} \in \mathbb{R}^{m\times n} \tag{20}$$

$$\Phi_{2^l}^{(l)} = \begin{bmatrix} \Phi_b & \cdots & \Phi_{bc} \\ \vdots & \vdots & \vdots \\ \Phi_{br} & \cdots & \Phi_{bb} \end{bmatrix} \in \mathbb{R}^{r\times n}, \tag{21}$$

m is a multiple of p, r satisfies $2^{l-1}m + r = N$, normally $m \approx r, 2^l m \approx N$, and m is not necessarily larger than n. Then H (α is sometimes omitted for simplicity) becomes

$$H = \begin{bmatrix} \Phi_1^{(l)} K \Phi^{(l)^T} & \Phi_1^{(l)} K \Phi_2^{(l)^T} & \cdots & \Phi_1^{(l)} K \Phi_{2^l}^{(l)^T} \\ \Phi_2^{(l)} K \Phi_1^{(l)^T} & \Phi_2^{(l)} K \Phi_2^{(l)^T} & \cdots & \Phi_2^{(l)} K \Phi_{2^l}^{(l)^T} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_{2^l}^{(l)} K \Phi_1^{(l)^T} & \Phi_{2^l}^{(l)} K \Phi_2^{(l)^T} & \cdots & \Phi_{2^l}^{(l)} K \Phi_{2^l}^{(l)^T} \end{bmatrix} + \sigma^2 \mathrm{diag}(I_m, I_m, \ldots, I_r), \tag{22}$$

which is an HSS matrix, as shown in the proposition below.

Proposition 2. *Suppose that Assumption 1 holds. Then H in* (7) *or* (22) *is a HSS matrix with identity translation operator.*

According to [21], [22], [28], it is possible to develop fast factorization of H according to its HSS structure, and the factorization is based on the full QR decomposition of Φ_N. In the next section, we show that by exploring the periodic structure of Φ_N, it is possible to propose an efficient full QR decomposition algorithm of Φ_N.

2) Efficient QR Decomposition for Periodic Φ_N: We first consider the QR decomposition:

$$\Phi_N(1:N, 1:p) = \begin{bmatrix} \Phi_b \\ \vdots \\ \Phi_b \\ \Phi_{br} \end{bmatrix} = Q \begin{bmatrix} R_b \\ 0_{(N-p)\times p} \end{bmatrix}, \tag{23}$$

where $Q \in \mathbb{R}^{N\times N}$ with $QQ^T = Q^T Q = I_N$, and $R_b \in \mathbb{R}^{p\times p}$. Then Q^T can be applied to the whole Φ_N:

$$Q^T \Phi_N = \begin{bmatrix} R_b & .. & R_b & R_{bc} \\ 0_{(N-p)\times p} & .. & 0_{(N-p)\times p} & 0_{(N-p)\times(n\%p)} \end{bmatrix} \triangleq \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{24}$$

where $R_{bc} = \begin{bmatrix} R_b(1:n\%p, 1:n\%p) \\ 0_{(p-n\%p)\times(n\%p)} \end{bmatrix}$, and $R \in \mathbb{R}^{p\times n}$ is dense only in the upper triangular part of each R_b and R_{bc}. Then the following proposition shows that $Q(1:N, 1:p)$ is also periodic with period p.

Proposition 3. *Suppose that Assumption 1 holds and $rank(\Phi_p) = p$. Then it holds that*

$$Q(i,j) = Q(i+tp, j) \, for \, 1 \le i, i+tp \le N, 1 \le j \le p, t \in \mathbb{Z}.$$

Proposition 3 can be applied to efficiently solve the least-square problems. In the following, we consider the QR decomposition based on the Householder transformation because of its nice properties when factorizing periodic Φ_N.

By referring to [29, Algorithms 5.1.1 and 5.2.1], we propose an adapted algorithm to compute the QR decomposition in (23):

1) **Algorithm 1** computes the Householder transformation of a given periodic vector $\mathbf{x} \in \mathbb{R}^N$. Since the Householder vector $\mathbf{v} \in \mathbb{R}^N$ is periodic except the first entry, we set $\mathbf{v}_b$ and constant β as outputs such that
   - $\tilde{\mathbf{v}}_b = \mathbf{v}_b(2:p+1)$;
   - $\mathbf{v} = [\mathbf{v}_b(1)\ \underbrace{\tilde{\mathbf{v}}_b^T \ \cdots\ \tilde{\mathbf{v}}_b^T}_{m \text{ times}}\ \tilde{\mathbf{v}}_b^T(1:r)]^T$, where $m = \lfloor (N-1)/p \rfloor$, $r = (N-1)\%p$.
   - The transformation can be expressed as $H_{\mathbf{v}} = I_N - \beta \mathbf{v}\mathbf{v}^T$.
2) **Algorithm 2** computes the Householder QR decomposition of the given periodic matrix $\Phi_N(1:N, 1:p) \in \mathbb{R}^{N\times p}$. The outputs are: upper triangular matrix $R_b \in \mathbb{R}^{p\times p}$; vectors $\mathbf{v}_b^{(k)}$ and constants $\beta^{(k)}$, $k = 1, \ldots, p$. Then the Q in (23) can be represented by following:
   - $\tilde{\mathbf{v}}_b^{(k)} = \mathbf{v}_b^{(k)}(2:p+1)$.
   - $\mathbf{v}^{(k)} = [\mathbf{v}_b^{(k)}(1)\ \underbrace{\tilde{\mathbf{v}}_b^{(k)T} \ \cdots\ \tilde{\mathbf{v}}_b^{(k)T}}_{m^{(k)} \text{ times}}\ \tilde{\mathbf{v}}_b^{(k)T}(1:r^{(k)})]^T$, where $m^{(k)} = \lfloor (N-k)/p \rfloor$, $r^{(k)} = (N-k)\%p$.
   - $H_{\mathbf{v}_k} = I_{N-k+1} - \beta^{(k)}(\mathbf{v}^{(k)})(\mathbf{v}^{(k)})^T$
   - $Q = \prod_{k=1}^{p} \operatorname{diag}(I_{k-1}, H_{\mathbf{v}_k})$, where $\operatorname{diag}(A, B)$ means the diagonal matrix generated by square blocks A and B.

All the outputs in both algorithms are represented using their periodic part, as we use Φ_b to represent Φ_N. It is because (7) can be computed by only using the periodic part rather than explicitly forming the Q and R in (24). The computational costs of the two algorithms as well as matrix-vector product are shown in the following proposition.

Proposition 4. *Given a periodic vector $\mathbf{v} \in \mathbb{R}^N$ and a periodic matrix $\Phi_N \in \mathbb{R}^{N\times n}$, both having the period p, The cost of doing Householder QR decomposition with periodic part-only outputs for Φ_N using is about $2p^2(p-1) = \mathcal{O}(p^3)$ flops. Besides, given a vector $\mathbf{x} \in \mathbb{R}^N$, $Q^T\mathbf{x}$ costs about $3Np = \mathcal{O}(Np)$ flops,*

3) Computational Complexity by Exploring the HSS structure: According to [21], [22], [28], the computational cost for computing the factorization of $H(\alpha)$ by exploring the HSS structure and Householder QR decomposition using **Algorithm 1** and **Algorithm 2** is given in the theorem below.

