Towards a Unified Quadrature Framework for Large-Scale Kernel Machines

Fanghui Liu *Member, IEEE*, Xiaolin Huang *Senior Member, IEEE*,
Yudong Chen, Johan A.K. Suykens *Fellow, IEEE*

Abstract—In this paper, we develop a quadrature framework for large-scale kernel machines via a numerical integration representation. Considering that the integration domain and measure of typical kernels, e.g., Gaussian kernels, arc-cosine kernels, are *fully symmetric*, we leverage a numerical integration technique, deterministic fully symmetric interpolatory rules, to efficiently compute quadrature *nodes* and associated weights for kernel approximation. Thanks to the *full symmetric* property, the applied interpolatory rules are able to reduce the number of needed nodes while retaining a high approximation accuracy. Further, we randomize the above deterministic rules by the classical Monte-Carlo sampling and *control variates* techniques with two merits: 1) The proposed stochastic rules make the dimension of the feature mapping flexibly varying, such that we can control the discrepancy between the original and approximate kernels by tuning the dimnension. 2) Our stochastic rules have nice statistical properties of unbiasedness and variance reduction. In addition, we elucidate the relationship between our deterministic/stochastic interpolatory rules and current typical quadrature based rules for kernel approximation, thereby unifying these methods under our framework. Experimental results on several benchmark datasets show that our methods compare favorably with other representative kernel approximation based methods.

Index Terms—random features, quadrature methods, fully symmetric interpolatory rule, kernel approximation

1 Introduction

Kernel methods [1], [2], [3] have shown to be powerful in statistical machine learning, but often scale poorly to large datasets in terms of space and time complexity [4], [5]. To make kernel methods scalable, the class of random Fourier features (RFF) [6] is one of the most effective kernel approximation techniques. RFF samples random features from a specific distribution, corresponding to the original kernel function, transforms input features to a new space via random features, and then conducts linear learning in this space. It spawns the new direction on kernel approximation for scaling up traditional kernel methods [7], [8], recent neural tangent kernel [9], [10], [11], and attention in Transformers [12], [13]. Partly due to its remarkable repercussions, Rahimi and Recht [6] won the test-of-time award for their seminal work on RFF at NeurIPS 2017.

Formally, given a positive definite kernel $k(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\mapsto\mathbb{R}$, we focus on kernel approximation in which the kernel k admits the following d-dimensional integral representation I_d

$$k(\boldsymbol{x},\boldsymbol{y}):=I_d\left(f_{\boldsymbol{xy}}\right)=\int_{\mathbb{R}^d} f_{\boldsymbol{xy}}(\boldsymbol{\omega})\mu(\mathrm{d}\boldsymbol{\omega})=\mathbb{E}_{\boldsymbol{\omega}\sim\mu}[f_{\boldsymbol{xy}}(\boldsymbol{\omega})]\,, \tag{1}$$

with the integral (probability) measure μ being standard multivariate Gaussian, i.e., $\boldsymbol{\omega}=[\omega_1,\cdots,\omega_d]^\top\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_d)$. The integrand $f_{\boldsymbol{xy}}$, f for short, is defined as $f(\boldsymbol{\omega}):=\langle\phi\left(\boldsymbol{\omega}^\top\boldsymbol{x}\right),\phi\left(\boldsymbol{\omega}^\top\boldsymbol{y}\right)\rangle$ with a nonlinear activation function ϕ. As demonstrated by [14], [15], various kernels admit this d-dimensional integration representation by choosing different ϕ. For example, the popular Gaussian kernel corresponds to $\phi(x)=[\cos(x),\sin(x)]^\top$; the zero-order arc-cosine kernel [16] admits this representation by choosing $\phi(x)$ as the Heaviside function; and the first-order arc-cosine kernel [16] corresponds to $\phi(x)=\max\{0,x\}$, i.e., the ReLU activation function commonly-used in deep neural networks.

To approximate the kernel function in Eq. (1), RFF[1] uses Monte-Carlo sampling to draw random features $\{\boldsymbol{\omega}_i\}_{i=1}^N$ from $\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_d)$ such that $k(\boldsymbol{x},\boldsymbol{y})\approx 1/N\sum_{i=1}^N f_{\boldsymbol{xy}}(\boldsymbol{\omega}_i)$. Apart from such random sampling based scheme, an alternative way is to use quadrature rules in a deterministic fashion

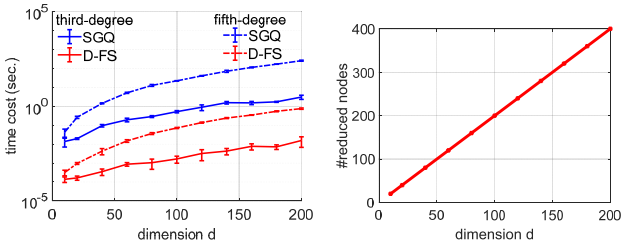
$$k(\boldsymbol{x},\boldsymbol{y})\approx\sum_{i=1}^N a_i f_{\boldsymbol{xy}}(\boldsymbol{\gamma}_i)=\langle\Phi(\boldsymbol{x}),\Phi(\boldsymbol{y})\rangle\,, \tag{2}$$

where $\boldsymbol{\gamma}_i\in\mathbb{R}^d$ is called the quadrature *node*, $a_i\in\mathbb{R}$ is the corresponding weight, and $\Phi:\mathbb{R}^d\to\mathbb{R}^N$ is the related explicit feature mapping. The nodes and weights are *deterministically* given by various quadrature rules such that there is no approximation error whenever the integrand f belongs to all polynomials with a total degree up to $2L-1$, where L is the accuracy level. For example, in the univariate case ($d=1$), Gaussian quadrature (GQ) uses L nodes to deliver the exact value of polynomials up to $(2L-1)$-degree without approximation error for $\omega_1^{i_1}\omega_2^{i_2}\cdots\omega_d^{i_d}$ with $\sum_{j=1}^d i_j\le 2L-1$. If the integrand f is general, beyond a $(2L-1)$-degree polynomial, Gaussian quadrature still works well. To be

F. Liu and J.A.K. Suykens are with the Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, B-3001 Leuven, Belgium (email: {fanghui.liu;johan.suykens}@esat.kuleuven.be).
X. Huang is with Department of Automation, and also with the MOE Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, P.R. China (e-mail: xiaolinhuang@sjtu.edu.cn).
Y. Chen is with School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850 USA (e-mail: yudong.chen@cornell.edu).

Manuscript received xx Nov. 2020; revised xx May 2021; accepted 05 Oct. 2021. Date of publication xx xxxx 2021; date of current version xx xxxx 2021. (Corresponding author: Fanghui Liu and Xiaolin Huang.)
Recommended for acceptance by xxx.
Digital Object Identifier no. xxx

1. In the original paper [6], RFF builds on Bochner's theorem [17] that requires the kernel to be shift-invariant, i.e., $k(\boldsymbol{x},\boldsymbol{y})=k(\boldsymbol{x}-\boldsymbol{y})$, which excludes arc-cosine kernels used in this paper. However, RFF is still able to provide an unbiased approximation of arc-cosine kernels by Monte-Carlo sampling according to the integral representation (1).



(a) Time cost (b) $N_{\text{SGQ}} - N_{\text{D-FS}}$

Figure 1. Benefits of D-FS against SGQ in time cost (a), and the reduction on the required *nodes* in fifth-degree rules (b).

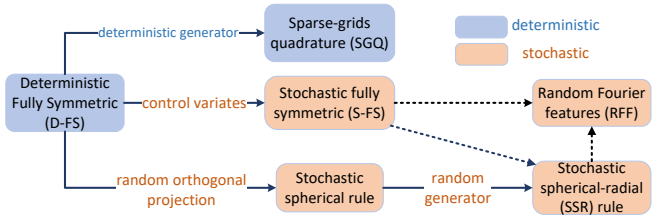


Figure 2. Relationship between quadrature based methods.

specific, if f has c-order bounded derivatives, the mean squared error (MSE) of Gaussian quadrature decreases asymptotically as $\mathcal{O}(N^{-c})$, which is better than the $N^{-1/2}$-consistency of Monte-Carlo sampling [18]. Gaussian quadrature in the univariate case can be easily extended to multidimensional cases ($d > 1$) by product rules but suffers from "curse of dimensionality": the number of required nodes is $N = L^d$ in an exponential order of d. To tackle this issue, sparse grid quadrature (SGQ) [19] uses a linear combination of low-level tensor products of univariate quadrature rules, and thus the number of nodes N by SGQ can be decreased in a polynomial order of d.

Recall Eq. (1), where the integral is not generic but has a nice property: both the integration domain $\mathbb{R}^d$ and the Gaussian measure are *fully symmetric*, with definition deferred to Section 2.2. Benefiting from this, the nodes and weights can be efficiently obtained from a *generator* vector admitting permutations and sign changes of its coordinates. Such *fully symmetric* property is helpful to reduce the number of the required nodes N in quadrature rules. For example, considering $d = 25$ with an accuracy level $L = 4$ for seventh-degree polynomial exactness approximation, Gaussian quadrature requires 4^{25} nodes; SGQ needs 24,751 nodes; while the Deterministic Fully Symmetric intepolatory rule (termed as D-FS) [20] needs 22,151 nodes, which reduces over 10% nodes. In some cases, the required nodes can be even reduced over 50% [21]. Figure 1 demonstrates the superiority of D-FS against SGQ on time cost and the reduction on required nodes[2].

Based on the above analysis, we propose to use deterministic *fully symmetric* interpolatory rules [20], i.e., D-FS, for kernel approximation. Further, we randomize such deterministic rules to new stochastic versions, termed as S-FS (here "S" denotes stochastic), which exhibit nice statistical properties: unbiased estimation and variance reduction. In addition, we elucidate the relationship among SGQ [22], stochastic spherical-radial (SSR) rules [15] and the developed D-FS/S-FS. Thereby, the proposed framework unifies these methods, as shown in Figure 2. We make the following contributions:

- By virtue of the *fully symmetric* property of the integration (1), we derive the third/fifth-degree D-FS for kernel approximation. The obtained feature mapping $\Phi(\cdot)$ is fixed-size given d, e.g., $N = 2d + 1$ in the third-degree rule and $N = 1 + 2d^2$ in the fifth-degree rule, and thus our method achieves $\mathcal{O}(d)$ time and space complexity, see Section 3.
- We randomize D-FS to a stochastic version, S-FS, by combining the classical Monte-Carlo sampling and *control variates* techniques. The proposed S-FS has two merits: 1) The dimension of the obtained feature mapping by S-FS can be easily tuned to an arbitrary value for practical requirements. 2) S-FS is theoretically demonstrated to be an unbiased estimator for kernel approximation and achieves variance reduction, see in Section 4.
- We build a unifying quadrature framework for kernel approximation as shown in Figure 2, which unifies our D-FS/S-FS, SGQ and SSR. We show that i) by choosing suitable nodes and weights in the third-degree SGQ, it is equivalent to the third-degree D-FS; ii) SSR can be regarded as a doubly stochastic version of D-FS: one stochasticity comes from random projection and another source is using a randomized *generator* vector; see in Section 5.

Besides, experimental results on several benchmark datasets show that the developed deterministic/stochastic fully symmetric interpolatory rules achieve promising kernel approximation quality and also performs well on classification tasks.

2 Related Works and Preliminaries

In this section, we give an overview of representative random features based algorithms for kernel approximation, refer to a recent survey [23] for details. Then we briefly introduce the related *fully symmetric* concepts and basic ideas behind deterministic fully symmetric interpolatory rules in numerical integration.

2.1 Related Works

To approximate the kernel function in Eq. (1), current kernel approximation methods for finding the weights and nodes $\{a_i, \boldsymbol{\gamma}_i\}_{i=1}^N$, given by Eq. (2), can be divided into *Monte Carlo* and *quadrature* based approaches.

Monte Carlo based methods are often equal-weight rules where the nodes $\{\boldsymbol{\gamma}_i\}_{i=1}^N$ are obtained by variants of Monte Carlo sampling, and then provide an unbiased estimator of the original kernel. For example, to approximate the kernel in Eq. (1), the standard RFF adopts $\boldsymbol{\gamma}_i \equiv \boldsymbol{\omega}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ by Monte Carlo sampling and the equal weights $a_1 = \cdots = a_N \equiv 1/N$. To reduce the approximation variance, orthogonal random features (ORF) [24] incorporates an orthogonality constraint on the transformation matrix $\boldsymbol{W} = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_N]$, demonstrated by theoretical guarantees on variance reduction [25]. Sampling theory [26] suggests that the convergence rate of Monte-Carlo used in RFF/ORF can be significantly improved by sampling in a deterministic scheme instead of i.i.d. version. Accordingly, quasi-Monte Carlo (QMC) sampling [27], as a possible middle-ground method, utilizes a low-discrepancy sequence for sampling, and achieves a convergence rate of order $\mathcal{O}((\log N)^c/N)$ on discrepancy [28], where c is a constant independent of N, but may

2. D-FS requires the same number of nodes $N = 2d + 1$ with SGQ in the third-degree rule but needs smaller N than SGQ in higher-degree rules.

depend on d. The convergence rate can be further improved if the integrand has higher-order smoothness [29], [30]. In fact, a series of empirical and theoretical results [14], [31] have demonstrated that, coupling samples to be orthogonal to one another (i.e., uniformly distributed over the space), rather than being i.i.d., can significantly improve statistical efficiency. Apart from the above data-independent sampling schemes used in random features, another line is to utilize data-dependent sampling strategy for better approximation quality and generalization properties for random features. Typical examples include leverage score based sampling [32], fast leverage score approximation [33], [34], Christoffel functions [35], and Fourier sparse leverage scores [36].

In quadrature based methods, the nodes are usually given by *deterministic* rules (can be extended to stochastic versions) and the weights are often not equal. Examples include Gaussian quadrature [37] and SGQ [22] based on the Smolyak formula [19]. Instead of directly approximating the d-dimensional integration, the stochastic spherical-radial (SSR) rules [38] transform the integration in Eq. (1) to a double-integral over the unit d-sphere and over the radius, which are then approximated by stochastic spherical rules and stochastic radial rules, respectively. The idea of SSR has been successfully applied to kernel approximation [15] and achieves promising approximation quality. If the integrand $f(\cdot)$ belongs to a RKHS, the above polynomial quadrature schemes are transformed to kernel-based quadrature [39], [40]. Under this setting, the studied problem is different from polynomial quadrature in terms of functional spaces, model formulation, and scope of application.

2.2 Preliminaries: Fully Symmetric Properties and Rules

Here we briefly introduce *fully symmetric sets* and related symmetry concepts, which is needed in this paper.

Definition 1. *(Fully symmetric set [18], [39], [41]) Given an integer-valued vector $\boldsymbol{p} = [p_1, p_2, \cdots, p_d]$ with $p_i \in \{0, 1, \ldots, m\}$, let $\Pi_{\boldsymbol{p}}$ be the set of all permutations of $\boldsymbol{p}$ and $\mathcal{V}_d$ be the set of all vectors with the form $\boldsymbol{\nu} = [\nu_1, \nu_2, \cdots, \nu_d]$ with $\nu_i = \pm 1$. Then, given a vector $\boldsymbol{\lambda_p} = [\lambda_{p_1}, \lambda_{p_2}, \cdots, \lambda_{p_d}]^\top$, the point set*

$$\{\boldsymbol{\lambda_p}\} := \bigcup_{q \in \Pi_{\boldsymbol{p}}} \bigcap_{\boldsymbol{\nu} \in \mathcal{V}_d} \left\{ (\nu_1 \lambda_{q_1}, \nu_2 \lambda_{q_2}, \ldots, \nu_d \lambda_{q_d}) \right\} \subset \mathbb{R}^d ,$$

is the fully symmetric set generated by $\boldsymbol{\lambda_p}$.

