WPEDU-PR(17) 18400

A Yurova et al.

Stable Evaluation of Gaussian Radial Basis Functions Using Hermite Polynomials

Preprint of Paper to be submitted for publication in
SIAM Journal on Scientific Computing

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

This document is intended for publication in the open literature. It is made available on the clear understanding that it may not be further circulated and extracts or references may not be published prior to publication of the original when applicable, or without the consent of the Publications Officer, EUROfusion Programme Management Unit, Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK or e-mail Publications.Officer@euro-fusion.org

Enquiries about Copyright and reproduction should be addressed to the Publications Officer, EUROfusion Programme Management Unit, Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK or e-mail Publications.Officer@euro-fusion.org

The contents of this preprint and all other EUROfusion Preprints, Reports and Conference Papers are available to view online free at http://www.euro-fusionscipub.org. This site has full search facilities and e-mail alert options. In the JET specific papers the diagrams contained within the PDFs on this site are hyperlinked

STABLE EVALUATION OF GAUSSIAN RADIAL BASIS FUNCTIONS USING HERMITE POLYNOMIALS*

ANNA YUROVA† AND KATHARINA KORMANN†

Abstract. Gaussian radial basis functions can be an accurate basis for multivariate interpolation. In practise, high accuracies are often achieved in the flat limit where the interpolation matrix becomes increasingly ill-conditioned. Stable evaluation algorithms have been proposed by Fornberg, Larsson & Flyer based on a Chebyshev expansion of the Gaussian basis and by Fasshauer & McCourt based on a Mercer expansion with Hermite polynomials. In this paper, we propose another stabilization algorithm based on Hermite polynomials but derived from the generating functions of Hermite polynomials. The new expansion does not require a complicated choice of parameters and offers a simple extension to high-dimensional tensor grids as well as a generalization for anisotropic multivariate basis functions using Hagedorn generating functions.

Key words. radial basis functions, stable evaluation, ill-conditioning, Hermite polynomials, generating functions

AMS subject classifications. 65D05, 65D15, 65F35, 41A63, 33C45

1. Introduction. Multivariate interpolation is a topic of recent interest, for instance appearing in the semi-Lagrangian solution of high-dimensional advection problems. Gaussian radial basis function interpolation generalizes to higher dimensions in a simple way and can yield spectral accuracy [4]. However, it is known that rather small values of the *shape parameter* (width of the Gaussian) are often required for optimal accuracy. In this case the basis functions become increasingly flat and the interpolation matrix becomes ill-conditioned. Tarwater has described this phenomenon in 1985 [19] and the problem has been extensively studied in the literature (see [6] for a review). The eigenvalues of the interpolation matrix are proportional to increasing powers of the shape parameter as has been quantified by Fornberg and Zuev [11].

A direct collocation solution of the interpolation problem, referred to as RBF-Direct in the literature, computes the expansion coefficients of the Gaussian interpolant by inverting the collocation matrix and then evaluating the expansion. This procedure suffers from inaccuracies in floating point arithmetics due to the ill-conditioning of the matrices. In recent years, several algorithms have been proposed to stabilize the computations of the radial basis functions interpolation problems. These stabilization algorithms directly evaluate the interpolant in a sequence of well-conditioned steps by a transformation to a different basis. The first method was the Contour-Padé approximation proposed by Fornberg and Wright for multiquadrics [10]. Later Fornberg and Piret [9] proposed the so-called RBF-QR method for stable interpolation with Gaussians on the sphere. The Gaussian basis is expanded in spherical harmonics. The expansion allows to isolate the ill-conditioning in a diagonal matrix that can be inverted in a well-conditioned procedure.

The method has been extended to more general domains in one to three dimensions by Fornberg, Larsson & Flyer [7]. This expansion is based on a combination

*Submitted to the editors DATE.

Funding: This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

†Max-Planck-Institut für Plasmaphysik, 85748 Garching, Germany, and Technische Universitat München, Zentrum Mathematik, 85748 Garching, Germany. (ayurova@ipp.mpg.de, katharina.kormann@ipp.mpg.de).

of Chebyshev polynomials and spherical harmonics. This method will be referred to as Chebyshev-QR in this paper. The technique has also been used for the stable computation of difference matrices by Larsson et al. [14] and by Fornberg et al. [8] for RBF-FD stencils. In order to treat complex domains, the Chebyshev-QR method has been combined with a partition of unity approach by Larsson, Shcherbakov & Heryudono [15].