Theorem 5. *Suppose that Assumption 1 holds and the deepest level of $H(\alpha)$ is l. Then the computational complexity of*

Algorithm 1 Householder transformation for a periodic vector (House)

```
Input: Periodic part $\mathbf{x}_b = \mathbf{x}(1:p)$, size $N$.
Output: First $p+1$ elements of Householder vector $\mathbf{v}_b = \mathbf{v}(1:p+1)$, $\beta$.
 1: $\mathbf{x}'_b \leftarrow \begin{bmatrix} \mathbf{x}_b(2:p) \\ \mathbf{x}_b(1) \end{bmatrix}$;
 2: $m \leftarrow \left\lfloor \frac{N-1}{p} \right\rfloor$; $r \leftarrow (N-1)\%p$;
 3: $\sigma \leftarrow m(\mathbf{x}'_b)^T\mathbf{x}'_b + (\mathbf{x}'_b)^T(1:r)\mathbf{x}'_b(1:r)$;
 4: $\mathbf{v}_b \leftarrow \begin{bmatrix} 1 \\ \mathbf{x}'_b \end{bmatrix}$;
 5: if $\sigma = 0$ then
 6:     $\beta \leftarrow \sigma$;
 7: else
 8:     $\mu \leftarrow \sqrt{(\mathbf{x}_b(1))^2 + \sigma}$;
 9:     if $\mathbf{x}_b(1) \le 0$ then
10:         $\mathbf{v}_b(1) \leftarrow \mathbf{x}_b(1) - \mu$
11:     else
12:         $\mathbf{v}_b(1) \leftarrow \frac{-\sigma}{\mathbf{x}_b(1)+\mu}$;
13:     end if
14:     $\beta \leftarrow \frac{2(\mathbf{v}_b(1))^2}{\sigma + (\mathbf{v}_b(1))^2}$;
15:     $\mathbf{v}_b \leftarrow \frac{\mathbf{v}_b}{\mathbf{v}_b(1)}$;
16: end if
17: $\tilde{\mathbf{v}}_b \leftarrow \mathbf{v}_b(2:p+1)$;
18: $\mathbf{v} \leftarrow [\mathbf{v}_b^T(1)\ \underbrace{\tilde{\mathbf{v}}_b^T \ \cdots\ \tilde{\mathbf{v}}_b^T}_{m \text{ times}}\ \tilde{\mathbf{v}}_b^T(1:r)]^T$.
```

Algorithm 2 Householder QR decomposition for a periodic matrix

```
Input: Periodic part $\Phi_b = \Phi_{trun}(1:p, 1:p)$, size $N$.
Output: First $p+1$ elements of each Householder vector $\mathbf{v}_b^{(k)} = \mathbf{v}^{(k)}(1:p+1), \beta^{(k)}, R_b$.
 1: $\Phi_b^{(1)} \leftarrow \Phi_b$;
 2: for $k := 1$ to $p$ do
 3:     $n^{(k)} \leftarrow p - (k-1)$;
 4:     $m^{(k)} \leftarrow \left\lfloor \frac{N-k}{p} \right\rfloor$; $r^{(k)} \leftarrow (N-k)\%p$;
 5:     $[\mathbf{v}_b^{(k)}, \beta^{(k)}] \leftarrow \text{House}(\Phi_b^{(k)}(1:p,1), N-(k-1))$;
 6:     $\Phi_b'^{(k)} \leftarrow \begin{bmatrix} \Phi_b^{(k)}(2:p, 1:n^{(k)}) \\ \Phi_p^{(k)}(1, 1:n^{(k)}) \end{bmatrix}$;
 7:     $S_f^{(k)} \leftarrow \mathbf{v}_b^{(k)}(1)\Phi_b^{(k)}(1, 1:n^{(k)})$;
 8:     $S_m^{(k)} \leftarrow m^{(k)}(\mathbf{v}_b^{(k)})^T(2:p+1)\Phi_b'^{(k)}$;
 9:     $S_r^{(k)} \leftarrow (\mathbf{v}_b^{(k)})^T(1:r+1)\Phi_b'^{(k)}(1:r, 1:n^{(k)})$;
10:     $S^{(k)} \leftarrow S_f^{(k)} + S_m^{(k)} + S_r^{(k)}$;
11:     $T_b^{(k)} \leftarrow \beta^{(k)}\mathbf{v}_b^{(k)}S^{(k)}$;
12:     $\tilde{\Phi}_b'^{(k)} \leftarrow \begin{bmatrix} \Phi_b^{(k)}(1, 1:n^{(k)}) - T_b^{(k)}(1, 1:n^{(k)}) \\ \Phi_b'^{(k)} - T_b^{(k)}(2:p+1, 1:n^{(k)}) \end{bmatrix}$;
13:     $R_b(k, k:p) \leftarrow \tilde{\Phi}_b'^{(k)}(1, 1:n^{(k)})$;
14:     if $k \ne p$ then
15:         $\Phi_b^{(k+1)} \leftarrow \tilde{\Phi}_b'^{(k)}(2:p+1, 2:n^{(k)})$;
16:     end if
17: end for
```

computing both $Y_N^T H(\alpha)^{-1} Y_N$ and $\log|H(\alpha)|$ is

$$\underbrace{[3Np + (\frac{7}{3} + \frac{5}{3}l)p^3 + (12\cdot 2^l - 2l - 8)p^2]}_{\textit{irrespective of } \alpha} + \underbrace{f(n,p,k)}_{\textit{depends on } \alpha} \tag{25}$$

where $f(n,p,k)$ is the cost of computing the part dependent on α and and the lower-order terms such as $\mathcal{O}(p)$ is ignored. Moreover, the computational complexity (25) *is minimized at $l = 0$, and the corresponding one becomes*

$$\underbrace{[3Np + \frac{7}{3}p^3 + 4p^2]}_{\textit{irrespective of } \alpha} + \underbrace{f(n,p,k)}_{\textit{depends on } \alpha}. \tag{26}$$

Theorem 5 shows that to have the most efficient implementation, it is enough to keep $H(\alpha)$ as a whole and just compute the full Householder QR decomposition for Φ_N. As a result, we propose to compute $Y_N^T H(\alpha)^{-1} Y_N$ and $\log|H(\alpha)|$ in the following four steps:

i) Compute full QR decomposition of $\Phi_N = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$ using **Algorithm 2**, $Q \in \mathbb{R}^{N\times N}, R \in \mathbb{R}^{p\times n}$ is periodic. Then

$$H(\alpha) = Q\begin{bmatrix} D(\alpha) & \\ & \sigma^2 I_{N-p} \end{bmatrix} Q^T \tag{27}$$

where $D(\alpha) = RK(\alpha)R^T + \sigma^2 I_p$.

ii) Compute $\bar{Y}_N = Q^T Y_N$. Let

$$\bar{Y}_N = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix},\ Y_1 = \bar{Y}_N(1:p),\ Y_2 = \bar{Y}_N(p+1:N).$$

Compute $S_2 = \sigma^{-2} Y_2^T Y_2$.

iii) Compute explicit $D(\alpha)$ using the semiseparable structure of $K(\alpha)$:
   - Compute KR^T, each matrix-vector multiplication of $KR^T(1:n,i), i = 1,\dots,p$ is obtained by [32, Algorithm 4.1.], which provides an efficient way to calculate semiseparable matrix-vector product.
   - Compute $R(KR^T) + \sigma^2 I_p$ using the normal computation way.

Then compute the inverse $D(\alpha)^{-1}$. We have

$$S_1(\alpha) = Y_1^T D(\alpha)^{-1} Y_1 \tag{28}$$

The result of $Y_N^T H(\alpha) Y_N = S_1(\alpha) + S_2$.

iv) Compute

$$\log|H(\alpha)| = (N-p)\log(\sigma^2) + \log|D(\alpha)| \tag{29}$$

directly using explicit $D(\alpha)$.

It is worth to note that the first two steps, i.e., i)-ii), are irrespective of α, but the last two steps depend on α. The computational cost $f(n,p,k)$ in Theorem 5 of the last two steps, i.e., iii)-iv), is shown in the next section.

4) Computation Complexity $f(n,p,k)$ of (28) *and* (29)*:* Since K changes as α changes, each update of α in the solution of (7) requires the re-computation of (28) and (29) with fixed R and their direct computation need $\mathcal{O}(n^2p)$ flops. As will be shown in the following theorem, it is possible to reduce the computational complexity of (28) and (29) by exploring the semiseparable structure of K in (11).