Based on the above definition, the concepts of fully symmetric domain, function, and measure follow naturally. To be specific, a point domain $\mathcal{A} \subseteq \mathbb{R}^d$ is said to be *fully symmetric* if $\boldsymbol{\lambda} \in \mathcal{A}$ implies $\boldsymbol{\lambda}' \in \mathcal{A}$, where $\boldsymbol{\lambda}'$ is obtained by permutations and sign changes on the coordinates of $\boldsymbol{\lambda}$. Naturally, $\mathbb{R}^d$ is a fully symmetric domain. A function $f : \mathbb{R}^d \to \mathbb{R}$ is *fully symmetric* if it is constant in each fully symmetric set, i.e., $f(\boldsymbol{x}) = f(\boldsymbol{x}')$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \{\boldsymbol{\lambda_p}\}$. A measure μ is *fully symmetric* if its density (with respect to the Lebesgue measure) is a fully symmetric function. The Gaussian measure used in Eq. (1) satisfies this condition. In Definition 1, $\boldsymbol{\lambda_p}$ is called a *generator* vector and its individual elements are called *generators*. Further, assuming $\lambda_0 = 0$, the *fully symmetric basic rule* $f(\boldsymbol{\lambda_p})$ is defined by [20]

$$f(\boldsymbol{\lambda_p}) = \sum_{q \in \Pi_{\boldsymbol{p}}} \sum_{\boldsymbol{\nu} \in \mathcal{V}_d} f\left(\nu_1 \lambda_{q_1}, \nu_2 \lambda_{q_2}, \ldots, \nu_d \lambda_{q_d}\right) .$$

For example, when $d = 4$ and $\boldsymbol{p} = (2, 0, 0, 0)$, we have

$$\begin{aligned} f(\boldsymbol{\lambda_p}) = {} & f(\lambda_2, 0, 0, 0) + f(-\lambda_2, 0, 0, 0) + f(0, \lambda_2, 0, 0) \\ & + f(0, -\lambda_2, 0, 0) + f(0, 0, \lambda_2, 0) + f(0, 0, -\lambda_2, 0) \\ & + f(0, 0, 0, \lambda_2) + f(0, 0, 0, -\lambda_2) . \end{aligned}$$

Definition 2. *(Fully symmetric interpolatory rules [20]) Define $\mathcal{P}^{(m,d)}$ as a set of all distinct d-partitions of the integers $\{0, 1, \ldots, m\}$, i.e.*

$$\mathcal{P}^{(m,d)} = \left\{ \boldsymbol{p} \in \mathbb{N}^d | p_1 \geq p_2 \geq \cdots \geq p_d \geq 0, \|\boldsymbol{p}\|_1 \leq m \right\} ,$$

the fully symmetric interpolatory rule is defined as

$$Q^{(m,d)}(f) = \sum_{\boldsymbol{p} \in \mathcal{P}^{(m,d)}} a_{\boldsymbol{p}}^{(m,d)} f(\boldsymbol{\lambda_p}) , \tag{3}$$

where the weight $a_{\boldsymbol{p}}^{(m,d)}$ is given by

$$a_{\boldsymbol{p}}^{(m,d)} = 2^{-K} \sum_{\|\boldsymbol{u}\|_1 \leq m - \|\boldsymbol{p}\|_1} \prod_{i=1}^{d} \frac{b_{u_i + p_i}}{\prod_{j=0, \neq p_i}^{u_i + p_i} \left(\lambda_{p_i}^2 - \lambda_j^2\right)} , \tag{4}$$

where $\boldsymbol{u} = [u_1, u_2, \cdots, u_d]$ is the set of the integers $\{0, 1, \ldots, m\}$ and K is the number of nonzero components in $\boldsymbol{p}$. If $\boldsymbol{q}$ is one of the permutations of $\boldsymbol{p}$, then $a_{\boldsymbol{q}}^{(m,d)} = a_{\boldsymbol{p}}^{(m,d)}$. The coefficient $b_0 = 1$ and b_i $(i \geq 1)$ satisfies

$$b_i = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-x^2/2} \prod_{j=0}^{i-1} \left(x^2 - \lambda_j^2\right) \mathrm{d}x \quad (i \geq 1) . \tag{5}$$

According to Eq. (3), $Q^{(m,d)}(f)$ stands for the weighted sum of evaluations of f at the nodes of the fully symmetric set on the distinct d-partitions $\mathcal{P}^{(m,d)}$. The theory for fully symmetric interpolatory rules [20] demonstrates that $Q^{(m,d)}(f)$ is an approximation to $I_d(f)$ that is exact for all polynomials with the total degree $2m + 1$ or less. The third/fifth-degree rules $Q^{(1,d)}(f)$ and $Q^{(2,d)}(f)$ correspond to $m = 1$ and $m = 2$, respectively. Note that, although the mathematical foundations and derivations of the fully symmetric rule are relatively complex, the obtained feature mapping for kernel approximation in this paper is quite simple and easy to be implemented. We will illustrate this in the next section.

Different from previous works: When compared to the original work on fully symmetric interpolatory rules [20], the contribution of this paper lies in developing deterministic rules for kernel approximation, especially the derivation of the fifth-degree rule, devising a new stochastic version as an unbiased estimator for kernel approximation, demonstrating nice statistical properties with theoretical guarantees, and casting typical quadrature rules in a unifying framework.

3 Deterministic Rules for Kernel Approximation

In this section, we present the third-degree and fifth-degree D-FS for kernel approximation based on the fully symmetric interpolatory rules [20]. We do not employ higher-degree rules in this paper due to sufficient approximation and efficient computation, refer to [42] with detailed discussion.

According to Eq. (3), $Q^{(m,d)}(f)$ is a weighted sum of *fully symmetric* basic rules $f(\boldsymbol{\lambda_p})$. Therefore, the kernel k, a.k.a. the d-dimensional integration (1), can be approximated by a weighted

sum of evaluations of f at the nodes of the fully symmetric set on the distinct d-partitions $\mathcal{P}^{(m,d)}$

$$k(\boldsymbol{x},\boldsymbol{y}) \approx Q^{(m,d)}(f) = \sum_{\boldsymbol{p}\in\mathcal{P}^{(m,d)}} a_{\boldsymbol{p}}^{(m,d)} f(\boldsymbol{\lambda}_{\boldsymbol{p}}), \tag{6}$$

where the weights $a_{\boldsymbol{p}}^{(m,d)}$ and the *generator* vector $\boldsymbol{\lambda}_{\boldsymbol{p}}$ play significant roles in quadrature rules. Different generation schemes for $\boldsymbol{\lambda}_{\boldsymbol{p}}$ lead to various quadrature based approaches. For example, SGQ [22] uses *deterministic* values to generate $\boldsymbol{\lambda}_{\boldsymbol{p}}$; while $\boldsymbol{\lambda}_{\boldsymbol{p}}$ in SSR [15] is sampled from a probability distribution. The developed D-FS in this section follows [20] that selects $\boldsymbol{\lambda}_{\boldsymbol{p}}$ in a *deterministic* scheme. In the next we present this generation procedure equipped with the third/fifth-degree rules for kernel approximation.

Third-degree rule: the kernel k in Eq. (1) is approximated by the third-degree rule $Q^{(1,d)}(f)$ such that $k(\boldsymbol{x},\boldsymbol{y}) \approx Q^{(1,d)}(f)$

$$Q^{(1,d)}(f) = a_0^{(1,d)} f(\boldsymbol{0}) + a_1^{(1,d)} \sum_{i=1}^{d} [f(\lambda_1 \boldsymbol{e}_i) + f(-\lambda_1 \boldsymbol{e}_i)], \tag{7}$$

where $\boldsymbol{e}_i$ is a unit vector with the i-th element being 1. The weights are given by $a_0^{(1,d)} = 1 - d/\lambda_1^2$ and $a_1^{(1,d)} = 1/(2\lambda_1^2)$ according to Eq. (4). Finally, the third-degree rule outputs $\{a_i, \boldsymbol{\gamma}_i\}_{i=0}^{2d}$ with

$$\begin{cases} \boldsymbol{\gamma}_i = \boldsymbol{0}_{d\times 1};\ a_i = 1 - d/\lambda_1^2;\ i = 0 \\ \boldsymbol{\gamma}_i = \lambda_1 \boldsymbol{e}_i;\ a_i = 1/2\lambda_1^2;\ 1 \le i \le d \\ \boldsymbol{\gamma}_i = -\lambda_1 \boldsymbol{e}_i;\ a_i = 1/2\lambda_1^2;\ d+1 \le i \le 2d, \end{cases} \tag{8}$$

which results in the number of nodes $N = 2d + 1$. The *generator* vector $\boldsymbol{\lambda} = [\lambda_0, \lambda_1]^\top$ with $\lambda_0 = 0$ usually selects λ_1 by successive extensions of the one-dimensional 3-point Gauss–Hermite rule so that certain sets of weights vanish, i.e., $\lambda_1 = \sqrt{3}$.

Feature mapping: According to the third-degree rule $Q^{(1,d)}(f)$, we finally obtain the explicit feature mapping Φ for kernel approximation

$$\Phi(\boldsymbol{x}) = [\sqrt{a_0}\phi(\boldsymbol{\gamma}_0^\top \boldsymbol{x}), \sqrt{a_1}\phi(\boldsymbol{\gamma}_1^\top \boldsymbol{x}), \cdots, \sqrt{a_{2d}}\phi(\boldsymbol{\gamma}_{2d}^\top \boldsymbol{x})]^\top, \tag{9}$$

such that $k(\boldsymbol{x},\boldsymbol{y}) \approx Q^{(1,d)}(f) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle$. Here the weight $a_0 = 1 - d/\lambda_1^2$ might be negative, and we consider the complex number $\sqrt{a_0}$, and thus the approximated kernel is still real-valued. It can be observed that, generating $\{a_i, \boldsymbol{\gamma}_i\}_{i=0}^{2d}$ is data-independent and deterministic. The transformation matrix $\boldsymbol{W} = [\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_{2d}] \in \mathbb{R}^{d\times(2d+1)}$ can be obtained by Eq. (8) as the following

$$\boldsymbol{W} = \begin{bmatrix} 0 & -\lambda_1 & \lambda_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\lambda_1 & \lambda_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & -\lambda_1 & \lambda_1 \end{bmatrix}. \tag{10}$$

For better illustration of our method, here we take the Gaussian kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\|\boldsymbol{x}-\boldsymbol{y}\|_2^2/(2\sigma^2)\right)$ as an example and discuss the difference with RFF. According to Eq. (1), the Gaussian kernel can be approximated by $k(\boldsymbol{x},\boldsymbol{y}) \approx \sum_{i=1}^{N} a_i \cos[\boldsymbol{\omega}_i^\top(\boldsymbol{x}-\boldsymbol{y})]$ with the transformation matrix $\boldsymbol{W} = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_N] \in \mathbb{R}^{d\times N}$ with N features as follows.

In RFF, $a_i \equiv 1/N$ and $W_{ij} \sim \mathcal{N}(0, 1/\sigma^2)$ by Monte Carlo sampling. The number of random features N can be manually specified to an arbitrary value; while in D-FS, the transformation matrix $\boldsymbol{W} \in \mathbb{R}^{d\times(2d+1)}$ and the weights $\{a_i\}_{i=0}^{2d}$ are *deterministic*. Given d, the number of needed nodes is $N = 2d + 1$ and cannot be easily tuned. Nevertheless, one oblivious advantage of D-FS is that, $\boldsymbol{W}$ is extremely sparse with only $2d$ non-zero elements $\pm\lambda_1$. Accordingly, generating $\boldsymbol{W}$ needs $\mathcal{O}(d)$ space and time complexity, which is better than RFF with $\mathcal{O}(Nd)$ complexity. More importantly, when d is given, the nodes, the weights, and the transformation matrix in our deterministic rules can be directly determined, see Eqs. (8) and (10). That means, our deterministic rules can be much more efficient for kernel approximation by a look-up table.

$$\begin{cases} \text{RFF:} \begin{cases} \text{dense: } \boldsymbol{W} = [W_{ij}]_{d\times N} \text{ with } W_{ij} \sim \mathcal{N}(0, 1/\sigma^2) \\ a_i \equiv 1/N \end{cases} \\ \text{D-FS:} \begin{cases} \text{sparse: } \boldsymbol{W} = [\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_{2d}] \text{ in Eq. (10)} \\ \text{the weight } a_i \text{ is given in Eq. (8)} \end{cases} \end{cases}$$

Fifth-degree rule: When choosing $m = 2$ in Eq. (6), we obtain a fifth-degree rule $Q^{(2,d)}$ with $\|\boldsymbol{p}\|_1 \le 2$ to further improve the approximation quality. To derive the fifth-degree D-FS, we cast it to three cases, i.e., $\|\boldsymbol{p}\|_1 = 0$, $\|\boldsymbol{p}\|_1 = 1$, and $\|\boldsymbol{p}\|_1 = 2$. Note that, the derivation of the fifth-degree rule is relatively technical and lengthy, so we put it in Appendix A. In fifth-degree rules, the number of required nodes in D-FS is $N = 1 + 2d^2$, which is smaller than SGQ with $1 + 2d^2 + 2d$. Further, the feature mapping in our fifth-degree rule can be obtained in a similar way with that of the third-degree rule, and thus we omit it here.

4 Stochastic Rules and Its Properties

The above D-FS rules are *determinstic*: given d, the number of required nodes N is fixed, e.g., $N = 2d+1$ in the third-degree rule and $N = 1 + 2d^2$ in the fifth-degree rule. Unlike RFF, we cannot flexibly tune N to control the discrepancy between the original and approximate kernels. This is a common issue in deterministic rules, e.g., SGQ [22]. To tackle this issue for kernel approximation, we randomize the above deterministic rules by combining the classical Monte-Carlo sampling and control variates techniques [43] for the design of stochastic rules S-FS. By doing so, we can flexibly tune the dimension of the obtained feature mapping with nice statistical properties.

4.1 Formulation of Stochastic Rules

To design the third-degree stochastic rule, we introduce a "semi-stochastic" version according to Eq. (10): we randomize the weights in Eq. (8) but keep the (deterministic) nodes unchanged to maintain the sparse transformation matrix. Observing that $\mathbb{E}[\sum_{i=1}^{d} \omega_i^2] = d$ with $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_d]^\top \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, we define the weights in our third-degree S-FS as functions of $\boldsymbol{\omega}$

$$\begin{cases} \tilde{a}_0^{(1,d)}(\boldsymbol{\omega}) \equiv \tilde{a}_0^{(1,d)} = 1 - \sum_{i=1}^{d} \omega_i^2/\lambda_1^2 \\ \tilde{a}_1^{(1,d)}(\boldsymbol{\omega}) \equiv \tilde{a}_1^{(1,d)} = \sum_{i=1}^{d} \omega_i^2/(2d\lambda_1^2). \end{cases} \tag{11}$$

Accordingly, by randomizing the weights, the "semi-stochastic" version of the third-degree D-FS $Q^{(1,d)}(f)$ in Eq. (7) is given by

$$M^{(1,d)}(f,\boldsymbol{\omega}) = \tilde{a}_0^{(1,d)} f(\boldsymbol{0}) + \tilde{a}_1^{(1,d)} \sum_{i=1}^{d} \Big[f(\lambda_1 \boldsymbol{e}_i) + f(-\lambda_1 \boldsymbol{e}_i) \Big]. \tag{12}$$

Note that the third-degree stochastic rule can be extended to general degrees

$$M^{(m,d)}(f,\boldsymbol{\omega}) = \sum_{\boldsymbol{p}\in P^{(m,d)}} \tilde{a}^{(m,d)}(\boldsymbol{\omega}) f(\boldsymbol{\lambda}_{\boldsymbol{p}}),$$

where the nodes $\boldsymbol{\lambda}_{\boldsymbol{p}}$ are the same as that of deterministic rules in Eq. (6), while the randomized weights $\tilde{a}_{\boldsymbol{p}}^{(m,d)}(\boldsymbol{\omega})$ are defined as

$$\tilde{a}_{\boldsymbol{p}}^{(m,d)}(\boldsymbol{\omega}) = \begin{cases} 1; \text{ if } \|\boldsymbol{p}\|_1 = 0 \text{ and } \|\boldsymbol{u}\|_1 = 0 \\ \dfrac{2^{-K}}{d} \displaystyle\sum_{\|\boldsymbol{u}\|_1 \le m-\|\boldsymbol{p}\|_1} \sum_{i=1}^{d} \frac{\prod_{j=0}^{u_i+p_i-1}\left(\omega_i^2-\lambda_j^2\right)}{\prod_{j=0,\neq p_i}^{u_i+p_i}\left(\lambda_{p_i}^2-\lambda_j^2\right)}, \end{cases}$$

where K is the number of nonzero components in $\boldsymbol{p}$. The formulation of $\tilde{a}_{\boldsymbol{p}}^{(m,d)}(\boldsymbol{\omega})$ is based on Eq. (4) to ensure the summation to 1. Besides, the continued product in Eq. (4) is substituted by the summation to provide a tighter estimate, as $\mathbb{E}[(\sum_{i=1}^{d}\omega_i^2)^2] = d^2+2d \le \mathbb{E}(\prod_{i=1}^{d}\omega_i^4) = 3^d$. Since typical quadrature based methods for kernel approximation, e.g., SGQ [22] and SSR [15], adopt the third-degree rule instead of higher-degree rules, in the next we focus on the third-degree stochastic rule.