Fasshauer and McCourt [5] have developed another RBF-QR method, called Gauss-QR, that relies on a Mercer expansion of the Gaussian kernel. The basis transformation involves exponentially scaled Hermite polynomials. Compared to the Chebyshev-QR method by Fornberg, Larsson & Flyer [7], the Gauss-QR algorithm extends to higher dimensions in a simpler way and does not require transformation of the computational domain into the unit squere. On the other hand, the method introduces an additional parameter that needs to be hand-tuned. In this paper, we propose an expansion built on Hermite generating functions. Our new basis is similar to the one in [5] with the difference that only one parameter is introduced that can easily be chosen. Our focus is on enabling high-dimensional interpolation where we propose a tensor product approach that yields a memory-sparse representation of the interpolation matrices. Moreover, we propose a stabilization algorithm for anisotropic multivariate Gaussians: We consider an expansion of so-called Hagedorn wave packets, combination of multivariate versions of Hermite polynomials and anisotropic Gaussians that have been discussed in semi-classical quantum dynamics [16, 12]. As for Hermite polynomials, Hagedorn generating functions have been derived in [3, 13] that enable a generalization of our Hermite expansion to the anisotropic case.

The paper is organized as follows: In the next section we introduce our HermiteGF expansion of the radial basis functions and discuss its convergence. In section 3, we discuss two main ideas of truncating the expansion: one based on a direct transform to the HermiteGF basis and another following the RBF-QR idea. Extentions to multivariate interpolation are discussed in section 4. Numerical results show the accuracy of our method in section 5 and computational complexity and performance are discussed in section 6. Finally, section 7 concludes the paper.

2. HermiteGF expansion.

In this section, similarly to [9, 7, 5], we propose an expansion of the radial basis functions in a "better" basis, that spans the same space, but avoids instabilities related to the flat limit.

2.1. Interpolation problem.

Before introducing our expansion of the Gaussian basis, let us briefly define the interpolation problem in one dimension. Given a set $\{\phi_k(x)\}_{k=1}^N$ of basis functions and the values $\{f_i\}$ of the function f at points $\{x_i^{\text{col}}\}_{i=1}^N$ we seek to find an interpolant of the following form,

$$s(x) = \sum_{k=1}^{N} \alpha_k \phi_k(x), \tag{1}$$

such that it satisfies the N collocation conditions,

$$s(x_i^{\text{col}}) = f_i \quad \text{for} \quad i = 1 \dots N. \tag{2}$$

The straightforward approach is to find the coefficients $\{\alpha_i\}$ as a solution of the linear system,

$$\Phi^{\text{col}} \alpha = f, \quad \text{with} \quad \Phi_{ij}^{\text{col}} = \phi_j(x_i^{\text{col}}). \tag{3}$$

The matrix Φ^{col} is called *collocation matrix*. Then, the interpolant (1) can be evaluated at any point of the domain.

Here we focus on Gaussian radial basis functions,

$$\phi_k(x) = \exp(-\varepsilon^2 \|x - x_k^{\text{cen}}\|^2), \tag{4}$$

with shape parameter $\varepsilon > 0$.

2.2. Definition. Let $\{h_n\}_{n\geq 0}$ be the Hermite polynomials in the physicists' version, that satisfy the following recurrence relation,

$$h_{n+1}(x) = 2xh_n(x) - 2nh_{n-1}(x). \tag{5}$$

The following upper bound holds for the magnitude of Hermite polynomials [1, Expression 22.14.17],

$$|h_n(x)| \leq e^{\frac{x^2}{2}} c 2^{\frac{n}{2}} \sqrt{n!}, \quad c \approx 1.086435. \tag{6}$$

The factors $\sqrt{n!}$, $2^{n/2}$ grow very fast with n. Therefore, in order to avoid overflow for large n, it is advantageous for numerical computations to scale the Hermite polynomials with the factor $\sqrt{2^n n!}$. Let us therefore define the following basis functions,

$$H_n^{\gamma,\varepsilon}(x) = \frac{1}{\sqrt{2^n n!}} h_n(\gamma x) \mathrm{e}^{-\varepsilon^2 x^2}, \quad \varepsilon > 0,\ \gamma > 0, \tag{7}$$

that we refer to as *HermiteGF functions*. Based on the generating function theory we derive an infinite expansion of the one dimensional Gaussian RBFs in the new HermiteGF basis $\{H_n^{\gamma,\varepsilon}\}$.