Proposition 6. *Suppose that Assumption 1 holds and let $K \in \mathbb{R}^{n\times n}$ be a semiseparable matrix in the form of* (11) *with the semiseparability rank k, and $R \in \mathbb{R}^{p\times n}$ be the QR decomposition of Φ_N. Then, it holds that*

- *the computation cost of* (28) *is $8npk + np^2 + \frac{4}{3}p^3$;*
- *the computation cost of* (29) *is $\mathcal{O}(p^3)$;*

and hence

$$f(n,p,k) = [8npk + np^2 + \frac{4}{3}p^3] + [\mathcal{O}(p^3)]. \tag{30}$$

5) Overall Computational Complexity:

Theorem 7 (Complexity of *Route 1*)**.** *Following* (26)*, the total cost for both $Y_N^{-1} H(\alpha)^{-1} Y_N$ and $\log|H(\alpha)|$ is*

$$\underbrace{[3Np + \frac{7}{3}p^3 + 4p^2]}_{\textit{i), ii): fixed part}} + \underbrace{[8npk + np^2 + \frac{4}{3}p^3] + [\mathcal{O}(p^3)]}_{\textit{iii), iv): changing part, } f(n,p,k)}. \tag{31}$$

C. Route 2: Efficient implementation by using FFT and thin QR decomposition of Φ_N

In this section, we show that it is possible to reduce the computation cost of (7) by using the FFT based Toeplitz matrix-vector product and thin QR decomposition of Φ_N. In particular, we propose to compute $Y_N^T H(\alpha)^{-1} Y_N$ and $\log|H(\alpha)|$ in the following four steps:

i) Use **Algorithm 3** to compute the thin QR decomposition of $\Phi_N(1:N, 1:p) = QR_b$, $Q \in \mathbb{R}^{N\times p}$, $R_b \in \mathbb{R}^{p\times p}$. We only need to compute R_b and then compute

$$R := [R_b\ R_b\ \cdots\ R_{bc}] \in \mathbb{R}^{p\times n}$$

where the size is the same as the first p rows of (24), $Q^T\Phi_N = R$. By the relation $\Phi_N^T\Phi_N = R^T R$, H^{-1} can be expressed as

$$\begin{aligned} H^{-1} &= \sigma^{-2}(I_N - \Phi_N \bar{H} \Phi_N^T) \\ \bar{H} &= \sigma^{-2}(K - KR^T(\sigma^2 I_p + RKR^T)^{-1} RK^T). \end{aligned}$$

.

ii) Compute $\mathbf{z}_n := \Phi_N^T Y_N$ using FFT [29], and $S_1 = Y_N^T Y_N$.

iii) (Changing part for $Y_N^T H^{-1} Y_N$) Compute $S_2(\alpha) = \mathbf{z}_n^T \bar{H} \mathbf{z}_n$.
   - $s_{21}(\alpha) := \mathbf{z}_n^T K \mathbf{z}_n$ using [32, Algorithm 4.1.];
   - $\hat{\mathbf{z}}_n := RK^T\mathbf{z}_n = RK\mathbf{z}_n$;
   - Directly compute $D' = RKR^T + \sigma^2 I_p$ using [32, Algorithm 4.1.], which is essentially the same as $D(\alpha)$ in *Route 1*;
   - $s_{22}(\alpha) := \hat{\mathbf{z}}_n^T D'^{-1} \hat{\mathbf{z}}_n$.

– Final result of $Y_N^T H^{-1} Y_N = \sigma^{-2}(S_1 - S_2(\alpha)), S_2(\alpha) = \sigma^{-2}(s_{21}(\alpha) - s_{22}(\alpha))$.

(iv) (Changing part for log-determinant) We have the formula

$$\log|H| = (N-p)\log(\sigma^2) + \log|D'|$$

by Sylvester's determinant theorem. The computation directly uses explicit D' which has been obtained in iii).

Now we analyze the complexity of each step.

*1) Cost of step 1 (**Algorithm 3**):* First, before analyzing the cost of **Algorithm 3**, we should consider the periodic Toeplitz matrix-vector multiplication, e.g. $\mathbf{z}_n = \Phi_N^T Y_N$, where $\mathbf{z}_n$ has period p.

Proposition 8. *Let $\Phi_N \in \mathbb{R}^{N\times n}$ be a periodic Toeplitz matrix with period p, $Y_N \in \mathbb{R}^N$, then $\mathbf{z}_n = \Phi_N^T Y_N$ has period p, and computing the periodic part $\mathbf{z}_b := \mathbf{z}_n(1:p)$ costs $\frac{9}{2}N\log p$ flops. Moreover, if Y_N also has period p, then the cost is $9p\log p$.*

Since Qiao's hybrid algorithm achieves both good computational and numerical performance in thin QR decomposition for Toeplitz matrix, we adapt Qiao's algorithm with additional periodic structure, to compute $\Phi_N(1:N,1:p) = QR_b$ [30]. Direct usage of Qiao's method costs $Np + 6.5p^2$, but the modified version can further reduce the cost.

Proposition 9. *The cost for computing $R_b \in \mathbb{R}^{p\times p}$ in $\Phi_N(1:N,1:p) = QR_b$ is $9p\log p + 6.5p^2$ flops.*

For details about the notations, see the appendix.

Remark III.1. *Similarly, by computing only the periodic part in the hybrid algorithm, the cost of obtaining thin $Q \in \mathbb{R}^{N\times p}$, which satisfies $T = QR_b$, can also be reduced from $12Np$ to $12p^2$. The total cost is $9p\log p + 18.5p^2$.*

2) Complexity of Route 2:

Theorem 10. *The complexity of Route 2 is*

$$\underbrace{[N(\frac{9}{2}\log p + 2) + 6.5p^2]}_{\textit{i), ii): fixed part}} + \underbrace{[n(8pk + 11k + p^2) + \frac{4}{3}p^3 + \mathcal{O}(p^3)]}_{\textit{iii), iv): changing part}}. \tag{32}$$

The order of the fixed part's complexity is smaller than *Route 1*, whereas the flops needed for the changing part are higher than *Route 1*.

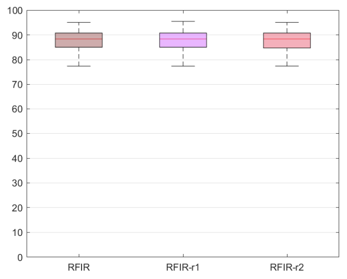
IV. Numerical Simulation

In this section, we run numerical simulations to test the efficacy of the proposed two routes. In particular, we compare the performances of route 1 (denoted by `RFIR-r1`) and route 2 (denoted by `RFIR-r2`) with Algorithm 2 in [10] (denoted by `RFIR`). We test the KRM with the Tuned-Correlated (TC) kernel which is extended 1-semiseparable [11]. In particular, we estimate the hyper-parameters and the