The feature mapping associated with $M^{(1,d)}(f)$ is given by

$$\widetilde{\Phi}(\boldsymbol{x},\boldsymbol{\omega}) = [\sqrt{\tilde{a}_0(\boldsymbol{\omega})}\phi(\boldsymbol{\gamma}_1^\top\boldsymbol{x}), \cdots, \sqrt{\tilde{a}_{2d}(\boldsymbol{\omega})}\phi(\boldsymbol{\gamma}_{2d}^\top\boldsymbol{x})]^\top, \quad (13)$$

where $\{\boldsymbol{\gamma}\}_{i=0}^{2d}$ are given by Eq. (8), the randomized weights $\{\tilde{a}_i\}_{i=0}^{2d}$ refer to Eq. (11), and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$. Therefore, $M^{(1,d)}(f,\boldsymbol{\omega})$ is a *randomized* rule such that $M^{(1,d)}(f,\boldsymbol{\omega}) \approx \langle\widetilde{\Phi}(\boldsymbol{x}), \widetilde{\Phi}(\boldsymbol{y})\rangle$. Unfortunately, unlike that RFF is an unbiased estimator of the original kernel, the obtained estimator $M^{(1,d)}(f,\boldsymbol{\omega})$ is biased, i.e., $\mathbb{E}_{\boldsymbol{\omega}}[M^{(1,d)}(f,\boldsymbol{\omega})] = Q^{(1,d)}(f) \neq I_d(f)$. Besides, albeit stochastic, the designed $M^{(1,d)}(f,\boldsymbol{\omega})$ still outputs the fixed dimension of the feature mapping, i.e., $N = 2d+1$. In this case, we cannot flexibly tune it for practical requirements.

To tackle the above two issues, by virtue of Monte-Carlo sampling and control variates techniques [43], the designed S-FS is to pursue an unbiased estimator based on the formulation of $M^{(1,d)}(f,\boldsymbol{\omega})$. Besides, the dimension of the feature mapping by S-FS can be flexibly tuned. According to Eq. (1), we have the following equality

$$\begin{aligned} k(\boldsymbol{x},\boldsymbol{y}) &= \mathbb{E}_{\boldsymbol{\omega}}[M^{(1,d)}(f,\boldsymbol{\omega})] + \mathbb{E}_{\boldsymbol{\omega}}[f(\boldsymbol{\omega}) - M^{(1,d)}(f,\boldsymbol{\omega})] \\ &= Q^{(1,d)}(f) + \mathbb{E}_{\boldsymbol{\omega}}[f(\boldsymbol{\omega}) - M^{(1,d)}(f,\boldsymbol{\omega})]. \end{aligned}$$

As a result, the Monte-Carlo sampling for $f(\boldsymbol{\omega})$ in Eq. (1) is transformed to estimate the difference $f(\boldsymbol{\omega}) - M^{(1,d)}(f,\boldsymbol{\omega})$. If $M^{(1,d)}(f,\boldsymbol{\omega})$ is close to $f(\boldsymbol{\omega})$ in the sense that the difference has smaller variance than $f(\boldsymbol{\omega})$, variance reduction can be achieved.[3] Formally, by defining

$$R_1(f,\boldsymbol{\omega}) = Q^{(1,d)}(f) + f(\boldsymbol{\omega}) - M^{(1,d)}(f,\boldsymbol{\omega}), \quad (14)$$

then our third-degree S-FS is defined as $\bar{R}_1(f,\boldsymbol{\omega})$ such that

$$k(\boldsymbol{x},\boldsymbol{y}) \approx \bar{R}_1(f,\boldsymbol{\omega}) := \frac{1}{D}\sum_{i=1}^{D} R_1(f,\boldsymbol{\omega}_i), \quad (15)$$

with $\{\boldsymbol{\omega}_i\}_{i=1}^{D} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$. Then by defining

$$\varphi(\boldsymbol{x}) = 1/\sqrt{D}[\phi(\boldsymbol{\omega}_1^\top\boldsymbol{x}), \cdots, \phi(\boldsymbol{\omega}_D^\top\boldsymbol{x})]^\top \in \mathbb{R}^D,$$

the final feature mapping associated with $\bar{R}_1(f,\boldsymbol{\omega})$ is given by

$$\widehat{\Phi}(\boldsymbol{x}) = \left[\varphi(\boldsymbol{x})^\top, \left(\frac{\mathrm{i}}{D}\sum_{i=1}^{D}\widetilde{\Phi}(\boldsymbol{x},\boldsymbol{\omega}_i)\right)^\top, \Phi(\boldsymbol{x})^\top\right]^\top \in \mathbb{R}^{D+4d+2}, \quad (16)$$

where the symbol i is the imaginary unit, $\{\boldsymbol{\omega}_i\}_{i=1}^{D} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, and the mappings $\Phi(\boldsymbol{x})$ and $\widetilde{\Phi}(\boldsymbol{x},\boldsymbol{\omega}_i)$ are given by Eqs. (9) and (13), respectively. As a consequence, we have $k(\boldsymbol{x},\boldsymbol{y}) = \mathbb{E}\langle\widehat{\Phi}(\boldsymbol{x}), \widehat{\Phi}(\boldsymbol{y})\rangle$.

Remark: We make the following remarks.

1) The kernels used in this paper are real-valued. To approximate them, we have to introduce the imaginary unit in the feature mapping (16) due to the difference operation, i.e., $a-b = \langle(\sqrt{a}, \mathrm{i}\sqrt{b}), (\sqrt{a}, \mathrm{i}\sqrt{b})\rangle$ for any $a, b \ge 0$, but the approximated kernels still remain real-valued.

2) The discrepancy between the original and approximated kernels can be controlled by varying D in the feature mapping $\widehat{\Phi}(\boldsymbol{x}) \in \mathbb{R}^{D+4d+2}$. Note that, the feature mappings $\widetilde{\Phi}(\boldsymbol{x},\boldsymbol{\omega}_i)$ and $\Phi(\boldsymbol{x})$ in Eq. (16) have only $2d$ non-zero elements. The nodes are independent of the sampling process and can be pre-given by Eq. (8). In this case, S-FS still achieves the same space and time complexity $\mathcal{O}(Dd)$ with RFF.

3) Sampling $\{\boldsymbol{\omega}_i\}_{i=1}^{D} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ is not limited to the standard Monte Carlo sampling. It can be extended to other advanced approaches, e.g., QMC, SSR, as alternative ways, for pursuing further variance reduction. Our experimental results also verify this, see Section 6.3 for details.

4.2 Statistical Properties

This subsection elucidates that i) our third-degree S-FS is unbiased, see Theorem 1; ii) exhibits a variance reduction property in Theorem 2.

Theorem 1. *(Unbiased estimation) Our stochastic rule $\bar{R}_1(f)$ in Eq. (15) is an unbiased third-degree rule for $I_d(f)$ in Eq. (1).*

Proof. Refer to Appendix B.1. □

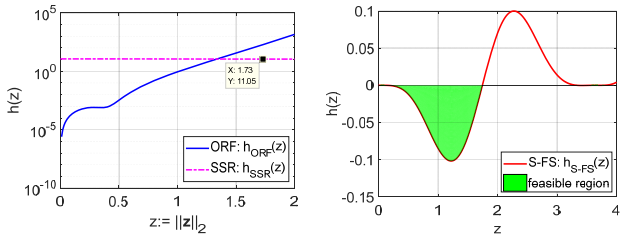
Remark: We have $Q^{(1,d)}(f) = \mathbb{E}_{\boldsymbol{\omega}\sim\mu}[M^{(1,d)}(f,\boldsymbol{\omega})]$, and thus $M^{(1,d)}(f,\boldsymbol{\omega})$ is an asymptotically unbiased estimator of $I_d(f)$.

Based on the above unbiased estimation, in the next, we derive the variance of our third-degree S-FS for Gaussian kernel approximation. Before proceeding, we introduce some notations and definitions. For the Gaussian kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\|\boldsymbol{x}-\boldsymbol{y}\|_2^2/(2\sigma^2)\right)$, we use the convenient shorthands $\boldsymbol{z} := (\boldsymbol{x}-\boldsymbol{y})/\sigma$ and $z := \|\boldsymbol{z}\|_2$. For an algorithm $\mathtt{A}$ sampling $\{\boldsymbol{\omega}_i\}_{i=1}^{D} \sim \mu$, we define its expectation $\mathbb{E}[\mathtt{A}] := \mathbb{E}_{\boldsymbol{\omega}_i\sim\mu}\left[1/D\sum_{i=1}^{D}\cos(\boldsymbol{\omega}_i^\top\boldsymbol{z})\right]$ and variance $\mathbb{V}[\mathtt{A}] := \mathbb{V}_{\boldsymbol{\omega}_i\sim\mu}\left[1/D\sum_{i=1}^{D}\cos(\boldsymbol{\omega}_i^\top\boldsymbol{z})\right]$.

Theorem 2. *(Lower variance) For the Gaussian kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\|\boldsymbol{x}-\boldsymbol{y}\|_2^2/(2\sigma^2)\right)$, denoting $z := \|\boldsymbol{z}\|_2$ with $\boldsymbol{z} := (\boldsymbol{x}-\boldsymbol{y})/\sigma$, $Q := Q^{(1,d)}(f)$ for notational simplicity, then the variance of our third-degree S-FS (15) is*

$$\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})] - \mathbb{V}[RFF] = \frac{2}{Dd}\underbrace{\left(\left[(1-Q) - \frac{1}{2}z^2e^{-\frac{z^2}{2}}\right]^2 - \frac{1}{4}z^4e^{-z^2}\right)}_{\triangleq h_{S\text{-}FS}(\boldsymbol{z})}, \quad (17)$$

3. It is possible to design other estimators close to $f(\boldsymbol{\omega})$ for variance reduction. Roughly speaking, if the estimator is closer to $f(\boldsymbol{\omega})$, then more variance reduction can be achieved.



(a) $d = 10$ (b) $d = 1$

Figure 3. Comparison of $h(z)$ versus the distance $z := \|\boldsymbol{z}\|_2$ across ORF, SSR (a) and S-FS (b). Since $h_{\text{S-FS}}(\boldsymbol{z})$ is no loner a radial function of $\boldsymbol{z}$ due to Q depending on $\boldsymbol{z}$, we just present the univariate case of $h_{\text{S-FS}}(z)$ for intuitive display.

where $\mathbb{V}[RFF] = \left(1 - e^{-z^2}\right)^2/(2D)$ *is given by [24]. In particular, the variance reduction can be achieved by*

$$\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})] - \mathbb{V}[RFF] < 0 \;\; \textit{when} \;\; 1 - Q < z^2 e^{-\frac{z^2}{2}}. \quad (18)$$

Proof. Refer to Appendix B.2. □

Remark: The condition $1 - Q < z^2 e^{-\frac{z^2}{2}}$ in Eq. (18) holds for most cases with detailed discussion in Section 4.3. Even if this condition does not hold in some rare cases, there is an alternative way to make it attainable: normalizing $z := \|\boldsymbol{x} - \boldsymbol{y}\|_2/\sigma$ to $z := \|\boldsymbol{x} - \boldsymbol{y}\|_2/\sqrt{d\sigma^2}$ by a scaling factor $\sqrt{d}$. This normalization strategy implies that the used Gaussian kernel admits $k(\boldsymbol{x},\boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|_2^2/(d\sigma^2))$, which is quite common in practice and theory. For example, in SSR [15], the authors directly employ the formulation $k(\boldsymbol{x},\boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|_2^2/d)$ in their algorithm implementation. In theory, this setting is well studied in random matrix theory and high-dimensional statistics, see [44], [45], [46], and accordingly the used normalization strategy depending d is common and fair.

Here we compare the obtained theoretical results with other representative methods on the estimated variance reduction.

Variance of ORF [24] is bounded by

$$\mathbb{V}[\text{ORF}] - \mathbb{V}[\text{RFF}] \leq \frac{1}{D} \underbrace{\left(\frac{g(z)}{d} - \frac{(d-1)e^{-z^2}z^4}{2d}\right)}_{\triangleq h_{\text{ORF}}(z)},$$

where the function g is $g(z) = e^{z^2}\left(z^8 + 6z^6 + 7z^4 + z^2\right)/4 + e^{z^2} z^4 \left(z^6 + 2z^4\right)/(2d)$, at an exponential growth of z.

Variance of SSR [15] is bounded by

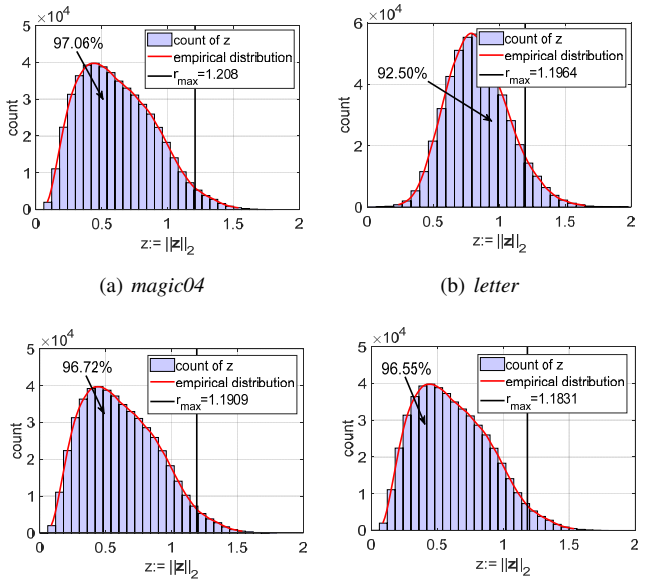
$$\mathbb{V}[\text{SSR}] - \mathbb{V}[\text{RFF}] \leq \frac{1}{D} \underbrace{\left(\frac{8d+12}{d-2} - \frac{(1-e^{-z^2})^2}{2}\right)}_{\triangleq h_{\text{SSR}}(z) > 0}, \quad (19)$$

with the positive $h_{\text{SSR}}(z)$ satisfying $\lim_{z\to\infty} h_{\text{SSR}}(z) = 8$.

For better illustration, we plot the function $h(z)$ including h_{ORF}, h_{SSR}, $h_{\text{S-FS}}$ versus the distance $z := \|\boldsymbol{z}\|_2$ in Figure 3 for intuitive explanation. In our simulation, we set $d = 10$ as an example. It can be found that, 1) h_{ORF} is positive and thus the variance reduction cannot be demonstrated in theory. More specifically, h_{ORF} almost increases at an exponential order of z, which leads to a quite loose bound for variance estimation. 2) SSR cannot strictly guarantee $\mathbb{V}[\text{SSR}] < \mathbb{V}[\text{RFF}]$ due to $h_{\text{SSR}}(z) > 0$ in Eq. (19). Instead, our theoretical result in Theorem 2 admits $\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})] < \mathbb{V}[\text{RFF}]$ under the condition in Eq. (18) for variance reduction, as demonstrated by Figure 3(b) in the univariate case. Even if this condition does not hold, we still have $\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})] - \mathbb{V}[\text{RFF}] \leq 2/(Dd)$ at a certain $\mathcal{O}\left(1/(Dd)\right)$ rate as $h_{\text{S-FS}}(z)$ is bounded. This is faster than SSR converging at a certain $\mathcal{O}(1/D)$ rate.

Table 1
The maximum radius of the hyper-ball $\mathcal{S}^d(r)$ under various d.

d	$r_{\max}$	d	$r_{\max}$
10 (*magic04*)	1.208	50	1.1837
16 (*letter*)	1.1964	54 (*covtype*)	1.1831
20	1.1896	100	1.18
22 (*ijcnn1*)	1.1909	200	1.1787



(a) *magic04* (b) *letter*

(c) *ijcnn1* (d) *covtype*

Figure 4. Empirical distribution of z in four datasets used in this paper.