THEOREM 2.1. *HermiteGF expansion*
For all $\varepsilon > 0$, $\gamma > 0$, $y \in \mathbb{R}$, we have a pointwise expansion

$$\phi_y(x) = \mathrm{e}^{-\varepsilon^2 (x-y)^2} = \exp\left(\varepsilon^2 y^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) \sum_{n\geq 0} \frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n \sqrt{n!}} y^n H_n^{\gamma,\varepsilon}(x). \tag{8}$$

The RBF interpolant $s(x)$ can then be pointwise computed as,

$$s(x) = \sum_{k=1}^{N} \alpha_k \exp\left(\varepsilon^2 (x_k^{\text{cen}})^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) \sum_{n\geq 0} \frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n \sqrt{n!}} (x_k^{\text{cen}})^n H_n^{\gamma,\varepsilon}(x), \tag{9}$$

where $\{x_k^{\text{cen}}\}_{k=1}^N$ are the centers of the RBFs.

Proof. The Hermite polynomial's generating function is given by (see e.g. [1, Expression 22.9.17],

$$\mathrm{e}^{2st - t^2} = \sum_{n\geq 0} \frac{t^n}{n!} h_n(s) \tag{10}$$

Choosing $t = \frac{\varepsilon^2 y}{\gamma}$ and $s = \gamma x$, we obtain

$$\sum_{n\geq 0} \frac{\varepsilon^{2n}}{\gamma^n n!} y^n h_n(\gamma x) = \exp\left(2\varepsilon^2 yx - \frac{\varepsilon^4 y^2}{\gamma^2}\right). \tag{11}$$

Hence, we get

$$\exp\left(\varepsilon^2 y^2\left(\frac{\varepsilon^2}{\gamma^2}-1\right)\right)\sum_{n\geq 0}\frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n\sqrt{n!}}y^n H_n^{\gamma,\varepsilon}(x) \tag{12}$$

$$=\exp\left(\varepsilon^2 y^2\left(\frac{\varepsilon^2}{\gamma^2}-1\right)\right)\sum_{n\geq 0}\frac{\varepsilon^{2n}}{\gamma^n n!}y^n h_n(\gamma x)\mathrm{e}^{-\varepsilon^2 x^2} \tag{13}$$

$$=\exp\left(\varepsilon^2 y^2\left(\frac{\varepsilon^2}{\gamma^2}-1\right)+2\varepsilon^2 yx-\frac{\varepsilon^4 y^2}{\gamma^2}-\varepsilon^2 x^2\right)=\mathrm{e}^{-\varepsilon^2(x-y)^2}, \tag{14}$$

which proves expansion (8). Using the expansion (8) in the interpolant (1), we get the representation (9). ∎

2.3. Basis centering. The Hermite polynomials are symmetric with respect to the axis $x = 0$. Due to the growth in the basis it is advantageous to center the interpolation interval $[A, B]$ at 0. For this reason, we symmetrize the basis around $x_0 := \frac{A+B}{2}$. The RBF $\phi_k(x)$ can be expanded as:

$$\phi_k(x)=\mathrm{e}^{-\varepsilon^2(x-x_k^{\mathrm{cen}})^2}=\mathrm{e}^{-\varepsilon^2(x-x_0-(x_k^{\mathrm{cen}}-x_0))^2} \tag{15}$$

$$=\mathrm{e}^{\left(\varepsilon^2(x_k^{\mathrm{cen}}-x_0)^2\left(\frac{\varepsilon^2}{\gamma^2}-1\right)\right)}\sum_{n\geq 0}\frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n\sqrt{n!}}(x_k^{\mathrm{cen}}-x_0)^n H_n^{\gamma,\varepsilon}(x-x_0). \tag{16}$$

Then, we have,

$$x-x_0\in\left[-\frac{B-A}{2},\frac{B-A}{2}\right], \tag{17}$$

i.e. the HermiteGF functions $H_n^{\gamma,\varepsilon}$ are evaluated on an interval centered around 0. For the sake of simplicity, we further consider symmetric intervals $[-L, L]$. However, the procedure can be applied to functions on arbitrary intervals by adding this translation by x_0.

2.4. The parameter γ. The parameter γ in the basis $\{H_n^{\gamma,\varepsilon}\}$ allows a control over the evaluation domain of the Hermite polynomials. When choosing γ, one has to consider two counteracting effects. For small values of γ, the collocation points are close which can yield ill-conditioning since the values of the basis functions at the collocation points are too similar. On the other hand, Hermite polynomials take very large values on large domains which can lead to an overflow. An optimal balance depends on the particular function and the number of basis functions. However, from our numerical experience, choosing γL between 3 and 5 yields good approximation quality in most cases.