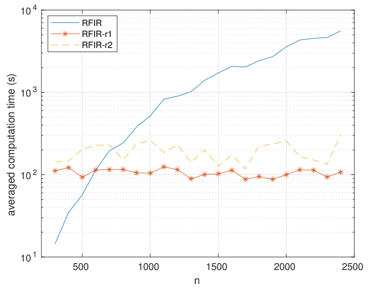
Algorithm 3 (Modification of Qiao's method) Toeplitz QR decomposition of a periodic matrix $T \in \mathbb{R}^{N\times p}$ for R_b

```
Input: Periodic part $T_b = T(1:p,1:p)$, size $N$.
Output: $R_b \in \mathbb{R}^{p\times p}$ such that $T = QR_b$;
 1: Initialize $R_{-1}, \bar{R}, \mathbf{a} \in \mathbb{R}^{p-1}, W_i, V_i, U_i, i = 1,\dots,p-1$;
 2: $m \leftarrow \left\lfloor \frac{N}{p} \right\rfloor$; $r \leftarrow N\%p$;
 3: $R_b(1,1) \leftarrow \|T(1:N,1)\|_2$ using periodic part $T_b$;
 4: $\eta_0 \leftarrow R_b(1,1)$;
 5: $\mathbf{x}_b \leftarrow \begin{bmatrix} T_b(2:p,1) \\ T_b(1,1) \end{bmatrix}$; $\mathbf{y} \leftarrow T_b(1,2:p)^T$;
 6: $\mathbf{x}^R \leftarrow \begin{bmatrix} T_b(2+r:p,1) \\ T_b(1:r,1) \end{bmatrix}$ and reverses the order of elements;
 7: $\mathbf{y}_b^R \leftarrow T_b(1,1:p)$ and reverses the order of elements;
 8: $\mathbf{b} \leftarrow T_{-1}^T\mathbf{x}$ using periodic property, $\mathbf{x}_b$ and FFT;
 9: for $k := 1$ to $p-1$ do
      /*Stage 1: Use $R(1:k,k), W_1,\dots,W_{k-1}$ to obtain $R_{-1}(1:k,k), W_k$*/
10:     $t_{N-k}^{(0)} \leftarrow \mathbf{x}^R(k)$;
11:     for $i = 1,\dots,k-1$ do
12:         $r_{i,k}^{-1} \leftarrow (r_{i,k} - sw_i \cdot t_{N-k}^{(i-1)})/cw_i$;
13:         $t_{N-k}^{(i)} \leftarrow -sw_i \cdot r_{i,k}^{(-1)} + cw_i \cdot t_{N-k}^{(i-1)}$;
14:     end for
15:     $r_{k,k}^{(-1)} \leftarrow \sqrt{r_{k,k}^2 - (t_{N-k}^{(k-1)})^2}$;
16:     $cw_k \leftarrow r_{k,k}^{(-1)}/r_{k,k},\ sw_k \leftarrow t_{N-k}^{(k-1)}/r_{k,k}$;

      /*Stage 2: Use $\mathbf{r}_k^{(-1)}, V_1,\dots,V_{k-1}, U_1,\dots,U_{k-1}$ to obtain $\mathbf{r}_{k+1}, U_k, V_k$*/
17:     $\bar{r}_{1,k} \leftarrow \mathbf{y}(k)$;
18:     for $i = 1,\dots,k-1$ do
19:         $\begin{bmatrix} \bar{r}_{i,k} \\ \bar{r}_{i+1,k} \end{bmatrix} \leftarrow \begin{bmatrix} cv_i & sv_i \\ -sv_i & cv_i \end{bmatrix} \begin{bmatrix} \bar{r}_{i,k} \\ r_{i,k}^{(-1)} \end{bmatrix}$;
20:     end for
21:     $\gamma \leftarrow \sqrt{\bar{r}_{k,k}^2 + (r_{k,k}^{-1})^2}$;
22:     $cv_k \leftarrow \bar{r}_{k,k}/\gamma$; $sv_k \leftarrow r_{k,k}^{(-1)}/\gamma$; $\bar{r}_{k,k} \leftarrow \gamma$;
23:     $\mathbf{a}(k) \leftarrow [\mathbf{b}(k) - \mathbf{r}_k^{(-1)^T}(1:k-1)\mathbf{a}(1:k-1)]/r_{k,k}^{(-1)}$;
24:     $\begin{bmatrix} \bar{a}_k \\ \hat{a}_k \end{bmatrix} \leftarrow \begin{bmatrix} cv_k & sv_k \\ -sv_k & cv_k \end{bmatrix} \begin{bmatrix} \hat{a}_{k-1} \\ a_k \end{bmatrix}$;
25:     $\eta_{k+1}^2 \leftarrow \eta_k^2 - \bar{a}_k^2$; $cu_k \leftarrow \bar{a}_k/\eta_k$; $su_k \leftarrow \eta_{k+1}/\eta_k$;
26:     for $i = k, k-1,\dots,1$ do
27:         $\begin{bmatrix} \bar{r}_{i,k} \\ r_{i+1,k+1} \end{bmatrix} \leftarrow \begin{bmatrix} cu_i & su_i \\ -su_i & cu_i \end{bmatrix} \begin{bmatrix} \bar{r}_{i,k} \\ \bar{r}_{i+1,k} \end{bmatrix}$
28:     end for
29:     $r_{1,k+1} \leftarrow \bar{r}_{1,k}$;
30: end for
```



(a) Box plot of model fits for `RFIR`, `RFIR-r1` and `RFIR-r2` in the accuracy test where the averaged model fits are 87.6618, 87.7077 and 87.5742, respectively.



(b) The averaged computation time by `RFIR` (blue line), `RFIR-r1` (orange line with stars) and `RFIR-r2` (yellow dashed line) with respect to n where the time axis is in a base-10 logarithmic scale.

noise variance σ^2 by maximizing the marginal likelihood (7), which is computed by `RFIR`, `RFIR-r1` and `RFIR-r2`. Then the corresponding regularized impulse response estimates (6) are computed.

The input signal is a periodic random Gaussian signal using the entire frequency range with period p and the output additive white Gaussian noise $v(t)$ is generated with variance one tenth of the variance of the noise-free output.

A. Accuracy test

In this test, we run Monte Carlo simulations to test the accuracy of the two routes. Specifically, we generate 80 discrete-time linear systems of 10th order with the moduli of all the poles within $[0.1, 0.9]$. The number of data points N is chosen to be 600, the FIR model order n is chosen to be 50 and the period p is chosen to be 40. To assess the estimation performance, we define the model fit:

$$\text{fit} = 100\left(1-\left[\frac{\sum_{k=1}^{n}|g_k^0-\hat{g}_k|}{\sum_{k=1}^{n}|g_k^0-\bar{g}^0|}\right]^{1/2}\right),\ \bar{g}^0=\frac{1}{n}\sum_{k=1}^{n}g_k^0 \tag{33}$$

where g_k^0 and $\hat{g}_k$ are the true and the estimated impulse response at the kth order, respectively.

The averaged model fits for `RFIR`, `RFIR-r1` and `RFIR-r2` are 87.6618, 87.7077 and 87.5742, respectively. The distribution of the model fits is shown in Fig. (a). We observe that `RFIR-r1` and `RFIR-r2` give almost the same accuracy performances as `RFIR`.

B. Efficiency test

In this test, we generate a data set from a fixed system

$$G(q)=\frac{0.2163q}{(q-0.9817)(q-0.9584)}. \tag{34}$$

The number of data points M is chosen to be 6000 and the period p is chosen to be 200 while we change the FIR model order n from 300 to 2400 with the step size 100. For each n, we identify the system for 10 times and measure the corresponding averaged computation time for the estimation tasks solved by `RFIR`, `RFIR-r1` and `RFIR-r2`. The averaged computation time with respect to the FIR model n is depicted in Fig. (b), which indicates that `RFIR-r1` and `RFIR-r2` are significantly faster than `RFIR` as n grows larger.

V. Conclusion

In this paper, we proposed an efficient implementation for kernel-based regularized system identification with semiseparable kernels and periodic input signals. The proposed implementation, as illustrated by the simulation results, is more efficient than the existing one and thus offers the users more efficient implementations of algorithms in practice.

References

[1] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, “Kernel methods in system identification, machine learning and function estimation: A survey,” *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.
[2] L. Ljung, T. Chen, and B. Mu, “A shift in paradigm for system identification,” *International Journal of Control*, vol. 93, no. 2, pp. 173–180, 2020.
[3] G. Pillonetto, T. Chen, A. Chiuso, G. De Nicolao, and L. Ljung, *Regularized System Identification: Learning Dynamic Models from Data*. Springer Nature, 2022.
[4] L. Ljung, *System identification: theory for the user*. Upper Saddle River, NJ: Prentice Hall, 1999.
[5] T. Chen, “On kernel design for regularized LTI system identification,” *Automatica*, vol. 90, pp. 109–122, 2018.
[6] A. Marconato, M. Schoukens, and J. Schoukens, “Filter-based regularisation for impulse response modelling,” *IET Control Theory & Applications*, vol. 11, pp. 194–204, 2016.
[7] M. Zorzi and A. Chiuso, “The harmonic analysis of kernel functions,” *Automatica*, vol. 94, pp. 125–137, 2018.
[8] M. Zorzi, “A second-order generalization of TC and DC kernels,” *arXiv preprint arXiv:2109.09562*, 2022.
[9] M. Bisiacco and G. Pillonetto, “On the mathematical foundations of stable RKHSs,” *Automatica*, vol. 118, p. 109038, 2020.
[10] T. Chen and L. Ljung, “Implementation of algorithms for tuning parameters in regularized least squares problems in system identification,” *Automatica*, vol. 49, no. 7, pp. 2213 – 2220, 2013.
[11] T. Chen and M. S. Andersen, “On semiseparable kernels and efficient implementation for regularized system identification and function estimation,” *Automatica*, vol. 132, p. 109682, 2021.
[12] B. Mu, T. Chen, and L. Ljung, “On asymptotic properties of hyperparameter estimators for kernel-based regularization methods,” *Automatica*, vol. 94, pp. 381–395, 2018.
[13] ——, “On the asymptotic optimality of cross-validation based hyperparameter estimators for regularized least squares regression problems,” *arXiv preprint arXiv:2104.10471*, 2021.
[14] Y. Ju, B. Mu, L. Ljung, and T. Chen, “Asymptotic theory for regularized system identification part I: Empirical Bayes hyper-parameter estimator,” *IEEE Transactions on Automatic Control*, pp. 1–16, 2023.
[15] J. Zhang, Y. Ju, B. Mu, R. Zhong, and T. Chen, “An efficient implementation for spatial–temporal Gaussian process regression and its applications,” *Automatica*, vol. 147, p. 110679, 2023.