4.3 Discussion on the Condition (18) in Theorem 2

Here we verify that the condition (18) for $\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})] - \mathbb{V}[\text{RFF}] < 0$ in Theorem 2 holds for most cases. The description of the used four datasets (*magic04*, *letter*, *ijcnn1*, *covtype*) for numerical validation is deferred to our experiments in Section 6.

Under the Gaussian kernel setting, recall our third-degree D-FS (7), the condition (18) is equivalent to

$$\frac{d}{3} - \frac{1}{3}\sum_{i=1}^{d} \cos(\sqrt{3}\boldsymbol{e}_i^\top \boldsymbol{z}) - \|\boldsymbol{z}\|_2^2 \exp(-\|\boldsymbol{z}\|_2^2/2) < 0\,. \quad (20)$$

For notational simplicity, we denote the left-hand side of the above inequality as $J(\boldsymbol{z})$. In the next, we first study the existence of solutions to Eq. (20) and then numerically validate that the condition under these solutions holds for most cases.

4.3.1 *Existence*

We consider a simple case: finding a d-dimensional Euclidean ball $\mathcal{S}^d(r) = \{\boldsymbol{z} \in \mathbb{R}^d : \|\boldsymbol{z}\|_2 \leq r\}$ as the feasible region, such that all points in $\mathcal{S}^d(r)$ admit $J(\boldsymbol{z}) < 0$. As a result, our

target is transformed to maximize r by solving a one-dimensional optimization problem

$$\max\ r\ ,\ \text{s.t.}\ \frac{1}{3}-\frac{1}{3}\cos\left(\frac{\sqrt{3}r}{\sqrt{d}}\right)-\frac{r^2}{d}\exp\left(-\frac{r^2}{2}\right)<0\,.$$

After numerical calculation, the maximum radius $r_{\max}$ under different d is reported in Table 1. That means, given d, there exists a hyper-ball $\mathcal{S}^d(r_{\max})$ such that any vector $\boldsymbol{z}\in\mathbb{R}^d$ with $z:=\|\boldsymbol{z}\|_2\le r_{\max}$ admits the condition (18). We remark that, the above inequality is not equivalent to Eq. (20) but a special case for existence.

4.3.2 Numerical validation

Here we numerically validate that the obtained $r_{\max}$ in Table 1 holds for most cases in four datasets used in this paper.

In our numerical simulation, on each dataset, we randomly select 1,000 data points $\{\boldsymbol{x}_i\}_{i=1}^{1000}$ to compute the distance $z_{ij}=\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2$ with $1\le i,j\le 1000$, and then construct a histogram with 30 bins for counting z_{ij}.[4] Figure 4 shows the histogram for counting z_{ij} and the fitted empirical distribution of z on these four datasets. Observe that all the datasets admit $z_{ij}<2$, which shows the consistency with [24] (see Figure 2(c) in their paper). Further, we also plot $r_{\max}$ in Table 1 on each dataset (see the black line) in Figure 4, and find that, over 90% of $\{z_{ij}\}_{i,j=1}^{1000}$ satisfy $z:=\|\boldsymbol{z}\|_2\le r_{\max}$. That means, our condition (18) holds for most cases and thus is fair and attainable.

5 Unifying Framework for Quadrature Methods

In this section, we investigate the relations among third-degree rules, including SGQ [22], SSR [15], and our deterministic/stochastic rules, i.e., D-FS/S-FS. Subsequently, we cast them in our unifying framework for kernel approximation.

5.1 Relations to SGQ

The sparse grids used in [22] are based on the Smolyak rule [19] which can be approximated by a sequence of nested univariate quadrature rules in a tensor product fashion

$$I_d(f)\approx A_{d,L}(f)=\sum_{q=0}^{L-1}\sum_{\boldsymbol{i}\in\mathcal{C}_q^d}(\Delta_{i_1}\otimes\cdots\otimes\Delta_{i_d})(f)\,,\qquad(21)$$

with the index vector $\boldsymbol{i}=[i_1,i_2,\cdots,i_d]$. The set $\mathcal{C}_q^d=\left\{\boldsymbol{i}\in\mathbb{N}^d:\sum_{j=1}^d i_j=d+q\right\}$ determines the possible accuracy level i_j for each univariate quadrature and the nonnegative q prescribes the range of the accuracy level i_j in each dimension. V_{i_j} is the univariate quadrature rule with the accuracy level $i_j\in\boldsymbol{i}$, which generates the difference $\Delta_i(f)=V_i(f)-V_{i-1}(f)$, $\forall i\in\mathbb{N}$. This rule is a weighted sum of product rules with different combinations of accuracy levels $\boldsymbol{i}$.

To study the relationship between SGQ and D-FS, we construct the third-degree SGQ in Eq. (21) using the symmetric univariate quadrature point set $\{-\hat{p}_1,0,\hat{p}_1\}$ and the weights $(\hat{a}_1,\hat{a}_0,\hat{a}_1)$, then the integration $I_d(f)$ can be approximated by SGQ

$$I_d(f)\approx(1-d+d\hat{a}_0)\,f(\boldsymbol{0})+\hat{a}_1\sum_{j=1}^d\left[f\left(\hat{p}_1\boldsymbol{e}_j\right)+f\left(-\hat{p}_1\boldsymbol{e}_j\right)\right].$$

4. The diagonal elements $z_{ii}=0$ are not counted and non-diagonal elements are counted only once.

Table 2
Relationship between typical kernel approximation methods.

Methods	Parameters in Eq. (23)
SSR	$\beta:=d$
$M^{(1,d)}(f)$ in Eq. (12)	$\rho:=\lambda_1^2$ and $\boldsymbol{Q}:=\boldsymbol{I}$
ORF	$\beta:=d$ and $\rho\sim\chi(d)$
$Q^{(1,d)}(f)$ in Eq. (7)	$\rho:=\lambda_1^2$, $\boldsymbol{Q}:=\boldsymbol{I}$, and $\beta:=d$
SGQ	$\rho:=\lambda_1^2$, $\boldsymbol{Q}:=\boldsymbol{I}$, $\beta:=d$ $\{\hat{a}_0,\hat{p}_0,\hat{a}_1\}\leftarrow\lambda_1$

If the nodes and their associated weights are chosen by the following scheme

$$\hat{a}_0:=1-\frac{1}{\lambda_1^2},\ \hat{p}_1:=\lambda_1,\ \hat{a}_1=\frac{1}{2\lambda_1^2}\,,$$

then the third-degree SGQ is equivalent to D-FS in Eq. (7), as shown in Figure 2.

5.2 Relations to SSR

The key step in SSR [38] is a change of variable from $\boldsymbol{\omega}\in\mathbb{R}^d$ to a radius r and direction vector $\boldsymbol{a}\in\mathbb{R}^d$. Let $\boldsymbol{\omega}=r\boldsymbol{a}$ with $\boldsymbol{a}^\top\boldsymbol{a}=1$ and $r\in[0,\infty)$, we have

$$\begin{aligned}I_d(f)&=\frac{(2\pi)^{-\frac{d}{2}}}{2}\int_{U_d}\int_{-\infty}^{\infty}|r|^{d-1}e^{-\frac{r^2}{2}}f(r\boldsymbol{a})\mathrm{d}\tau(\boldsymbol{a})\mathrm{d}r\\&\approx f(\boldsymbol{0})\left(1-\frac{d}{\rho^2}\right)+\sum_{j=1}^d\frac{f(-\rho\boldsymbol{Q}\boldsymbol{e}_j)+f\left(\rho\boldsymbol{Q}\boldsymbol{e}_j\right)}{2\rho^2}\,,\end{aligned}$$

where $\boldsymbol{Q}$ is a random orthogonal matrix, $\tau(\cdot)$ is the spherical surface measure or the area element on U_d, and $\rho\sim\chi(d+2)$. SSR includes the following two stochastic integration rules: one is stochastic radial rule for approximating the infinite range integral $\int_{-\infty}^{\infty}e^{-\frac{r^2}{2}}|r|^{d-1}f(r)\mathrm{d}r$; the other is stochastic spherical rule for a surface integral over U_d

$$I_{\boldsymbol{Q},U_d}(f)=\frac{|U_d|}{2d}\sum_{j=1}^d\left[f\left(\boldsymbol{Q}\boldsymbol{e}_j\right)+f\left(-\boldsymbol{Q}\boldsymbol{e}_j\right)\right],\qquad(22)$$

where $|U_d|=2\sqrt{\pi^d}/\Gamma(d/2)$ is the surface area of the unit sphere with the Gamma function Γ.

Here we present the following theorem that states the relationship between the third-degree stochastic spherical rule and the third-degree D-FS.

Theorem 3. *The third-degree stochastic spherical integration rule (22) can be obtained by the random orthogonal projection of D-FS in Eq. (7).*

Proof. Refer to Appendix C. □

Accordingly, SSR can be obtained by D-FS in Eq. (7) with the following two randomized steps. 1) random projection: according to Theorem 3, by projecting D-FS to the spherical surface of U_d with a uniform random orthogonal matrix $\boldsymbol{Q}$, we can obtain the third-degree stochastic spherical rule. 2) random generator: the deterministic *generator* λ_1 in Eq. (7) by Gaussian quadrature is substituted by a random variable ρ with $\rho\sim\chi(d+2)$. By doing so, we can transform D-FS to SSR, as shown in Figure 2.

Table 3
Dataset statistics and the number of *nodes* in fifth-degree rules.

datasets	d	#training	#test	#nodes N	
				SGQ	Ours
magic04	10	9,510	9,510	221	201
letter	16	12,000	6,000	545	513
ijcnn1	22	49,990	91,701	1013	969
covtype	54	290,506	290,506	5941	5833

5.3 Unifying Framework

Apart from the relations between our deterministic rule and SSR, here we also study the relationship between S-FS and SSR. On the one hand, in Eq. (14), if we only consider $R_1(f,\boldsymbol{\omega}) := f(\boldsymbol{\omega})$, S-FS degenerates to RFF with the standard Monte-Carlo sampling scheme. On the other hand, RFF (also ORF) can be regarded as spacial cases of SSR as demonstrated by [15]. Furthermore, if we only consider $R_1(f,\boldsymbol{\omega}) := M^{(1,d)}(f,\boldsymbol{\omega})$ in Eq. (14), after the above two stochastic operations (random projection and random *generator*), it is a triple-stochastic rule with the following formulation

$$I_d(f) \approx f(\mathbf{0})\left(1-\frac{\beta}{\rho^2}\right) + \frac{\beta}{d}\sum_{j=1}^{d}\frac{f(-\rho\boldsymbol{Q}\boldsymbol{e}_j)+f(\rho\boldsymbol{Q}\boldsymbol{e}_j)}{2\rho^2}, \tag{23}$$

with $\beta \sim \chi(d)$ and $\rho \sim \chi(d+2)$. Clearly, this rule is also an unbiased estimator of $I_d(f)$. Finally, we summarize the relations between D-FS/S-FS, SGQ, SSR, ORF under the unifying framework in Table 2.

6 Empirical Results

In this section, we empirically compare our deterministic/stochastic rules, D-FS and S-FS, with several representative approaches for kernel approximation, and then incorporate them into the kernel ridge regression (KRR) for classification on several benchmark datasets. Given nodes and weights in Eq. (8), our algorithm is straightforward to be implemented for the feature mapping in Eq. (9) by our deterministic rule and Eq. (16) by our stochastic rule. We implement them in MATLAB and carry out on a PC with Intel® i7-8700K CPU (3.70 GHz) and 64 GB RAM. The source code of our implementation can be found in http://www.lfhsgre.org.

6.1 Experimental Settings

Kernel: According to the integral representation (1), we choose the popular Gaussian kernel and the first-order arc-cosine kernel for experimental validation. Here we use the following formulation of the Gaussian kernel

$$k(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2d\sigma^2}\right), \tag{24}$$

where the feature dimension d is introduced into the kernel width for scaling as suggested by the remark in Theorem 2. The parameter σ^2 is tuned via 5-fold inner cross validation over a grid of $\{0.1, 0.5, 1, 5, 10\}$. The first-order arc-cosine kernel [16] used in this paper is given by

$$k(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{\pi}\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2\left(\sin\theta + (\pi-\theta)\cos\theta\right),$$

with $\theta = \cos^{-1}\left(\frac{\boldsymbol{x}^\top\boldsymbol{y}}{\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2}\right)$.

Datasets: We consider four typical classification datasets including *magic04*, *letter*, *ijcnn1*, and *covtype*; see Table 3 for an overview. These datasets can be downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ or the UCI Machine Learning Repository[5]. The *ijcnn1* dataset by the provider has been already scaled to $[0,1]^d$ by the winner's transformation [47]. The remaining three datasets rescale each attribute/feature to $[0,1]$ by the min-max normalization. Regarding to the training/test partition, it has been pre-given on the *letter* and *ijcnn1* datasets. For the remaining two datasets, we randomly pick half of the data for training and the rest for test.

Compared methods: We compare the developed D-FS/S-FS with the following algorithms:

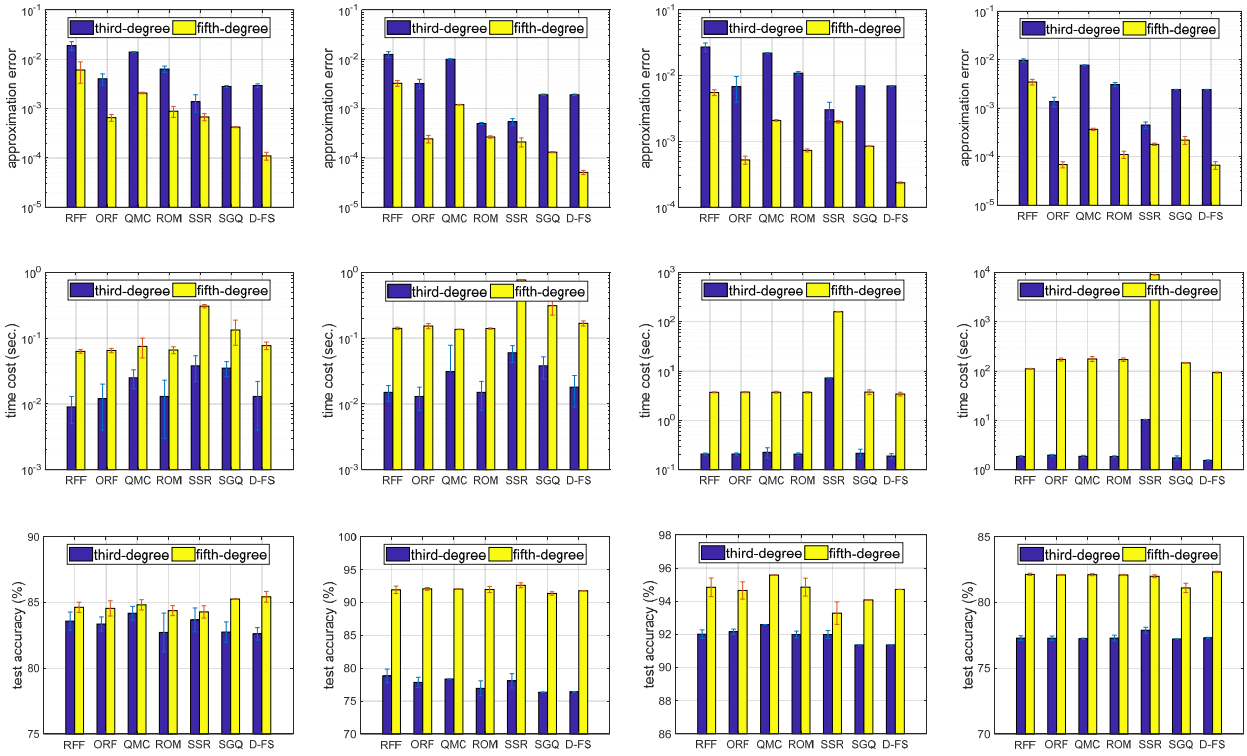
- RFF/MC [6]: The transformation matrix $\boldsymbol{W}_{\text{RFF}}$ is constructed by the standard Monte Carlo sampling scheme with $W_{ij} \sim \mathcal{N}(0, 1/(d\sigma^2))$ in Eq. (24) for Gaussian kernel approximation and $W_{ij} \sim \mathcal{N}(0,1)$ for the first-order arc-cosine kernel approximation.
- ORF [24]: The transformation matrix $\boldsymbol{W}_{\text{ORF}}$ is constructed by a random orthogonal matrix with $\boldsymbol{W} = \boldsymbol{\Lambda}\boldsymbol{Q}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix with $\Lambda_{ii} \sim \chi(d)$ and $\boldsymbol{Q}$ is obtained from the QR decomposition of $\boldsymbol{W}_{\text{RFF}}$. Note that, this approach can be applied to the arc-cosine kernel in practice but lacks theoretical guarantees.
- ROM [25]: The transformation matrix $\boldsymbol{W}_{\text{ROM}}$ is constructed by a series of structural random orthogonal matrices with $\boldsymbol{W} = c\prod_{i=1}^{t}\boldsymbol{H}\boldsymbol{\Lambda}_i$, where $\boldsymbol{H}$ is a normalized Hadamard matrix and $\boldsymbol{\Lambda}_i$ is the Rademacher matrix with $\mathbb{P}(\Lambda_{ii}=\pm 1)=1/2$. Here c is chosen as $\sqrt{2/\sigma^2}$ for Gaussian kernel approximation and $\sqrt{d}$ for arc-cosine kernel approximation.
- QMC [27]: The transformation matrix $\boldsymbol{W}_{\text{QMC}}$ is constructed by a deterministic low-discrepancy Halton sequence.
- GQ/SGQ [22]: These two algorithms are deterministic quadrature methods. GQ generates nodes and weights along each dimension and thus the dimension of the obtained feature mapping can be manually adjusted. However, the feature dimension generated by SGQ is directly fixed if d is given. Accordingly, we compare SGQ with D-FS in Section 6.2 and compare GQ with S-FS in Section 6.3. For fair comparison, we set the *generator* vector $\boldsymbol{\lambda} = [0,\sqrt{3}]^\top$ in GQ, SGQ and D-FS/S-FS to be the same.
- SSR [15]: The feature mapping is constructed by the third-degree stochastic spherical-radial rule with random orthogonal matrices obtained by butterfly matrices [48].