[16] X. Yu, X. Fang, B. Mu, and T. Chen, "Kernel-based regularized iterative learning control of repetitive linear time-varying systems," *arXiv preprint arXiv:2303.03822*, 2023.

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

[18] F. P. Carli, A. Chiuso, and G. Pillonetto, "Efficient algorithms for large scale linear system identification using stable spline estimators," *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 119–124, 2012, 16th IFAC Symposium on System Identification.

[19] R. Vandebril, M. V. Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices: Linear Systems*. Johns Hopkins University Press, Baltimore, 2008.

[20] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Fast algorithms for hierarchically semiseparable matrices," *Numerical Linear Algebra with Applications*, vol. 17, no. 6, pp. 953–976, 2010.

[21] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A. J. van der Veen, and D. White, "Some fast algorithms for sequentially semiseparable representations," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 2, pp. 341–364, 2005.

[22] S. Chandrasekaran, M. Gu, and T. Pals, "A fast ULV decomposition solver for hierarchically semiseparable representations," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 603–622, 2006.

[23] S. Ambikasaran, M. O'Neil, and K. R. Singh, "Fast symmetric factorization of hierarchical matrices with applications," *arXiv preprint arXiv:1405. 0223*, 2016.

[24] C. J. Geoga, M. Anitescu, and M. L. Stein, "Scalable Gaussian process computations using hierarchical matrices," *Journal of Computational and Graphical Statistics*, vol. 29, no. 2, pp. 227–237, 2020.

[25] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil, "Fast direct methods for Gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 252–265, 2016.

[26] S. Ambikasaran and E. Darve, "An $\mathcal{O}(nlogn)$ fast direct solver for partial hierarchically semi-separable matrices," *Journal of Scientific Computing*, vol. 57, no. 3, p. 477–501, 2013.

[27] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Superfast multifrontal method for large structured linear systems of equations," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1382–1411, 2010.

[28] S. Massei, L. Robol, and D. Kressner, "hm-toolbox: Matlab software for HODLR and HSS matrices," *SIAM Journal on Scientific Computing*, vol. 42, no. 2, pp. C43–C68, 2020.

[29] G. H. Golub and C. F. van Loan, *Matrix Computations*, 4th ed. Johns Hopkins University Press, Baltimore, 2013.

[30] S. Qiao, "Hybrid algorithm for fast toeplitz orthogonalization," *Numerische Mathematik*, vol. 53, no. 3, p. 351–366, 1988.

[31] A. W. Bojanczyk, R. P. Brent, and F. R. de Hoog, "QR factorization of toeplitz matrices," *Numerische Mathematik*, vol. 49, no. 1, p. 81–94, 1986.

[32] M. S. Andersen and T. Chen, "Smoothing splines and rank structured matrices: Revisiting the spline kernel," *SIAM Journal on Matrix Analysis and Applications*, vol. 41, no. 2, pp. 389–412, 2020.

VI. Appendix

A. Proof of **Proposition 2**

In equation (23), the off-diagonal blocks $\Phi^{(l)}K{\Phi^{(l)}}^T, \Phi^{(l)}_{2^l}K{\Phi^{(l)}}^T, \Phi^{(l)}K{\Phi^{(l)}_{2^l}}^T$ and $\Phi^{(l)}_{2^l}K{\Phi^{(l)}_{2^l}}^T$ are of low-rank, since $\mathrm{rank}(\Phi^{(l)}) \le p$, $\mathrm{rank}(\Phi^{(l)}K{\Phi^{(l)}}^T) \le p$, and the other cases are similar. More generally,

$$rank(\begin{bmatrix} \Phi^{(l)}_iK\Phi^{(l)T}_j & \Phi^{(l)}_iK\Phi^{(l)T}_{j+1} & \cdots & \Phi^{(l)}_iK\Phi^{(l)T}_k \\ \Phi^{(l)}_{i+1}K\Phi^{(l)T}_j & \Phi^{(l)}_{i+1}K\Phi^{(l)T}_{j+1} & \vdots & \Phi^{(l)}_{i+1}K\Phi^{(l)T}_k \\ \vdots & \vdots & \vdots & \vdots \\ \Phi^{(l)}_hK\Phi^{(l)T}_j & \Phi^{(l)}_hK\Phi^{(l)T}_{j+1} & \cdots & \Phi^{(l)}_hK\Phi^{(l)T}_k \end{bmatrix}) \le p \tag{35}$$

where $\mathrm{i} \neq \mathrm{j}$ for all $\mathrm{i} = i, i+1, ..., h$ and $\mathrm{j} = j, j+1, ..., k$ such that all the blocks in (25) are not the diagonal blocks. Now we use the original notation as (20) for $\Phi^{(l)}_i$ for a more clear illustration.

Now, write H as

$$H^{(0)}_1 := H = \begin{bmatrix} H^{(1)}_1 & H^{(1)}_{12} \\ H^{(1)}_{21} & H^{(1)}_2 \end{bmatrix} \tag{36}$$

and

$$H^{(k)}_i = \begin{bmatrix} H^{(k+1)}_{2i-1} & H^{(k+1)}_{(2i-1)(2i)} \\ H^{(k+1)}_{(2i)(2i-1)} & H^{(k+1)}_{2i} \end{bmatrix} \tag{37}$$

for $k = 1, \dots, l-1$ and $i = 1, 2, \dots, 2^{k-1}$. Follow the same notations as (17), (18), write

$$U^{(k)}_{2i-1} = \begin{bmatrix} \Phi^{(l)T}_{(2i-2)2^{l-k}+1} & \cdots & \Phi^{(l)T}_{(2i-1)2^{l-k}} \end{bmatrix}^T; \tag{38}$$

$$U^{(k)}_{2i} = \begin{bmatrix} \Phi^{(l)T}_{(2i-1)2^{l-k}+1} & \cdots & \Phi^{(l)T}_{(2i)2^{l-k}} \end{bmatrix}^T; \tag{39}$$

$$V^{(k)}_{2i-1} = U^{(k)}_{2i-1}; V^{(k)}_{2i} = U^{(k)}_{2i}; \tag{40}$$

$$K^{(k)}_{(2i-1)(2i)} = K_{(2i)(2i-1)} = K. \tag{41}$$

Then the off-diagonal blocks in (27) are expressed as

$$H^{(k+1)}_{(2i-1)(2i)} = U^{(k+1)}_{2i-1}K^{(k+1)}_{(2i-1)(2i)}{V^{(k+1)}_{2i}}^T, \tag{42}$$

$$H^{(k+1)}_{(2i)(2i-1)} = U^{(k+1)}_{2i}K^{(k+1)}_{(2i)(2i-1)}{V^{(k+1)}_{2i-1}}^T. \tag{43}$$

By (24), $\mathrm{rank}(H^{(k+1)}_{(2i-1)(2i)}) \le p, \mathrm{rank}(H^{(k+1)}_{(2i)(2i-1)}) \le p$. And the diagonal blocks are

$$H^{(k+1)}_{2i-1} = U^{(k+1)}_{2i-1}K{V^{(k+1)}_{2i-1}}^T, \tag{44}$$

$$H^{(k+1)}_{2i} = U^{(k+1)}_{2i}K{V^{(k+1)}_{2i}}^T. \tag{45}$$

By construction, H is a HODLR matrix. Furthermore, since

$$U^{(k)}_{2i-1} = \begin{bmatrix} U^{(k-1)}_{4i-3} \\ U^{(k-1)}_{4i-2} \end{bmatrix}, \; U^{(k)}_{2i} = \begin{bmatrix} U^{(k-1)}_{4i-1} \\ U^{(k-1)}_{4i} \end{bmatrix} \tag{46}$$

for all k and i, $V^{(k)}_{2i-1}, V^{(k)}_{2i}$ are the same, H has translation property and hence is a HSS matrix.

B. Proof of **Proposition 3**

The proof uses the well-known property of spanning space of columns of original matrix and orthogonal transformation [29]. More specifically, since $\Phi_N(:, 1:p)$ has full column rank p, we have the relationship

$$ran(\Phi_N(:, 1:p)) = ran(Q(:, 1:p)) \tag{47}$$

[29]. For convenience, let

$$\Phi_N(:, 1:p) = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_p \end{bmatrix}$$
$$Q(:, 1:p) = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_p \end{bmatrix}$$

where $\boldsymbol{\phi}_k(i) = \boldsymbol{\phi}_k(i+tp)$ for $1 \le i, i+tp \le N, t \in \mathbb{Z}$ and any $k = 1, 2, \dots, p$ by periodicity of Φ_N. By (9), for any $j = 1, 2, \dots, p$, write

$$\boldsymbol{q}_j = \sum_{k=1}^{p} c_{kj}\boldsymbol{\phi}_k.$$

We have

$$\boldsymbol{q}_j(i) = \sum_{i=k}^{p} c_{kj}\boldsymbol{\phi}_k(i) = \sum_{i=k}^{p} c_{kj}\boldsymbol{\phi}_k(i+tp) = \boldsymbol{q}_j(i+tp)$$

for $1 \le i, i+tp \le N, t \in \mathbb{Z}$. Finally, we obtain $Q(i,j) = \boldsymbol{q}_j(i) = \boldsymbol{q}_j(i+tp) = Q(i+tp,j)$, which finishes the proof.

C. *Proof of* ***Proposition 4***

1) Trivially, the cost of doing a Householder transformation for $\mathbf{v}$ using **Algorithm 1** is about $3p = \mathcal{O}(p)$ flops. To see the cost of **Algorithm 2**, in the k^{th} iteration, $k = 1,\dots,p$, the cost is determined by $(I_{N-k+1} - \beta^{(k)}(\mathbf{v}^{(k)})(\mathbf{v}^{(k)})^T)\Phi_N^{(k)}(k:N,k:p)$, where $\Phi_N^{(k)}(1:N,1:p)$ denotes the changed $\Phi_N(1:N,1:p)$ by Householder transformation after $(k-1)$ iterations. The costs are:
   - $2p(p-k)$ for $(\mathbf{v}^{(k)})^T\Phi_N^{(k)}(k:N,k:p)$: line 7, 8, 9, 10, which is the calculation of $S^{(k)}$.
   - $p(p-k)$ for $\beta^{(k)}\mathbf{v}^{(k)}(\cdot)$: line 11.
   - $p(p-k)$ for the subtraction $\Phi_N^{(k)}(k:N,k:p)-(\cdot)$: line 12.

   Hence, by counting the cost of $3p$ flops for Householder transformation in each iteration, the total cost is

$$\sum_{k=1}^{p} 4p(p-k) + 3p = p^2(2p+1) \tag{48}$$

2) Given a vector $\mathbf{x} \in \mathbb{R}^N$ and $Q \in \mathbb{R}^{N\times N}$ obtained in **Algorithm 2**, $Q^T\mathbf{x}$ costs about $3Np = \mathcal{O}(Np)$ flops. We can use the Householder vectors $\mathbf{v}^{(k)}$ and parameter $\beta^{(k)}, k = 1,\dots,p$ to compute $Q^T\mathbf{x}, \mathbf{x} \in \mathbb{R}^N$ by

$$Q^T\mathbf{x} = \left[\prod_{k=p,p-1,\dots,1} \mathrm{diag}(I_{k-1}, H_{\mathbf{v}_k})\right]\mathbf{x}$$

   where $H_{\mathbf{v}_k} = I_{N-k+1} - \beta^{(k)}(\mathbf{v}^{(k)})(\mathbf{v}^{(k)})^T$. Computing $\mathrm{diag}(I_{k-1}, H_{\mathbf{v}_k})\mathbf{x}_{k-1}$ for a vector $\mathbf{x}_{k-1} \in \mathbb{R}^N$ costs $3N - 2k + p + 4$ flops, the total cost is $3Np$.

D. *Proof of* ***Theorem 5***

1) Framework: First, compute full QR decomposition of $\Phi^{(l)}(1:m,1:p)$ and $\Phi_{2^l}^{(l)}(1:r,1:p)$:

$$\Phi^{(l)}(1:m,1:p) = Q^{(l)}\begin{bmatrix}\hat{R}^{(l)} \\ 0_{(m-p)\times p}\end{bmatrix}, \tag{49}$$

$$\Phi_{2^l}^{(l)}(1:r,1:p) = Q_{2^l}^{(l)}\begin{bmatrix}\hat{R}_{2^l}^{(l)} \\ 0_{(r-p)\times p}\end{bmatrix} \tag{50}$$

where $Q^{(l)} \in \mathbb{R}^{m\times m}, Q_{2^l}^{(l)} \in \mathbb{R}^{r\times r}, \hat{R}^{(l)}, \hat{R}_{2^l}^{(l)} \in \mathbb{R}^{p\times p}$. Let

$$R^{(l)} = \begin{bmatrix}\hat{R}^{(l)} & \hat{R}^{(l)} & \dots & \hat{R}^{(l)} & \hat{\hat{R}}^{(l)} \\ 0 & 0 & \dots & 0 & 0\end{bmatrix}_{m\times n}, \tag{51}$$

$$R_{2^l}^{(l)} = \begin{bmatrix}\hat{R}_{2^l}^{(l)} & \hat{R}_{2^l}^{(l)} & \dots & \hat{R}_{2^l}^{(l)} & \hat{\hat{R}}_{2^l}^{(l)} \\ 0 & 0 & \dots & 0 & 0\end{bmatrix}_{r\times n} \tag{52}$$

where

$$\hat{\hat{R}}^{(l)} = \begin{bmatrix}\hat{R}^{(l)}(1:n\%p, 1:n\%p) \\ 0\end{bmatrix}, \tag{53}$$

$$\hat{\hat{R}}_{2^l}^{(l)} = \begin{bmatrix}\hat{R}_{2^l}^{(l)}(1:n\%p, 1:n\%p) \\ 0\end{bmatrix}. \tag{54}$$

Then

$$\begin{aligned}\Phi^{(l)} &= Q^{(l)}\begin{bmatrix}R^{(l)} \\ 0\end{bmatrix}, \\ \Phi_{2^l}^{(l)} &= Q_{2^l}^{(l)}\begin{bmatrix}R_{2^l}^{(l)} \\ 0\end{bmatrix}\end{aligned} \tag{55}$$

as presented above. Let

$$W^{(l)} := \mathrm{diag}(\underbrace{Q^{(l)},\dots,Q^{(l)}}_{2^l-1}, Q_{2^l}^{(l)}), \tag{56}$$

then

$$\begin{aligned}H^{(l)} &:= H \\ &= W^{(l)}\left(\begin{bmatrix}\begin{bmatrix}R^{(l)}KR^{(l)T} & 0 \\ 0 & 0\end{bmatrix} & \dots & \begin{bmatrix}R^{(l)}KR_{2^l}^{(l)T} & 0 \\ 0 & 0\end{bmatrix} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix}R_{2^l}^{(l)}KR^{(l)T} & 0 \\ 0 & 0\end{bmatrix} & \dots & \begin{bmatrix}R_{2^l}^{(l)}KR_{2^l}^{(l)T} & 0 \\ 0 & 0\end{bmatrix}\end{bmatrix}\right. \\ &\quad \left. + \sigma^2\mathrm{diag}(I_p, I_{m-p},\dots,I_p,I_{r-p})\right){W^{(l)}}^T \\ &= W^{(l)}P^{(l)}\begin{bmatrix}H^{(l-1)} & 0 \\ 0 & \sigma^2 I_{N-2^l p}\end{bmatrix}{P^{(l)}}^T{W^{(l)}}^T\end{aligned} \tag{57}$$

where $P^{(l)}$ is a permutation matrix that combines the dense blocks into

$$H^{(l-1)} := \begin{bmatrix}R^{(l)}KR^{(l)T} + \sigma^2 I_p & \cdots & R^{(l)}KR_{2^l}^{(l)T} \\ \vdots & \ddots & \vdots \\ R_{2^l}^{(l)}KR^{(l)T} & \cdots & R_{2^l}^{(l)}KR_{2^l}^{(l)T} + \sigma^2 I_p\end{bmatrix} \tag{58}$$

which is the next level and has $2^l p$ dimension and also has HSS structure.