Evaluation metrics: We evaluate the performance of all the compared algorithms in terms of approximation error, time cost, and test accuracy. The used kernel approximation measure here is the relative error in Frobenius form $\|\boldsymbol{K}-\hat{\boldsymbol{K}}\|_{\text{F}}/\|\boldsymbol{K}\|_{\text{F}}$ on a randomly selected subset with 1,000 samples. We record the time cost of each algorithm on generating feature mappings. For prediction (binary classification), we directly use the closed-form formula of KRR [2] and the sign function to output a binary label. The regularization parameter in KRR is tuned via 5-fold inner cross validation over a grid of $\{0.0001, 0.001, 0.01, 0.1, 0.5, 1, 10\}$. All experiments are repeated 10 trials.

6.2 Evaluation for Deterministic Rules

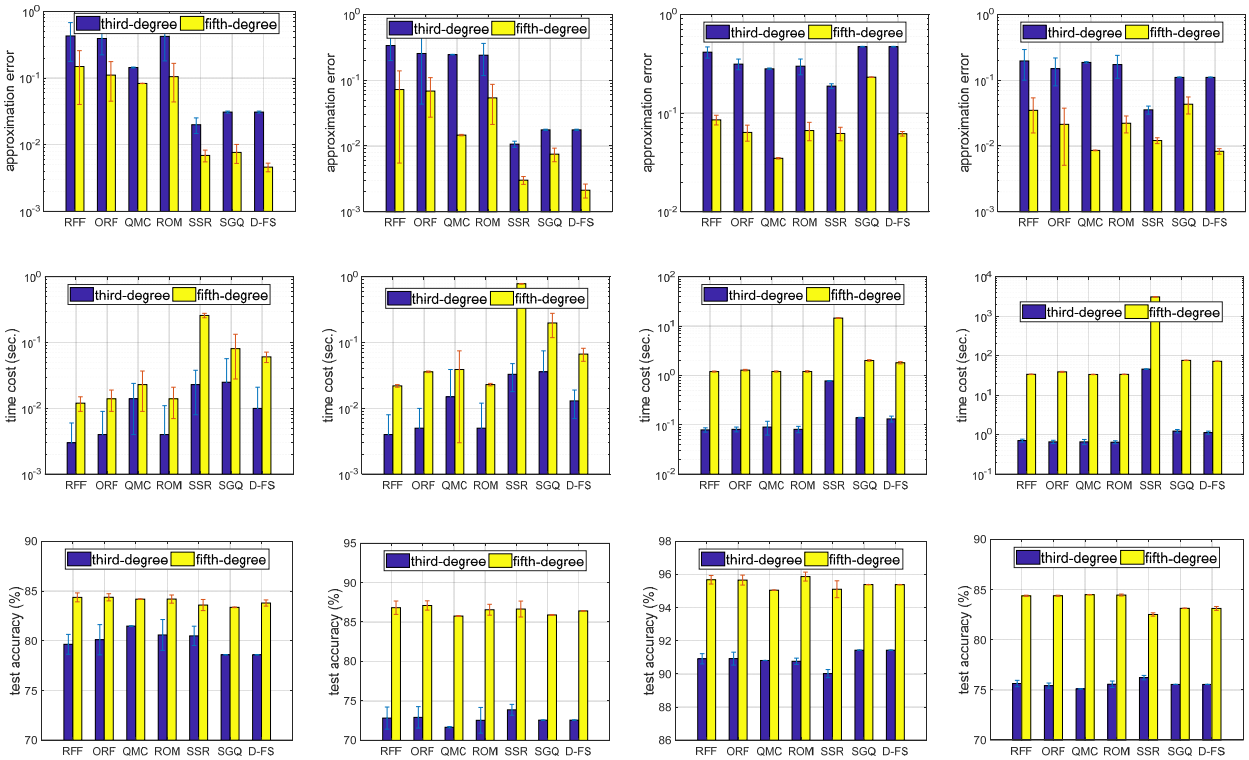
Our deterministic rules D-FS generate the fixed-size feature mapping $\Phi(\boldsymbol{x}) \in \mathbb{R}^N$ when d is given, e.g., $N = 2d+1$ in

5. https://archive.ics.uci.edu/ml/datasets.html.



(i) *magic04* (j) *letter* (k) *ijcnn1* (l) *covtype*

Figure 5. Results on the Gaussian kernel in terms of approximation error (top), time cost (middle), and test accuracy (bottom).



(i) *magic04* (j) *letter* (k) *ijcnn1* (l) *covtype*

Figure 6. Results on the first-order arc-cosine kernel in terms of approximation error (top), time cost (middle), and test accuracy (bottom).

our third-degree rule ($m = 1$) and $N = 1 + 2d^2$ in our fifth-degree rule ($m = 2$). In this case, we consider a deterministic setting, in which RFF, ORF, ROM, QMC and SSR are conducted under the same feature dimension N with our third/fifth-degree rules for fair comparison. Note that SGQ generates the same feature dimension $N = 2d + 1$ with D-FS in the third-degree rules but outputs the larger one $N = 1 + 2d^2 + 2d$ in the fifth rule, see in Table 3.

Results on Gaussian kernel: Figure 5 shows approximation error, time cost, and test accuracy (mean±std.) of all the compared algorithms across the Gaussian kernel in terms of the third-degree rules (see the blue bar) and the fifth-degree rules (see the yellow bar), respectively. We find that, our third-degree D-FS decreases the approximation error of RFF and QMC, achieves a comparable performance with ORF and ROM, but is slightly inferior to SSR. Besides, the third-degree SGQ performs the same with D-FS in terms of the approximation error as the generated nodes in these two algorithms are almost the same due to the same *generator* used. Nevertheless, our fifth-degree D-FS not only requires smaller N than SGQ, but also achieves the best approximation quality (with noticeable reduction) of all the compared algorithms.

In terms of time cost on generating the feature mapping, there is no distinct difference between our third/fifth-degree D-FS and RFF. Interestingly, our fifth-degree D-FS is more efficient than quadrature methods SSR and SGQ. For prediction, most algorithms achieve the similar test accuracy on these datasets. Good kernel approximation quality cannot guarantee the final good prediction, which still remains an open question in theory. The reason may be that the approximated kernel is not necessarily optimal for prediction, as discussed by [23], [32], [49]. Nevertheless, for the design of kernel approximation, it is reasonable to pursue small approximation errors.

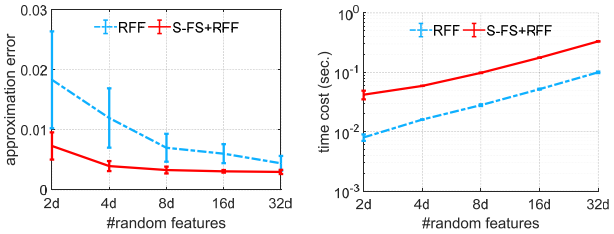
Results on arc-cosine kernel: Figure 6 shows the related results across the first-order arc-cosine kernel. The trends of the compared algorithms are analogous to those across the Gaussian kernel in Figure 5. Generally, the approximation error of each algorithm on the arc-cosine kernel is larger than that of Gaussian kernel. The reason may be that the integrand f for the Gaussian kernel corresponds to trigonometric functions that are infinitely differentiable; while f for the first-order arc-cosine kernel is actually a ReLU function that is non-differentiable. In fact, as we discussed in the introduction, the differentiable property on the integrand significantly affects the approximation performance in Monte Carlo sampling, QMC, and quadrature methods.

Based on the above results, we conclude that our deterministic third/fifth-degree rules are quite efficient to achieve promising performance on the approximation quality, and comparable results on classification accuracy.

6.3 Evaluation for Stochastic Rules

Here we evaluate the proposed third-degree S-FS under a dimension adjustment setting, in which the feature dimension in Eq. (16) is manually fixed with $D = \{2d, 4d, 8d, 16d, 32d\}$. In this case, S-FS generates the feature mapping $\widehat{\Phi}(\cdot) \in \mathbb{R}^{D+4d+2}$, but still achieves the same time/space complexity $\mathcal{O}(Dd)$ with RFF. We begin with an intuitive comparison of S-FS in Eq. (16) against RFF and then conduct a comprehensive experimental evaluation of all the randomized algorithms.

First, to validate the effectiveness of S-FS on variance reduction, Figure 7 shows the approximation error and the time cost across



(a) approximation error (b) time cost

Figure 7. Benefits of our S-FS rule in Eq. (16) against RFF across the Gaussian kernel on the *magic04* data set.

the Gaussian kernel on the *magic04* data set between S-FS and RFF. Both of them draw $\{\boldsymbol{\omega}_i\}_{i=1}^D \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ by Monte-Carlo sampling, so S-FS under this setting is termed as "S-FS+RFF". It can be noticed that, admittedly, "S-FS+RFF" takes a little more time than RFF on generating the feature mapping. However, it achieves significant improvement on RFF in terms of the approximation quality, which demonstrates the effectiveness of the used control variates technique in Eq. (16). Besides, we observe that, the variance reduction effect weakens or even disappears when D is large. One reason might be that, the variance of S-FS converges to that of RFF at a fast $\mathcal{O}(1/(Dd))$ rate, as demonstrated by Theorem 2.

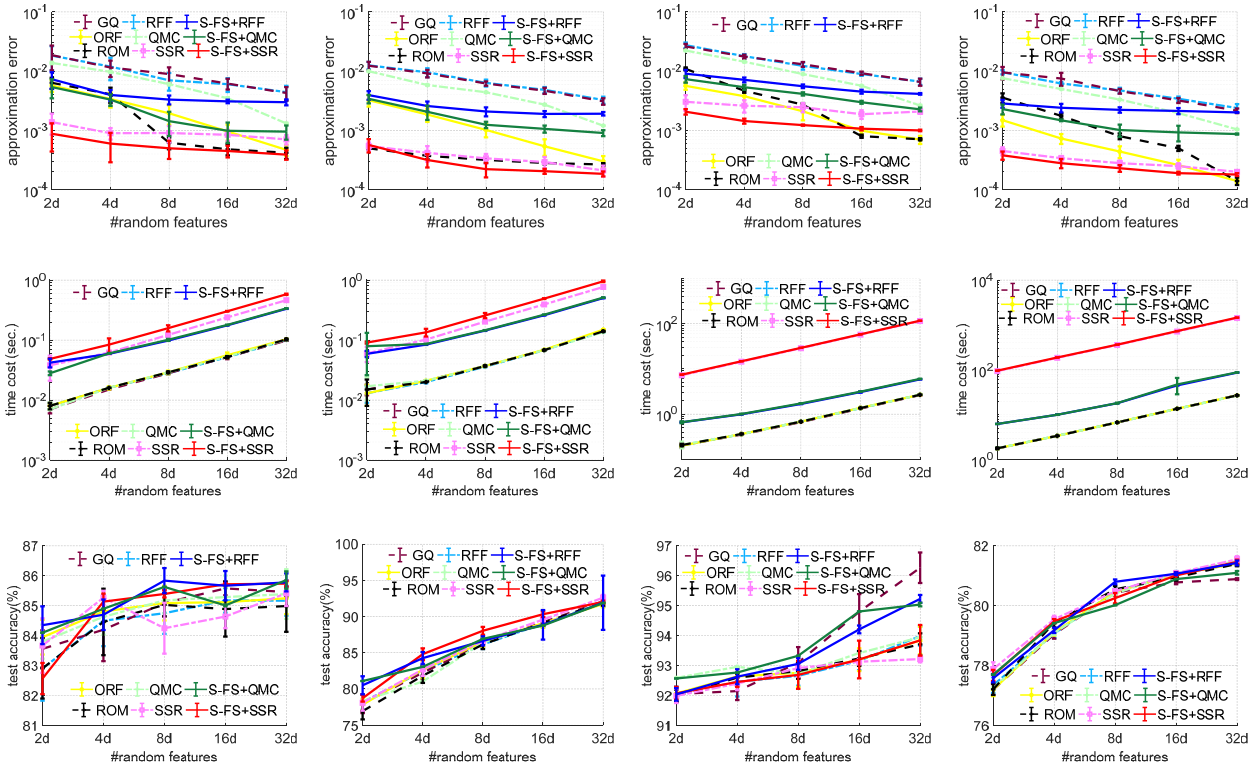
In the next, we present a comprehensive evaluation of the proposed S-FS rule with other representative approaches. To purse a better approximation performance, apart from the original Monte-Carlo sampling in S-FS, we also incorporate various sampling strategies into S-FS: $\{\boldsymbol{\omega}_i\}_{i=1}^D$ in Eq. (16) are obtained by QMC and SSR, termed as "S-FS+QMC" and "S-FS+SSR" respectively.

Results on Gaussian kernel: Figure 8 shows the approximation error, time cost, and test accuracy (mean±std.) of all the compared algorithms across the Gaussian kernel under different feature dimensionality with $D = \{2d, 4d, 8d, 16d, 32d\}$. We find that, when compared to the original RFF, QMC, SSR, our stochastic rules including "S-FS+RFF", "S-FS+QMC", "S-FS+SSR" manifest significant reduction on the approximation error, respectively. In terms of time complexity, due to the used control variates technique, our stochastic rules take more time than the original RFF/ORF/QMC/ROM/GQ. Among these three sampling strategies, "S-FS+RFF" and "S-FS+QMC" take the similar time cost on generating the feature mapping, achieving the same time complexity $\mathcal{O}(Dd)$ with RFF. However, "S-FS+SSR" is relatively time-consuming on the *ijcnn1* and *covtype* datasets as SSR itself requires more time to obtain random orthogonal matrices in large scale situations.

As mentioned before, the compared algorithms achieve the similar test accuracy in the deterministic setting. There is almost no distinct difference between these approaches on the final classification accuracy under varying feature dimensionlity.

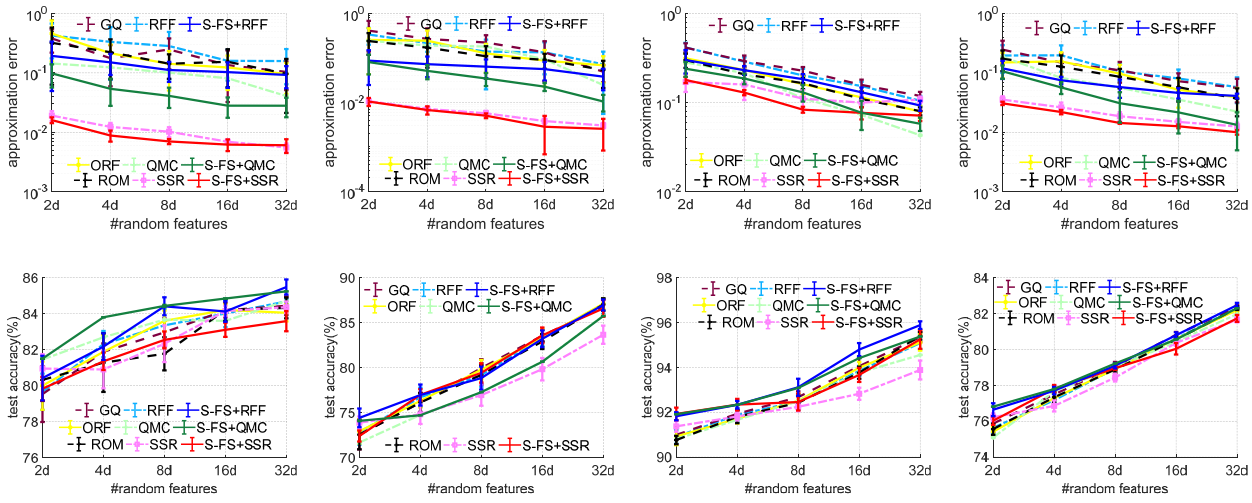
Results on arc-cosine kernel: Figure 9 shows the approximation error and test accuracy of all the compared algorithms across the first-order arc-cosine kernel on these four datasets. It can be found that, the compared algorithms across the arc-cosine kernel are generally inferior to them across the Gaussian kernel in terms of the approximation quality and generalization performance.

In sum, we experimentally validate that our stochastic rules are unbiased and achieve variance reduction in terms of the approximation error. Since "S-FS+QMC" is more efficient than



(i) *magic04* (j) *letter* (k) *ijcnn1* (l) *covtype*

Figure 8. Kernel approximation (top), time cost (middle), and test accuracy (bottom) across the Gaussian kernel.



(e) *magic04* (f) *letter* (g) *ijcnn1* (h) *covtype*

Figure 9. Kernel approximation error (top) and test accuracy (bottom) across the first-order arc-cosine kernel.

SSR on these datasets, and thus is demonstrated to achieve a good trade-off between the approximation quality and time cost.

7 Conclusion

We present deterministic/stochastic quadrature methods D-FS/S-FS based on the fully symmetric interpolatory rule to approximate the Gaussian kernel and the first-order arc-cosine kernel via the integration representation (1). Our third/fifth-degree deterministic rules achieve promising approximation quality while retaining the same time cost with RFF. Our S-FS rules exhibit variance reduction on the approximation error due to the used control variates technique, and performs well on real datasets. By studying the relations among the third-degree quadrature based methods, our unified framework mainly demonstrates that, 1) D-FS recovers SGQ by choosing suitable parameters; 2) SSR can be regarded as a doubly stochastic version of D-FS via a *random* projection scheme and a *randomized* generator.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. This work was supported in part by Research Council KU Leuven: Optimization frameworks for deep kernel machines C14/18/068; Flemish Government: FWO projects: GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant. This research received funding from the Flemish Government (AI Research Program). This work was supported in part by Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms), EU H2020 ICT-48 Network TAILOR (Foundations of Trustworthy AI - Integrating Reasoning, Learning and Optimization), Leuven.AI Institute; and in part by the National Natural Science Foundation of China 61977046, in part by National Science Foundation grant CCF-1704828 and CAREER Award CCF-2047910, and in part by SJTU Global Strategic Partnership Fund (2020 SJTU-CORNELL) and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

APPENDIX A
FIFTH-DEGREE RULE

When choosing $m = 2$ in Eq. (6), we obtain a fifth-degree rule $Q^{(2,d)}$ with $\|\boldsymbol{p}\|_1 \leq 2$ to further improve the approximation accuracy. To derive the fifth-degree rule, we cast it in three cases, i.e., $\|\boldsymbol{p}\|_1 = 0$, $\|\boldsymbol{p}\|_1 = 1$, and $\|\boldsymbol{p}\|_1 = 2$.

If $\|\boldsymbol{p}\|_1 = 0$, we have $p_i = 0$, $\boldsymbol{\lambda} = \boldsymbol{0}$, and $K = 0$. Then the weight $a_0^{(2,d)}$ is

$$\begin{aligned} a_0^{(2,d)} &= \sum_{\|\boldsymbol{u}\|_1 \leq 2} \prod_{i=1}^{d} \frac{b_{u_i}}{\prod_{j=0,\neq 0}^{u_i} \left(\lambda_0^2 - \lambda_j^2\right)} \\ &= 1 - \frac{d}{\lambda_1^2} + \frac{d(d-1)}{2\lambda_1^4} + \frac{d(3-\lambda_1^2)}{\lambda_1^2 \lambda_2^2}, \end{aligned} \tag{25}$$

where $b_2 = 3 - \lambda_1^2$ is obtained by Eq. (5). In our derivation, $\|\boldsymbol{u}\|_1 \leq 2$ is cast into three cases: $\boldsymbol{u} = \boldsymbol{0}$, $\|\boldsymbol{u}\|_1 = 1$, and $\|\boldsymbol{u}\|_1 = 2$ for calculation.

If $\|\boldsymbol{p}\|_1 = 1$, only one element of $\boldsymbol{p}$ is 1 and the remaining are zero. We thereby have $K = 1$ and $\boldsymbol{\lambda} = \lambda_1 \boldsymbol{e}_i$ with $i = 1, 2, \ldots, d$, where $\boldsymbol{e}_i$ is a unit vector with the i-th element being 1. Without loss of generality, assuming $\boldsymbol{p} = [1, 0, \cdots, 0]$, the weight $a_1^{(2,d)}$ is computed as

$$\begin{aligned} a_1^{(2,d)} &= \frac{1}{2} \sum_{\|\boldsymbol{u}\|_1 \leq 1} \prod_{i=1}^{d} \frac{b_{u_i+p_i}}{\prod_{j=0,\neq p_i}^{u_i+p_i} \left(\lambda_{p_i}^2 - \lambda_j^2\right)} \\ &= \frac{1}{2\lambda_1^2} + \frac{3-\lambda_1^2}{2\lambda_1^2\left(\lambda_1^2 - \lambda_2^2\right)} - \frac{d-1}{2\lambda_1^4}, \end{aligned} \tag{26}$$

where $\|\boldsymbol{u}\|_1 \leq 1$ is cast into two cases: $\boldsymbol{u} = \boldsymbol{0}$ and $\|\boldsymbol{u}\|_1 = 1$ for derivation.

If $\|\boldsymbol{p}\|_1 = 2$, the derivation is a little complex and we cast it into two cases. One is that there are two elements in $\boldsymbol{p}$ being 1, i.e., $p_i = p_j = 1$ with $i \neq j$. The other is only one element of $\boldsymbol{p}$ being 2, i.e., $p_i = 2$. For the $p_i = p_j = 1$ case, we have $K = 2$ and $\boldsymbol{\lambda} = \lambda_1 \boldsymbol{s}_l^+$ or $\boldsymbol{\lambda} = \lambda_1 \boldsymbol{s}_l^-$, where the point sets of $\boldsymbol{s}_l^+$ and $\boldsymbol{s}_l^-$ are given by

$$\begin{aligned} \{\boldsymbol{s}_l^+\}_{l=1}^{d(d-1)/2} &:= \{\boldsymbol{e}_i + \boldsymbol{e}_j : i < j,\ i, j = 1, 2, \cdots, d\} \\ \{\boldsymbol{s}_l^-\}_{l=1}^{d(d-1)/2} &:= \{\boldsymbol{e}_i - \boldsymbol{e}_j : i < j,\ i, j = 1, 2, \cdots, d\} \end{aligned}$$

Without loss of generality, assuming $\boldsymbol{p} = [1, 1, 0, \cdots, 0]$, the weight $a_2^{(2,d)}$ is

$$\begin{aligned} a_2^{(2,d)} &= \frac{1}{4} \prod_{i=1}^{d} \frac{b_{p_i}}{\prod_{j=0,\neq p_i}^{p_i} \left(\lambda_{p_i}^2 - \lambda_j^2\right)} \\ &= \frac{1}{4} \left[\frac{b_1}{(\lambda_1^2 - \lambda_0^2)}\right]^2 = \frac{1}{4\lambda_1^4}. \end{aligned} \tag{27}$$

For the $p_i = 2$ case, we have $K = 1$ and $\boldsymbol{\lambda} = \lambda_2 \boldsymbol{e}_i$. Without loss of generality, assuming $\boldsymbol{p} = [2, 0, \cdots, 0]$, the weight $a_3^{(2,d)}$ is computed as

$$a_3^{(2,d)} = \frac{1}{2} \prod_{i=1}^{d} \frac{b_{p_i}}{\prod_{j=0,\neq p_i}^{p_i} \left(\lambda_{p_i}^2 - \lambda_j^2\right)} = \frac{3-\lambda_1^2}{2\lambda_2^2(\lambda_2^2 - \lambda_1^2)}. \tag{28}$$

Accordingly, combining the derived weights in Eqs. (25), (26), (27), and (28), the fifth-degree full symmetric interpolatory rule is

$$\begin{aligned} Q^{(2,d)}(f) &= a_0^{(2,d)} f(\boldsymbol{0}) + a_1^{(2,d)} \sum_{i=1}^{d} \left[f\left(\lambda_1 \boldsymbol{e}_i\right) + f\left(-\lambda_1 \boldsymbol{e}_i\right)\right] \\ &+ a_2^{(2,d)} \sum_{i=1}^{d(d-1)/2} \left[f\left(\lambda_1 \boldsymbol{s}_i^+\right) + f\left(-\lambda_1 \boldsymbol{s}_i^+\right) + f\left(\lambda_1 \boldsymbol{s}_i^-\right) + f\left(-\lambda_1 \boldsymbol{s}_i^-\right)\right] \\ &+ a_3^{(2,d)} \sum_{i=1}^{d} \left[f\left(\lambda_2 \boldsymbol{e}_i\right) + f\left(-\lambda_2 \boldsymbol{e}_i\right)\right] \\ &:= a_0^{(2,d)} f(\boldsymbol{0}) + \sum_{j=1}^{2d} \left(a_1^{(2,d)} f(\boldsymbol{P}_{j,1}) + a_3^{(2,d)} f(\boldsymbol{P}_{j,3})\right) \\ &+ a_2^{(2,d)} \sum_{j=1}^{2d(d-1)} f(\boldsymbol{P}_{j,2}), \end{aligned}$$

where

$$\boldsymbol{P}_{j,1} = \begin{cases} \lambda_1 \boldsymbol{e}_i;\ a_i = \dfrac{1}{2d};\ 1 \leq i \leq d \\ -\lambda_1 \boldsymbol{e}_{i-d};\ d+1 \leq i \leq 2d \end{cases}$$

$$\boldsymbol{P}_{j,3} = \begin{cases} \lambda_2 \boldsymbol{e}_i;\ 1 \leq i \leq d \\ -\lambda_2 \boldsymbol{e}_{i-d};\ d+1 \leq i \leq 2d \end{cases}$$

$$\boldsymbol{P}_{j,2} = \begin{cases} \lambda_1 \left(\boldsymbol{e}_i + \boldsymbol{e}_t\right) & i, t = 1, \ldots, d; i < t \\ \lambda_1 \left(\boldsymbol{e}_i - \boldsymbol{e}_t\right) & i, t = 1, \ldots, d; i < t \\ \lambda_1 \left(-\boldsymbol{e}_i + \boldsymbol{e}_t\right) & i, t = 1, \ldots, d; i < t \\ \lambda_1 \left(-\boldsymbol{e}_i - \boldsymbol{e}_t\right) & i, t = 1, \ldots, d; i < t \end{cases}$$

Similar to the third-degree rule, we also choose λ_i by successive extensions of the one-dimensional 3-point Gauss–Hermite rule.

APPENDIX B
STATISTICAL GUARANTEES OF STOCHASTIC INTERPOLATORY RULES

This section includes three parts:

- in Section B.1, we prove Theorem 1, that is, our third-degree S-FS $\bar{R}_1(f)$ is an unbiased third degree rule for $I_d(f)$.

- in Section B.2, we prove Theorem 2 that gives the variance of our third-degree stochastic interpolatory rule, i.e., $\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})]$.

B.1 Proof of Theorem 1

Proof. We compute $I_d(\tilde{a}_{\boldsymbol{p}}^{(1,d)}(\boldsymbol{\omega}))$ as follows. For $\tilde{a}_0^{(1,d)}$

$$I_d(\tilde{a}_0^{(1,d)}) = \int_{\mathbb{R}^d} \left(1 - \frac{\sum_{i=1}^d \omega_i^2}{\lambda_1^2}\right) \mu(\mathrm{d}\boldsymbol{\omega}) = a_0^{(1,d)}.$$

For $\tilde{a}_1^{(1,d)}$, we have

$$I_d(\tilde{a}_1^{(1,d)}) = \frac{1}{2d\lambda_1^2} \int_{\mathbb{R}^d} \left(\sum_{i=1}^d \omega_i^2\right) \mu(\mathrm{d}\boldsymbol{\omega}) = a_1^{(1,d)}.$$

So we have $a_{\boldsymbol{p}}^{(1,d)} = I_d[\tilde{a}_{\boldsymbol{p}}^{(1,d)}(\boldsymbol{\omega})]$, and thus $Q^{(1,d)}(f) = I_d[M^{(1,d)}(f,\boldsymbol{\omega})]$. Due to $I(f) = \mathbb{E}_{\boldsymbol{\omega}\sim\mu}[f(\boldsymbol{\omega})]$, we have $Q^{(1,d)}(f) = \mathbb{E}_{\boldsymbol{\omega}\sim\mu}[M^{(1,d)}(f,\boldsymbol{\omega})]$. Based on this, the expectation of $R_1(f,\boldsymbol{\omega})$ is

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\omega}}[R_1(f,\boldsymbol{\omega})] &= \mathbb{E}[f(\boldsymbol{\omega})] - \mathbb{E}[M^{(1,d)}(f,\boldsymbol{\omega})] + \mathbb{E}\{I_d[M^{(1,d)}(f)]\} \\ &= I_d[f(\boldsymbol{\omega})] - Q^{(1,d)}(f) + Q^{(1,d)}(f) \\ &= I_d(f)\,.\end{aligned}$$

Accordingly, due to $\{\boldsymbol{\omega}_i\}_{i=1}^D \sim \mu$, the average $\bar{R}_1(f)$ is unbiased for $I_d(f)$.

Besides, if we choose $f(\boldsymbol{\omega}) = \boldsymbol{\omega}^{2\boldsymbol{u}}$ with $\|\boldsymbol{u}\|_1 \leq 1$, then we have $R_1(f,\boldsymbol{\omega}) = Q^{(1,d)}(f) = I_d[M^{(1,d)}(f,\boldsymbol{\omega})]$. If we choose $f(\boldsymbol{\omega}) = \boldsymbol{\omega}^{\boldsymbol{u}}$ in which at least one element of $\boldsymbol{u}$ is odd, we have $R_1(f,\boldsymbol{\omega}) = 0$. So it means that $R_1(f,\boldsymbol{\omega})$ is a third-degree rule for $I_d(f)$. Hence, $\bar{R}_1(f,\boldsymbol{\omega})$ is an unbiased third-degree stochastic rule for $I_d(f)$, which concludes the proof. □

B.2 Proof of Theorem 2

This section aims to prove Theorem 2 including two parts. In Section B.2.1, we present Lemma 1 that is used to prove Theorem 2. The proof of Theorem 2 can be found in Section B.2.2.

B.2.1 Proof of Lemma 1

To aid the proof of Theorem 2, we need the following lemma.