Then, we factorize $H^{(l-1)}$ similarly but merge two adjacent $R^{(l)}$ or $R_{2^l}^{(l)}$ to be

$$R_{com}^{(l-1)} := R_{com,i}^{(l-1)} = \begin{bmatrix}R^{(l)T} & R^{(l)T}\end{bmatrix}^T, \; i = 1,\dots,2^{l-1}-1 \tag{59}$$

$$R_{com,2^{l-1}}^{(l-1)} = \begin{bmatrix}R^{(l)T} & {R_{2^l}^{(l)}}^T\end{bmatrix} \tag{60}$$

that have the same size $2p\times n$. Do QR decomposition similar to (55)

$$R_{com}^{(l-1)} = Q^{(l-1)}\begin{bmatrix}R^{(l-1)} \\ 0\end{bmatrix}, \tag{61}$$

$$R_{com,2^{l-1}}^{(l-1)} = Q_{2^{l-1}}^{(l-1)}\begin{bmatrix}R_{2^{l-1}}^{(l-1)} \\ 0\end{bmatrix}. \tag{62}$$

We construct $W^{(l-1)}, P^{(l-1)} \in \mathbb{R}^{2^l p\times 2^l p}$ and then

$$H^{(l-1)} = W^{(l-1)}P^{(l-1)}\begin{bmatrix}H^{(l-2)} & \\ & \sigma^2 I_{2^{l-1}p}\end{bmatrix}{P^{(l-1)}}^T{W^{(l-1)}}^T. \tag{63}$$

We continuously factorize $H^{(l-1)}$ to $H^{(l-2)}$, …, $H^{(1)}$ to $H^{(0)}$, where the relationship can be expressed as (15), $H^{(k)} \in \mathbb{R}^{2^{k+1}p\times 2^{k+1}p}$ for $k = l-1, l-2, \ldots, 2, 1$. In the last step ($l^{th}$ step) $H^{(1)}$ is denoted as (16). Note that for $k = l$, the size is slightly different, see (57).

In order to compute the first term $Y_N^T H^{-1} Y_N = Y_N^T {H^{(l)}}^{-1} Y_N$ in (7), we firstly observe that

$$ {H^{(l)}}^{-1} = W^{(l)} P^{(l)} \mathrm{diag}({H^{(l-1)}}^{-1}, \sigma^{-2} I_{N-2^l p}) {P^{(l)}}^T {W^{(l)}}^T. \tag{64}$$

Since it is symmetric, we only need to consider one side. After applying ${W^{(l)}}^T$ to Y_N, denoted as $\bar{Y}_N$, divide it into

$$\begin{aligned} \bar{Y}_N &= \begin{bmatrix} \hat{Y}_{N1}^T & \tilde{Y}_{N1}^T & \ldots & \hat{Y}_{N2^l}^T & \tilde{Y}_{N2^l}^T \end{bmatrix}^T, \\ \hat{Y}_{Ni} &= \bar{Y}_N((i-1)m+1 : (i-1)m+p), \ i = 1, ..., 2^l - 1 \\ \tilde{Y}_{Ni} &= \bar{Y}_N((i-1)m+p+1 : im), \ i = 1, ..., 2^l - 1 \\ \hat{Y}_{N2^l} &= \bar{Y}_N((2^l-1)m+1 : (2^l-1)m+p), \\ \tilde{Y}_{N2^l} &= \bar{Y}_N((2^l-1)m+p+1 : (2^l-1)m+r). \end{aligned} \tag{65}$$

By the effect of permutation matrix $P^{(l)}$, here, $\tilde{Y}_{Ni}$'s are only used for $\sigma^{-2}\tilde{Y}_{Ni}^T\tilde{Y}_{Ni}$, for $i = 1, \ldots, 2^l$, while

$$ \hat{Y}_N^{(l)} := \begin{pmatrix} \hat{Y}_{N1}^T & \ldots & \hat{Y}_{N2^l}^T \end{pmatrix}^T \in \mathbb{R}^{2^l p}. \tag{66}$$

is used in the next step where we calculate $\hat{Y}_N^{(l)T} {H^{(l-1)}}^{-1} \hat{Y}_N^{(l)}$. After applying ${W^{(l-1)}}^T$ to $\hat{Y}_N^{(l)}$, denoted as $\bar{\hat{Y}}_N^{(l)}$, we also divide it into 2^l subvectors as (65):

$$ \bar{\hat{Y}}_N^{(l)} = \begin{pmatrix} \bar{\hat{Y}}_{N1}^T & \bar{\hat{Y}}_{N2}^T & \ldots & \bar{\hat{Y}}_{N2^l}^T \end{pmatrix}^T \tag{67}$$

where each $\bar{\hat{Y}}_{Ni}$ is of size p. The vectors with an even index are used for computing dot product and multiplying σ^{-2}, while the vectors with an odd index are merged into

$$ \hat{Y}_N^{(l-1)} = \begin{pmatrix} \bar{\hat{Y}}_{N1}^T & \bar{\hat{Y}}_{N3}^T & \ldots & \bar{\hat{Y}}_{N(2^l-1)}^T \end{pmatrix}^T \in \mathbb{R}^{2^{l-1}p} \tag{68}$$

for the further calculation of $\hat{Y}_N^{(l-1)T} H^{(l-2)} \hat{Y}_N^{(l-1)}$. Repeat this procedure, after each iteration the size of $\hat{Y}_N^{(k)}$ will be halved until the last iteration,

$$ \hat{Y}_N^{(1)} := \begin{pmatrix} Y_0^T & {Y'_0}^T \end{pmatrix}^T \tag{69}$$

where both $Y_0, Y'_0 \in \mathbb{R}^p$. Finally, the quadratic form of inverse D is obtained by

$$ Y_0^T D^{-1} Y_0 \tag{70}$$

and the result of $Y_N^T H^{-1} Y_N$ is the summation of

- All the σ^{-2} times inner product of vectors as subvectors of even index in $\bar{Y}_N$ and $\hat{Y}_N^{(k)}, k = 1, 2, \ldots, l$.
- The quadratic form (70).

It is more strightforward to compute the log-determinant term $\log|H|$ in (7) after factorization since all $W^{(i)}, P^{(i)}, i = 1, \ldots, l$ have determinant 1. Hence it only requires

$$ \log|H| = (N-p)\log(\sigma^2) + \log|D| \tag{71}$$

where the direct calculation of $|D|$ requires $\mathcal{O}(p^3)$.

2) Complexity:

i) Obtaining $W^{(i)}, i = 1, \ldots, l$ for factorization costs

$$ (\frac{7}{3} + \frac{5}{3}l)p^3 + (4-2l)p^2 = \mathcal{O}(p^3) \tag{72}$$

because we need to do two QR decompostions in each level to obtain $W^{(l)}$. $W^{(l)}$ costs $2p^2(2p+1)$ by (48), and $W^{(i)}, i = 1, \ldots, l-1$ costs $2p^2(\frac{5}{6}p-1)$ since $R_{com}^{(i)}$ and $R_{com,2^i}^{(i)}$ consist of two upper triangular matrices with zero entries.

ii) Obtaining $\bar{Y}_N$ and $\bar{\hat{Y}}_N^{(i)}, i = 1, 2, \ldots, l$ after applying ${W^{(l)}}^T$ to Y_N and ${W^{(i-1)}}^T$ to $\hat{Y}_N^{(i)}$ by **Remark III.1** costs

$$ 3Np + 6p^2 \sum_{i=1}^{l} 2^i = 3Np + 12p^2(2^l - 1) = \mathcal{O}(Np) \tag{73}$$

iii) Dot product term costs $2(N-p)$ (neglected).

iv) (Changing part) $Y_0^T D^{-1} Y_0$ and $\log|D|$ cost $f(n, p, k)$.

By neglecting the lower-order terms, the total cost is approximately

$$ [3Np + (\frac{7}{3} + \frac{5}{3}l)p^3 + (12\cdot 2^l - 2l - 8)p^2] + f(n, p, k). \tag{74}$$

*E. Proof of **Proposition 6***

According to [32], for a vector $\mathbf{x} \in \mathbb{R}^n$,

- computation of $K\mathbf{x}$ using [32, Algorithm 4.1] takes $11nk$ flops;
- computation of $\mathbf{x}^T(K\mathbf{x})$ takes $2n$ flops;
- the overall cost for the computation of $\mathbf{x}^T K\mathbf{x}$ is $11nk + 2n$ flops.