Lemma 1. *Denote* $\boldsymbol{\omega} = [\omega_1, \omega_2, \cdots, \omega_d]^\top \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, $\boldsymbol{z} := \boldsymbol{x} - \boldsymbol{y}/\sigma = [z_1, z_2, \cdots, z_d]^\top$, *and* $f(\boldsymbol{\omega}) = \cos(\boldsymbol{\omega}^\top \boldsymbol{z})$, *we have*

$$\mathbb{E}_{\boldsymbol{\omega}}\left(f(\boldsymbol{\omega}) \sum_{j=1}^d \omega_j^2\right) = e^{-\frac{\|\boldsymbol{z}\|_2^2}{2}} (d - \|\boldsymbol{z}\|_2^2)\,.$$

Proof. We expand $\mathbb{E}_{\boldsymbol{\omega}}\left(f(\boldsymbol{\omega}) \sum_{j=1}^d \omega_j^2\right)$ as

$$\mathbb{E}_{\boldsymbol{\omega}}\left(f(\boldsymbol{\omega}) \sum_{j=1}^d \omega_j^2\right) = \sum_{j=1}^d \mathbb{E}_{\boldsymbol{\omega}}\left[\omega_j^2 \cos(\boldsymbol{\omega}^\top \boldsymbol{z})\right]\,.$$

The j-th term $\omega_j^2 \cos(\boldsymbol{\omega}^\top \boldsymbol{z})$ can be reformulated as

$$\begin{aligned}\omega_j^2 \cos(\boldsymbol{\omega}^\top \boldsymbol{z}) &= \omega_j^2 \cos\left(\omega_j z_j + \sum_{t=1,\neq j}^d \omega_t z_t\right) \\ &= \omega_j^2 \left[\cos(\omega_j z_j) \cos\left(\sum_{t=1,\neq j}^d \omega_t z_t\right) - \sin(\omega_j z_j) \sin\left(\sum_{t=1,\neq j}^d \omega_t z_t\right)\right].\end{aligned} \tag{29}$$

Now we compute $\mathbb{E}_{\omega_j}[\omega_j^2 \cos(\omega_j z_j)]$ as follows.

$$\begin{aligned}\mathbb{E}[\omega_j^2 \cos(\omega_j z_j)] &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \omega_j^2 \cos(\omega_j z_j) e^{-\frac{\omega_j^\top \omega_j}{2}} \mathrm{d}\omega_j \\ &= \mathrm{Re}\left(\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \omega_j^2 e^{-\frac{z_j^2}{2}} e^{-\frac{(\omega_j - \mathrm{i}z_j)^2}{2}} \mathrm{d}\omega_j\right) \\ &= e^{-\frac{z_j^2}{2}} \mathbb{E}_{\omega_j}\left(\omega_j^2\right) \text{ with } \omega_j \sim \mathcal{N}(\mathrm{i}z_j, 1) \\ &= e^{-\frac{z_j^2}{2}} \left[1 + (\mathrm{i}z_j)^2\right] = e^{-\frac{z_j^2}{2}} (1 - z_j^2)\,,\end{aligned} \tag{30}$$

where i denotes the imaginary unit. Similarly, $\mathbb{E}[\omega_j^2 \sin(\omega_j z_j)]$ can be computed as

$$\mathbb{E}[\omega_j^2 \sin(\omega_j z_j)] = \mathrm{Im}\left(\int_{-\infty}^{\infty} \frac{\omega_j^2 e^{-\frac{z_j^2}{2}}}{\sqrt{2\pi}} e^{-\frac{(\omega_j - \mathrm{i}z_j)^2}{2}} \mathrm{d}\omega_j\right) = 0\,. \tag{31}$$

Besides, by virtue of the above derivation, or directly using Bochner theorem [17] for the Gaussian kernel $\mathbb{E}_{\boldsymbol{\omega}}[\cos(\boldsymbol{\omega}^\top \boldsymbol{z})] = e^{-\|\boldsymbol{z}\|_2^2/2}$, we have

$$\mathbb{E}_{\{\omega_1, \cdots, \omega_d\} \setminus \omega_j}\left[\cos\left(\sum_{t=1,\neq j}^d \omega_t z_t\right)\right] = e^{-\frac{\|\boldsymbol{z}\|_2^2 - z_j^2}{2}}\,. \tag{32}$$

Combine the above equations in Eqs. (29), (30), (31), and (32), we have

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\omega}}\left[\omega_j^2 cos(\boldsymbol{\omega}^\top \boldsymbol{z})\right] &= \mathbb{E}_{\omega_j}\left[\omega_j^2 \cos(\omega_j z_j)\right] \\ &\quad \times \mathbb{E}_{\{\omega_1, \cdots, \omega_d\} \setminus \omega_j}\left[\cos\left(\sum_{t=1,\neq j}^d \omega_t z_t\right)\right] \\ &= e^{-\frac{z_j^2}{2}} (1 - z_j^2) e^{-\frac{\|\boldsymbol{z}\|_2^2 - z_j^2}{2}} \\ &= (1 - z_j^2) e^{-\frac{\|\boldsymbol{z}\|_2^2}{2}}\,.\end{aligned} \tag{33}$$

Accordingly, we can conclude

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\omega}}\left(f(\boldsymbol{\omega}) \sum_{j=1}^d \omega_j^2\right) &= \sum_{j=1}^d \mathbb{E}_{\boldsymbol{\omega}}\left[\omega_j^2 \cos(\boldsymbol{\omega}^\top \boldsymbol{z})\right] \\ &= e^{-\frac{\|\boldsymbol{z}\|_2^2}{2}} \sum_{i=1}^d (1 - z_j^2)\,,\end{aligned}$$

which yields the final result. □

B.2.2 Proof of Theorem 2

In the next, we are ready to prove Theorem 2.

Proof. For ease of description, we use some short notations including $\sum_{i=1}^d[f] := \sum_{i=1}^d [f(\lambda_1 \boldsymbol{e}_i) + f(-\lambda_1 \boldsymbol{e}_i)]$, $Q := Q^{(1,d)}(f)$, and $z := \|\boldsymbol{z}\|_2$.

Recall RFF, its kernel approximation form is obtained via the Monte Carlo sampling

$$\frac{1}{D} \sum_{i=1}^D \cos(\boldsymbol{\omega}_i^\top \boldsymbol{z}), \quad \boldsymbol{\omega}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)\,.$$

By virtue of $\mathbb{E}[\cos(\boldsymbol{\omega}^\top \boldsymbol{z})] = e^{-z^2/2}$ and $\mathbb{V}[\cos(\boldsymbol{\omega}^\top \boldsymbol{z})] = (1 - e^{-z^2})^2/2$ [24], we have

$$\mathbb{E}[\mathrm{RFF}] = e^{-z^2/2}, \mathbb{V}[\mathrm{RFF}] = \frac{(1 - e^{-z^2})^2}{2D}\,.$$

Due to $\mathbb{V}[\bar{R}_1(f,\boldsymbol{\omega})] = 1/D(\mathbb{V}[R_1(f,\boldsymbol{\omega})])$ with $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, in the next we focus on $\mathbb{V}[R_1(f)]$. The variance of $R_1(f)$ can be formulated as

$$\begin{aligned}\mathbb{V}[R_1(f)] &= \mathbb{E}\left[f(\boldsymbol{\omega}) - M^{(1,d)}(f) + Q^{(1,d)}(f)\right]^2 \\ &\quad - \left\{\mathbb{E}\left[f(\boldsymbol{\omega}) - M^{(1,d)}(f) + Q^{(1,d)}(f)\right]\right\}^2 \\ &= \mathbb{E}\left[f(\boldsymbol{\omega}) - M^{(1,d)}(f) + Q^{(1,d)}(f)\right]^2 - \left\{\mathbb{E}[f(\boldsymbol{\omega})]\right\}^2 \\ &= \mathbb{E}\left[f(\boldsymbol{\omega}) - M^{(1,d)}(f) + Q^{(1,d)}(f)\right]^2 - e^{-z^2},\end{aligned}$$

where the used Gaussian kernel admits $f(\boldsymbol{\omega}) = \cos(\boldsymbol{\omega}^\top \boldsymbol{z})$.

Further, the above equation can be rewritten as

$$\begin{aligned}\mathbb{V}[R_1(f)] &= \mathbb{E}[f^2(\boldsymbol{\omega})] + \mathbb{E}[(M^{(1,d)}(f))^2] + [Q^{(1,d)}(f)]^2 \\ &\quad + 2\mathbb{E}[f(\boldsymbol{\omega})Q^{(1,d)}(f)] - 2\mathbb{E}[f(\boldsymbol{\omega})M^{(1,d)}(f)] \\ &\quad - 2\mathbb{E}[M^{(1,d)}(f)Q^{(1,d)}(f)] - e^{-z^2} \\ &= D\mathbb{V}[\text{RFF}] - [Q^{(1,d)}(f)]^2 + 2e^{-z^2/2}Q^{(1,d)}(f) \\ &\quad + \mathbb{E}[(M^{(1,d)}(f))^2] - 2\mathbb{E}[f(\boldsymbol{\omega})M^{(1,d)}(f)]\,.\end{aligned} \tag{34}$$

Hence, we need to bound $\mathbb{E}[(M^{(1,d)}(f))^2]$ and $\mathbb{E}[f(\boldsymbol{\omega})M^{(1,d)}(f)]$ in Eq. (34). First, byy expanding $[M^{(1,d)}(f)]^2$ in Eq. (12), we have

$$\begin{aligned}[M^{(1,d)}(f)]^2 &= \left(1 - \frac{\sum_{i=1}^d \omega_i^2}{\lambda_1^2}\right)^2 + \frac{(\sum_{i=1}^d \omega_i^2)^2}{4\lambda_1^4 d^2}\left\{\sum_{i=1}^d [f]\right\}^2 \\ &\quad + \left(1 - \frac{\sum_{i=1}^d \omega_i^2}{\lambda_1^2}\right)\frac{\sum_{i=1}^d \omega_i^2}{\lambda_1^2 d}\sum_{i=1}^d [f] \\ &= 1 - \frac{2\sum_{i=1}^d \omega_i^2}{\lambda_1^2} + \frac{(\sum_{i=1}^d \omega_i^2)^2}{\lambda_1^4} + \frac{(\sum_{i=1}^d \omega_i^2)^2}{4\lambda_1^4 d^2}\left\{\sum_{i=1}^d [f]\right\}^2 \\ &\quad + \frac{\sum_{i=1}^d \omega_i^2}{\lambda_1^2 d}\sum_{i=1}^d [f] - \frac{(\sum_{i=1}^d \omega_i^2)^2}{\lambda_1^4 d}\sum_{i=1}^d [f]\,,\end{aligned}$$

where we use $f(\mathbf{0}) = 1$ for the Gaussian kernel. Accordingly, we have

$$\begin{aligned}\mathbb{E}[M^{(1,d)}(f)]^2 &= 1 - \frac{2d}{\lambda_1^2} + \frac{d^2+2d}{\lambda_1^4} + \frac{d+2}{4\lambda_1^4 d}\left\{\sum_{i=1}^d [f]\right\}^2 \\ &\quad + \frac{1}{\lambda_1^2}\sum_{i=1}^d [f] - \frac{d+2}{\lambda_1^4}\sum_{i=1}^d [f]\end{aligned} \tag{35}$$

where $\sum_{i=1}^d \omega_i^2 \sim \chi(d)$, $\mathbb{E}(\sum_{i=1}^d \omega_i^2) = d$, $\mathbb{V}(\sum_{i=1}^d \omega_i^2) = 2d$ and $\mathbb{E}([\sum_{i=1}^d \omega_i^2]^2) = d^2 + 2d$.

Second, we estimate $\mathbb{E}[f(\boldsymbol{\omega})M^{(1,d)}(f)]$ in Eq. (34). The notation $f(\boldsymbol{\omega})M^{(1,d)}(f)$ is formulated as

$$f(\boldsymbol{\omega})M^{(1,d)}(f) = \left(1 - \frac{\sum_{i=1}^d \omega_i^2}{\lambda_1^2}\right)f(\boldsymbol{\omega}) + \frac{\sum_{i=1}^d \omega_i^2}{2\lambda_1^2 d}f(\boldsymbol{\omega})\sum_{i=1}^d [f]\,.$$

Accordingly, we have

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{\omega})M^{(1,d)}(f)] &= e^{-z^2/2} + \left(\frac{\sum_{i=1}^d [f]}{2\lambda_1^2 d} - \frac{1}{\lambda_1^2}\right)\mathbb{E}\left[f(\boldsymbol{\omega})\sum_{i=1}^d \omega_i^2\right] \\ &= e^{-z^2/2} + \frac{1}{\lambda_1^2}\left(-1 + \frac{\sum_{i=1}^d [f]}{2d}\right)e^{-\frac{z^2}{2}}(d - z^2)\,,\end{aligned} \tag{36}$$

where we use $f(\mathbf{0}) = 1$ and $\mathbb{E}[f(\boldsymbol{\omega})] = e^{-z^2/2}$ and Lemma 1. In our third-degree rule with $m = 1$, Eq. (7) implies $\sum_{i=1}^d [f] = 2\lambda_1^2 Q - 2\lambda_1^2 + 2d$, and thus we have

$$\mathbb{E}[M^{(1,d)}(f)]^2 = 1 + \frac{d+2}{d}(Q-1)^2 + 2(Q-1)\,,$$

and

$$-2\mathbb{E}[f(\boldsymbol{\omega})M^{(1,d)}(f)] = -2e^{-z^2/2} - \frac{2(Q-1)}{d}e^{-\frac{z^2}{2}}(d - z^2)\,.$$

Hence, combining the above equations into Eq. (34), we have

$$\begin{aligned}\mathbb{V}[\bar{R}_1(f)] - \mathbb{V}[\text{RFF}] &= \frac{2}{Dd}\left((1-Q)^2 - (1-Q)z^2 e^{-\frac{z^2}{2}}\right) \\ &= \frac{2}{Dd}\left(\left[(1-Q) - \frac{1}{2}z^2 e^{-\frac{z^2}{2}}\right]^2 - \frac{1}{4}z^4 e^{-z^2}\right)\,.\end{aligned} \tag{37}$$

Since Q is the approximation of $I_d(f) = k(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]$ for the Gaussian kernel, we can also consider $Q^{(1,d)}(f) \in [0,1]$. Note that, even if the estimation $Q^{(1,d)}(f)$ is out of $[0,1]$, we can still set it to $[0,1]$ by a threshold operator and thus $1 - Q \geq 0$. Finally, Eq. (37) can be formulated as

$$\mathbb{V}[\bar{R}_1(f)] - \mathbb{V}[\text{RFF}] < 0 \;\text{ when }\; 1 - Q < z^2 e^{-\frac{z^2}{2}}\,,$$

which concludes the proof.

□

APPENDIX C
PROOF OF THEOREM 3

To prove Theorem 3, we need the following lemma.

Lemma 2. *(Theorem 4.1 in [50]) Denote $\boldsymbol{x}^M$ and $\boldsymbol{s}^M$ as polynomials with total degree M, then the following integral satisfies*

$$\begin{aligned}I'\left(\boldsymbol{\omega}^d\right) &= \int_{\mathbb{R}^n} \omega_1^{\alpha_1}\omega_2^{\alpha_2}\cdots\omega_d^{\alpha_d}\exp\left(-\boldsymbol{\omega}^\top\boldsymbol{\omega}\right)\mathrm{d}\boldsymbol{x} \\ &= \int_0^\infty r^{d-1+M}\exp\left(-r^2\right)\mathrm{d}r\int_{U_d}\boldsymbol{s}^M\mathrm{d}\tau(\boldsymbol{s})\,,\end{aligned}$$

can be exactly calculated by the quadrature rules $I'(\boldsymbol{\omega}) = \sum_{j=1}^{N_g}\bar{a}_j f(\bar{\gamma})$ with $f'(\boldsymbol{\omega}) = \boldsymbol{\omega}^M$. Then the spherical integral can be expressed as $\int_{U_d}\boldsymbol{s}^M\mathrm{d}\tau(\boldsymbol{s}) = \sum_{j=1}^{N_p} a_{s,j}\left(\mathrm{s}_j\right)^M$ with the nodes $\boldsymbol{s}_j = \frac{\bar{\gamma}_j}{\|\bar{\gamma}_j\|_2}$ and and the weights $a_{s,j}$ of the spherical rule are

$$a_{s,j} = \frac{\bar{a}_j\left(\boldsymbol{s}_j\right)^M}{\int_0^\infty r^{d-1+M}\exp\left(-r^2\right)\mathrm{d}r} = \frac{\bar{a}_j\left(\boldsymbol{s}_j\right)^M}{\Gamma(d/2 + M/2)/2}\,,$$

where N_p is the number of projected quadrature non-zero nodes. Note that $N_p \leq N_g$.

Formally, we are ready to prove Theorem 3.

Proof. The integral in Eq. (1) can be reformulated as

$$\begin{aligned}I_d\left(f_{\boldsymbol{xy}}\right) &= \int_{\mathbb{R}^d} f_{\boldsymbol{xy}}(\boldsymbol{\omega})\mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \boldsymbol{I}_d)\mathrm{d}\boldsymbol{\omega} \\ &= \pi^{-\frac{d}{2}}\int_{\mathbb{R}^d} e^{-\boldsymbol{\omega}^\top\boldsymbol{\omega}} f(\sqrt{2}\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega}\,.\end{aligned}$$

Hence, the integral $I'(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} f(\boldsymbol{\omega}) \exp(-\boldsymbol{\omega}^\top \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}$ can be approximated by our third-degree D-FS in Eq. (7), that is

$$I'(\boldsymbol{\omega}) = \pi^{\frac{d}{2}} \int_{\mathbb{R}^d} f(\frac{\boldsymbol{\omega}}{\sqrt{2}}) \mathcal{N}(\boldsymbol{\omega}; \mathbf{0}, \boldsymbol{I}_d) \mathrm{d}\boldsymbol{\omega} \approx \sum_{i=1}^{2d+1} \bar{a}_i f(\bar{\boldsymbol{\gamma}}_i)$$
$$= (1 - \frac{d}{\lambda_1^2}) \pi^{\frac{d}{2}} f(\mathbf{0}) + \frac{\pi^{\frac{d}{2}}}{2\lambda_1^2} \sum_{i=1}^{d} \left(f(\frac{\lambda_1}{\sqrt{2}} \boldsymbol{e}_i) + f(-\frac{\lambda_1}{\sqrt{2}} \boldsymbol{e}_i) \right),$$

with

$$\begin{cases} \bar{\boldsymbol{\gamma}}_i = \mathbf{0};\ \bar{a}_i = (1 - \frac{d}{\lambda_1^2}) \pi^{\frac{d}{2}};\ i = 0 \\ \bar{\boldsymbol{\gamma}}_i = \frac{\lambda_1}{\sqrt{2}} \boldsymbol{e}_i;\ \bar{a}_i = \frac{\pi^{\frac{d}{2}}}{2\lambda_1^2};\ 1 \le i \le d \\ \bar{\boldsymbol{\gamma}}_i = -\frac{\lambda_1}{\sqrt{2}} \boldsymbol{e}_{i-d};\ \bar{a}_i = \frac{\pi^{\frac{d}{2}}}{2\lambda_1^2};\ d+1 \le i \le 2d. \end{cases}$$

By projecting $\bar{\boldsymbol{\gamma}}_i$ on the surface of the unit U_d sphere with an uniform random orthogonal matrix $\boldsymbol{Q}$, we have

$$\boldsymbol{s}_i = \frac{\boldsymbol{Q}\bar{\boldsymbol{\gamma}}_i}{\|\boldsymbol{Q}\bar{\boldsymbol{\gamma}}_i\|_2^2} = \begin{cases} \boldsymbol{Q}\boldsymbol{e}_i;\ 1 \le i \le d, \\ -\boldsymbol{Q}\boldsymbol{e}_{i-d};\ d+1 \le i \le 2d, \end{cases} \tag{38}$$

with $\|\boldsymbol{Q}\bar{\boldsymbol{\gamma}}_i\|_2 = \|\bar{\boldsymbol{\gamma}}_i\|_2$. Note that the point at the origin has been omitted. By Lemma 2, for the third-degree, the polynomial degree M is set to 2. Accordingly, the weight $a_{s,j}$ of the spherical rule can be obtained by

$$a_{s,j} = \frac{\bar{a}_j \left(\boldsymbol{s}_j\right)^M}{\Gamma(d/2 + M/2)/2} = \frac{\pi^{\frac{d}{2}}}{2\lambda_1^2} \frac{\lambda_1^2}{2} \Big/ \left(\frac{\frac{d}{2}\Gamma(\frac{d}{2})}{2} \right) = \frac{|U_d|}{2d}. \tag{39}$$

Hence, using Eq. (38) and Eq. (39) yields the spherical rule

$$I_{\boldsymbol{Q},U_d}(s) = \frac{|U_d|}{2d} \sum_{j=1}^{d} \left[s\left(\boldsymbol{Q}\boldsymbol{e}_j\right) + s\left(-\boldsymbol{Q}\boldsymbol{e}_j\right) \right],$$

which is identical to the third-degree stochastic spherical integration rule in Eq. (22). □

REFERENCES

[1] Bernhard Schölkopf and Alexander J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT Press, 2003.

[2] Johan A.K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle, *Least Squares Support Vector Machines*, World Scientific, 2002.

[3] Mehran Kafai and Kave Eshghi, "CROification: accurate kernel classification with the efficiency of sparse linear SVM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 34–48, 2019.

[4] Fanghui Liu, Lei Shi, Xiaolin Huang, Jie Yang, and Johan A.K. Suykens, "Generalization properties of hyper-rkhs and its applications," *Journal of Machine Learning Research*, vol. 22, no. 140, pp. 1–38, 2021.

[5] Zhiyuan Dang, Xiang Li, Bin Gu, Cheng Deng, and Heng Huang, "Large-scale nonlinear AUC Maximization via triply stochastic gradients," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2020.

[6] Ali Rahimi and Benjamin Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, 2007, pp. 1177–1184.

[7] David Lopez-Paz, Suvrit Sra, Alex J. Smola, Zoubin Ghahramani, and Bernhard Schölkopf, "Randomized nonlinear component analysis," in *International Conference on Machine Learning*, 2014, pp. 1359–1367.

[8] Yitong Sun, Anna Gilbert, and Ambuj Tewari, "But how does it work in theory? Linear SVM with random features," in *Advances in Neural Information Processing Systems*, 2018, pp. 3383–3392.

[9] Arthur Jacot, Franck Gabriel, and Clément Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 8571–8580.

[10] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Russ R. Salakhutdinov, and Ruosong Wang, "On exact computation with an infinitely wide neural net," in *Advances in Neural Information Processing Systems*, 2019, pp. 8139–8148.

[11] Amir Zandieh, Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin, "Scaling neural tangent kernels via sketching and random features," *arXiv preprint arXiv:2106.07880*, 2021.

[12] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, and Weller Adrian, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021.

[13] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong, "Random feature attention," in *International Conference on Learning Representations*, 2021, pp. 1–19.

[14] Yueming Lyu, "Spherical structured feature maps for kernel approximation," in *34th International Conference on Machine Learning*. JMLR.org, 2017, pp. 2256–2264.

[15] Marina Munkhoeva, Yermek Kapushev, Evgeny Burnaev, and Ivan Oseledets, "Quadrature-based features for kernel approximation," in *Advances in Neural Information Processing Systems*, 2018, pp. 9147–9156.

[16] Youngmin Cho and Lawrence K Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems*, 2009, pp. 342–350.

[17] Salomon Bochner, *Harmonic Analysis and the Theory of Probability*, Courier Corporation, 2005.

[18] Philip J. Davis and Philip Rabinowitz, *Methods of numerical integration*, Courier Corporation, 2007.

[19] Florian Heiss and Viktor Winschel, "Likelihood approximation by numerical integration on sparse grids," *Journal of Econometrics*, vol. 144, no. 1, pp. 62–80, 2008.

[20] Alan Genz and Bradley D Keister, "Fully symmetric interpolatory rules for multiple integrals over infinite regions with gaussian weight," *Journal of Computational and Applied Mathematics*, vol. 71, no. 2, pp. 299–309, 1996.

[21] Erich Novak and Klaus Ritter, "Simple cubature formulas with high polynomial exactness," *Constructive approximation*, vol. 15, no. 4, pp. 499–522, 1999.

[22] Tri Dao, Christopher M. De Sa, and Christopher Ré, "Gaussian quadrature for kernel features," in *Advances in neural information processing systems*, 2017, pp. 6107–6117.

[23] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan A.K. Suykens, "Random features for kernel approximation: A survey on algorithms, theory, and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2021.

[24] Felix Xinnan Yu, Ananda Theertha Suresh, Krzysztof Choromanski, Daniel Holtmannrice, and Sanjiv Kumar, "Orthogonal random features," in *Advances in Neural Information Processing Systems*, 2016, pp. 1975–1983.

[25] Krzysztof M. Choromanski, Mark Rowland, and Adrian Weller, "The unreasonable effectiveness of structured random orthogonal embeddings," in *Advances in Neural Information Processing Systems*, 2017, pp. 219–228.

[26] Harald Niederreiter, *Random number generation and quasi-Monte Carlo methods*, vol. 63, SIAM, 1992.

[27] Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W. Mahoney, "Quasi-Monte Carlo feature maps for shift-invariant kernels," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 4096–4133, 2016.

[28] Russel E Caflisch et al., "Monte carlo and quasi-monte carlo methods," *Acta numerica*, vol. 1998, pp. 1–49, 1998.

[29] Gunther Leobacher and Friedrich Pillichshammer, *Introduction to quasi-Monte Carlo integration and applications*, Springer, 2014.

[30] Josef Dick et al., "Higher order scrambled digital nets achieve the optimal rate of the root mean square error for smooth integrands," *Annals of Statistics*, vol. 39, no. 3, pp. 1372–1398, 2011.

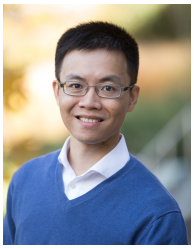
[31] Krzysztof Choromanski, Mark Rowland, Wenyu Chen, and Adrian Weller, "Unifying orthogonal Monte Carlo methods," in *International Conference on Machine Learning*, 2019, pp. 1203–1212.

[32] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh, "Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees," in *International Conference on Machine Learning*, 2017, pp. 253–262.

[33] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco, "On fast leverage score sampling and optimal learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 5672–5682.
[34] Fanghui Liu, Xiaolin Huang, Yudong Chen, Jie Yang, and Johan A.K. Suykens, "Random Fourier features via fast surrogate leverage weighted sampling," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020, pp. 4844–4851.
[35] Edouard Pauwels, Francis Bach, and Jean-Philippe Vert, "Relating leverage scores and density using regularized christoffel functions," in *Advances in Neural Information Processing Systems*, 2018, pp. 1663–1672.
[36] Tamás Erdélyi, Cameron Musco, and Christopher Musco, "Fourier sparse leverage scores and approximate kernel learning," in *Advances in Neural Information Processing Systems*, 2020.
[37] Gwynne Evans, *Practical numerical integration*, Wiley New York, 1993.
[38] Alan Genz and John Monahan, "Stochastic integration rules for infinite regions," *SIAM Journal on Scientific Computing*, vol. 19, no. 2, pp. 426–439, 1998.
[39] Toni Karvonen and Simo Sarkka, "Fully symmetric kernel quadrature," *SIAM Journal on Scientific Computing*, vol. 40, no. 2, pp. A697–A720, 2018.
[40] Bertrand Gauthier and Johan A.K. Suykens, "Optimal quadrature-sparsification for integral operator approximation," *SIAM Journal on Scientific Computing*, vol. 40, no. 5, pp. A3636–A3674, 2018.
[41] Ronald Cools, "Constructing cubature formulae: the science behind the art," *Acta Numerica*, vol. 6, pp. 1–54, 1997.
[42] Ienkaran Arasaratnam and Simon Haykin, "Cubature kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.
[43] Reuven Y Rubinstein and Ruth Marcus, "Efficiency of multivariate control variates in monte carlo simulation," *Operations Research*, vol. 33, no. 3, pp. 661–677, 1985.
[44] Noureddine El Karoui, "The spectrum of kernel random matrices," *Annals of Statistics*, vol. 38, no. 1, pp. 1–50, 2010.
[45] Arthur Jacot, Berfin Şimşek, Francesco Spadaro, Clément Hongler, and Franck Gabriel, "Kernel alignment risk estimator: Risk prediction from training data," in *Advances in Neural Information Processing Systems*, 2020, pp. 15568–15578.
[46] Tengyuan Liang and Alexander Rakhlin, "Just interpolate: Kernel "ridgeless" regression can generalize," *Annals of Statistics*, vol. 48, no. 3, pp. 1329–1347, 2020.
[47] Chih-chung Chang and Chih-Jen Lin, "Ijcnn 2001 challenge: Generalization ability and text decoding," in *International Joint Conference on Neural Networks*. IEEE, 2001, vol. 2, pp. 1031–1036.
[48] Alan Genz, "Methods for generating random orthogonal matrices," in *Monte-Carlo and Quasi-Monte Carlo Methods 1998*, pp. 199–213. 1998.
[49] Jian Zhang, Avner May, Tri Dao, and Christopher Re, "Low-precision random Fourier features for memory-constrained kernel approximation," in *22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 1264–1274.
[50] Bin Jia, Ming Xin, and Yang Cheng, "Relations between sparse-grid quadrature rule and spherical-radial cubature rule in nonlinear Gaussian estimation," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 199–204, 2015.

Fanghui Liu (M'19-) received the B.E. degree in Automation from Harbin Institute of Technology, China, and the Ph.D. degree from Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China, in 2014 and 2019, respectively. He worked as a postdoctoral researcher in ESAT-STADIUS, KU Leuven, Belgium from 2019 to 2021, and is currently a post-doctoral researcher with the Laboratory for Information and Inference Systems at École Polytechnique Fédérale de Lausanne (EPFL). He has a broad research interest in statistical machine learning on kernel methods and learning theory.

Xiaolin Huang (S'10-M'12-SM'18) received the B.S. degree in control science and engineering, and the B.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China in 2006. In 2012, he received the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China. From 2012 to 2015, he worked as a postdoctoral researcher in ESAT-STADIUS, KU Leuven, Leuven, Belgium. After that he was selected as an Alexander von Humboldt Fellow and working in Pattern Recognition Lab, the Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, where he was appointed as a group head. From 2016, he has been an Associate Professor at Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China. In 2017, he has been awarded as "1000-Talent"(Young Program). His current research areas include machine learning, optimization, and their applications.



Yudong Chen is an Associate Professor with the School of Operations Research and Information Engineering at Cornell University. He obtained his Ph.D. degree in Electrical and Computer Engineering in 2013 from The University of Texas at Austin, and M.S. and B.S. degrees in Control Science and Engineering from Tsinghua University. He was a postdoctoral scholar in the Electrical Engineering and Computer Sciences department at the University of California, Berkeley from 2013 to 2015. He has served as area chairs for AAAI, AISTATS and NeurIPS. His research work lies in machine learning, reinforcement learning, high-dimensional statistics, and optimization, with applications in network scheduling, wireless communication, and financial systems. He received a National Science Foundation CAREER award.

Johan A. K. Suykens (SM'05-F'15) was born in Willebroek Belgium, May 18 1966. He received the master degree in Electro-Mechanical Engineering and the PhD degree in Applied Sciences from the Katholieke Universiteit Leuven, in 1989 and 1995, respectively. In 1996 he has been a Visiting Postdoctoral Researcher at the University of California, Berkeley. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a full Professor with KU Leuven.

He is author of the books "Artificial Neural Networks for Modelling and Control of Non-linear Systems" (Kluwer Academic Publishers) and "Least Squares Support Vector Machines" (World Scientific), co-author of the book "Cellular Neural Networks, Multi-Scroll Chaos and Synchronization" (World Scientific) and editor of the books "Nonlinear Modeling: Advanced Black-Box Techniques" (Kluwer Academic Publishers), "Advances in Learning Theory: Methods, Models and Applications" (IOS Press) and "Regularization, Optimization, Kernels, and Support Vector Machines" (Chapman & Hall/CRC). In 1998 he organized an International Workshop on Nonlinear Modelling with Time-series Prediction Competition. He has served as associate editor for the IEEE Transactions on Circuits and Systems (1997-1999 and 2004-2007), the IEEE Transactions on Neural Networks (1998-2009), the IEEE Transactions on Neural Networks and Learning Systems (from 2017) and the IEEE Transactions on Artificial Intelligence (from April 2020). He received an IEEE Signal Processing Society 1999 Best Paper Award, a 2019 Entropy Best Paper Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He has served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven 2002), as a program co-chair for the International Joint Conference on Neural Networks 2004 and the International Symposium on Nonlinear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007, a co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning, and chair of ROKS 2013. He has been awarded an ERC Advanced Grant 2011 and 2017, has been elevated IEEE Fellow 2015 for developing least squares support vector machines, and is ELLIS Fellow. He is currently serving as program director of Master AI at KU Leuven.