Applying the above result, the procedure and the corresponding costs to compute D are described as follows:

- computation of KR^T using [32, Algorithm 4.1] takes about $8npk + 3nk$ flops.
  Let $\mathbf{r}_i := R^T(1 : n, i) \in \mathbb{R}^{n\times 1}, i = 1, ..., p$. When $i \le n\%p$, there are $\lceil\frac{n}{p}\rceil(i-1)$ zeros in $\mathbf{r}_i$; for $n\%p < i \le p$, there are $\lfloor\frac{n}{p}\rfloor(i-1) + n\%p$ zeros; especially when $n\%p = 0$, there are $\frac{n}{p}(i-1)$ zeros. $\lceil\frac{n}{p}\rceil$ is the smallest integer larger than $\frac{n}{p}$, while $\lfloor\frac{n}{p}\rfloor$ is the largest integer smaller than $\frac{n}{p}$. For $K\mathbf{x}$ with m zeros in $\mathbf{x}$, the cost is $11nk - 6mk$. Therefore, KR^T costs

$$\begin{aligned} &11npk - 6k(\sum_{i=1}^{n\%p}\lceil\frac{n}{p}\rceil(i-1) + \sum_{i=n\%p+1}^{p}(\lfloor\frac{n}{p}\rfloor(i-1) + n\%p) \\ &\approx 11npk - (3npk - 3kn) \\ &= 8npk + 3nk \end{aligned} \tag{75}$$

- $R(\cdot)$ costs about $n(p^2+p)$ flops. We need to calculate $\mathbf{r}_i^T(\cdot)$ for $i = 1, ..., p$. The cost is $2p(n - \lceil\frac{n}{p}\rceil(i-1)) \approx 2n(p-(i-1))$, the total cost is $\sum_{i=1}^{p} 2n(p-(i-1)) = 2n\sum_{i=1}^{p} p - i + 1 = n(p^2+p)$.
- Addition of $\sigma^2 I_p$ costs p flops.
- Hence the total cost to compute D is about

$$ n(8pk + 3k + p^2 + p) = \mathcal{O}(npk). \tag{76}$$

- For a vector $Y_0 \in \mathbb{R}^p$, the cost to compute $Y_0^T D^{-1} Y_0$ is

$$\begin{aligned} & n(8pk + 3k + p^2 + p) + \frac{4}{3}p^3 + \frac{7}{2}p^2 + \frac{13}{6}p \\ & \approx 8npk + np^2 + \frac{4}{3}p^3 \end{aligned} \tag{77}$$

including computing D, inversion of D with cost $(\frac{4}{3}p^3 + \frac{3}{2}p^2 - \frac{5}{6}p)$, quadratic form with cost $(2p^2 + 2p)$.
- The cost for $\log |D|$ is $\mathcal{O}(p^3)$.

Therefore, in (25), we have

$$f(n, p, k) = [8npk + np^2 + \frac{4}{3}p^3] + [\mathcal{O}(p^3)]. \tag{78}$$

F. *Proof of* ***Proposition 8***

It is trivial that $\mathbf{z}_n$ has period p. Let the periodic part be $\mathbf{z}_b = \mathbf{z}_n(1 : p)$, that is computed by

$$\begin{aligned} \mathbf{z}_b &= \Phi_N^T(1 : p, 1 : N) Y_N \\ &= \sum_{i=1}^{m} \Phi_b^T Y_N((i-1)p + 1 : ip) \\ &\quad + \Phi_b^T(1 : p, 1 : r) Y_N(N - r + 1 : N) \end{aligned} \tag{79}$$

where $m := \left\lfloor \frac{N}{p} \right\rfloor$, $r := N\%p$, Φ_b is defined in (10). By FFT, each term in the summation costs about $\frac{9}{2}p \log p$ flops, and hence the total cost is about $\frac{9}{2}N \log p$ flops. If Y_N is has period p, the computation cost of $\Phi_N^T \mathbf{x}$ is still about $9p \log p$ because the terms in the second line of (79) are the same.

G. *Notes on* ***Proposition 9*** *and* ***Algorithm 3***

We firstly present the notations Qiao used:

- $t_i := T(1, i+1), i = 1, \dots, N-1$; $t_{-j} := T(1, j+1), j = 1, \dots, p-1$;
- $T_{-1} = T(1 : N-1, 1 : p-1), \mathbf{x} = T(2 : N, 1)$;
- $T = P \begin{bmatrix} R_b^T & 0^T \end{bmatrix}^T$ is the QR decomposition of T, and $r_{i,j} := R_b(i, j), \mathbf{r}_k := R_b(1 : k, k)$.
- $T_{-1} = P_{-1} \begin{bmatrix} R_{-1}^T & 0^T \end{bmatrix}^T$ is QR decomposition of T_{-1}, and $r_{i,j}^{(-1)} := R_{-1}(i, j), \mathbf{r}_k^{(-1)} = R_{-1}(1 : k, k)$;
- $\bar{R} \in \mathbb{R}^{(p-1)\times(p-1)}$ is an upper-triangular matrix, $\bar{r}_{i,j} = \bar{R}(i, j), \bar{\mathbf{r}}_k = \bar{R}(1 : k, k)$;
- $W_i, V_i, U_i, i = 1, \dots, p-1$ are Givens rotations. Let $cw_i.sw_i$ be the cosine and sine of W_i, similar notations $cv_i, sv_i, cu_i.su_i$ are used for V_i, U_i.

For detailed meanings of each notations, see [30]. Compare with Qiao's algorithm, the main differences appear in line 3 and 8. In line 3, the 2-norm calculation is similar to **Algorithm 1** line 3 by periodic property. In line 8, the cost is about $9p \log p$ by **Proposition 7**. Other lines as well as the costs are the same as [30].

H. *Proof of* ***Theorem 10***

First, the fixed part is

(i) Obtain $R_b = R(1 : p, 1 : p)$. Cost: $9p \log p + 6.5p^2 \approx 6.5p^2$;

(ii) Compute $\mathbf{z}_n := \Phi_N^T Y_N$. Cost: $\frac{9}{2}N \log p$ flops (**Proposition 8**);

(iii) Compute $S_1 := Y_N^T Y_N$. Cost: $2N$ flops.

Second, $S_2(\alpha) = \mathbf{z}_n^T \bar{H} \mathbf{z}_n$ changes as α changes.

(iv) $s_{21}(\alpha) := \mathbf{z}_n^T K \mathbf{z}_n$. Cost: $11nk + 2n \approx 11nk$ flops by (see *proof of* ***Proposition 6***);

(v) $\hat{\mathbf{z}}_n := RK^T \mathbf{z}_n$. $K^T \mathbf{z}_n = K \mathbf{z}_n$ has been computed in iv); $R(\cdot)$ costs $n(1 + p - \frac{2}{p}) \approx np$ flops (neglected);

(vi) $s_{22}(\alpha) := \hat{\mathbf{z}}_n^T (RKR^T + \sigma^2 I_p)^{-1} \hat{\mathbf{z}}_n$. The cost is essentially the same as (53), which is

$$\begin{aligned} & n(8pk + 3k + p^2 + p) + \frac{4}{3}p^3 + \frac{7}{2}p^2 + \frac{13}{6}p \\ & \approx n(8pk + p^2) + \frac{4}{3}p^3 \end{aligned}$$

(vii) Final result of $Y_N^T H^{-1} Y_N = \sigma^{-2}(S_1 - S_2(\alpha)), S_2(\alpha) = \sigma^{-2}(s_{21}(\alpha) - s_{22}(\alpha))$. The cost is negligible.

(viii) $|H| = \sigma^{2(N-p)} \left| \sigma^2 I_p + RKR^T \right|$ by Sylvester's determinant theorem. Cost: $\mathcal{O}(p^3)$.

The total cost is

$$\begin{aligned} & \underbrace{[N(\frac{9}{2}\log p + 2) + 6.5p^2]}_{\text{(i), (ii), (iii): fixed part}} \\ & + \underbrace{[n(8pk + 11k + p^2) + \frac{4}{3}p^3 + \mathcal{O}(p^3)]}_{\text{(iv), (vi), (viii): changing part}}. \end{aligned} \tag{80}$$