2.5. Connection to Fasshauer and McCourt. An expansion of similar type was used by Fasshauer and McCourt [5] also for the stabilization of the RBF interpolation. Instead of the HermiteGF-expansion, an eigenfunction expansion of Gaussian RBF was used. The corresponding eigenfunctions look as follows,

$$\phi_n(x)=\frac{\sqrt{\beta}}{\sqrt{2^n n!}}\exp(-\delta^2 x^2)h_{n-1}(\alpha\beta x), \tag{18}$$

The parameters α and ε need to be chosen by the user. Then, the parameters β and δ are deduced from α and ε according to the formula:

$$\beta=\left(1+\frac{4\varepsilon^2}{\alpha^2}\right)^{1/4},\quad \delta^2=\frac{\alpha^2}{2}(\beta^2-1). \tag{19}$$

We now try to match that basis with the basis functions arising from the HermiteGF-expansion. To match the width of the exponential in the two expansions we need,

$$\delta = \varepsilon \tag{20}$$

and to match the argument of the Hermite polynomials it is necessary to have,

$$\alpha\beta = \gamma. \tag{21}$$

We now compute the values of the parameters α, β from the relations (19),

$$\varepsilon^2 = \frac{\gamma^2 - \alpha^2}{2} \implies \alpha = \sqrt{\gamma^2 - 2\varepsilon^2}. \tag{22}$$

The parameter β can then be calculated as

$$\beta = \left(1 + \frac{4\varepsilon^2}{\alpha^2}\right)^{1/4} = \left(1 + \frac{4\varepsilon^2}{\gamma^2 - 2\varepsilon^2}\right)^{1/4} \tag{23}$$

However, from the relation (21) β must be,

$$\beta = \frac{\gamma}{\alpha} = \frac{\gamma}{\sqrt{\gamma^2 - 2\varepsilon^2}} \tag{24}$$

One can see that if $\varepsilon \to 0$, both expressions converge to 1. However in a general case the values of expressions (23) and (24) for the parameter β differ. Hence, we cannot match both (20) and (21) at the same time. This means that there is no direct correspondence between the basis functions arising from the HermiteGF expansion and the ones used by Fasshauer and McCourt [5].

2.6. Convergence of the truncated HermiteGF expansion. In this section, we check the convergence of the expansion (9), if we cut the expansion (9) after M terms,

$$s(x) \approx s_M^\gamma(x) := \sum_{k=1}^{N} \alpha_k \exp\left(\varepsilon^2 (x_k^{\text{cen}})^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) \sum_{n=0}^{M-1} \frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n \sqrt{n!}} (x_k^{\text{cen}})^n H_n^{\gamma,\varepsilon}(x). \tag{25}$$

We later refer to $s_M^\gamma(x)$ as *HermiteGF interpolant*. Let us now prove that for a large enough M the approximation $s_M^\gamma(x)$ converges to $s(x)$.

THEOREM 2.2. *Let s be the RBF interpolant,*

$$s(x) = \sum_{k=1}^{N} \alpha_k \phi_k(x) = \sum_{k=1}^{N} \alpha_k \mathrm{e}^{-\varepsilon^2 (x - x_k^{\text{cen}})^2} \tag{26}$$

with $\{x_k^{\text{cen}}\}_{k=1}^N \subset [-L, L]$.

For all $x \in [-L, L]$, the HermiteGF interpolant $s_M^\gamma(x)$ given by (25) converges pointwise to $s(x)$, i.e.

$$|s(x) - s_M^\gamma(x)| \to 0 \quad \textit{for} \quad M \to \infty \tag{27}$$

For $\gamma > \sqrt{2}\varepsilon^2 L$, we also have the estimate

$$|s(x) - s_M^\gamma(x)| < C \frac{q^M}{(1-q)\sqrt{M!}}, \tag{28}$$

where $q = \frac{\sqrt{2}\varepsilon^2 L}{\gamma}$ and $C = C(\gamma, \varepsilon, L, \{\alpha_k\}) \in \mathbb{R}$ is a constant.

Proof. We build up the proof analogously to [18, § 3.1]. Combining (9) and (25), for each x we have,

(29)
$$|s(x) - s_M(x)| = \left| \sum_{k=1}^{N} \alpha_k \sum_{n=M}^{\infty} \exp\left(\varepsilon^2 (x_k^{\text{cen}})^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) \frac{\varepsilon^{2n}}{\gamma^n n!} . (x_k^{\text{cen}})^n h_n(\gamma x) \mathrm{e}^{-\varepsilon^2 x^2} \right|,$$

Denoting $\mathcal{A} = \max\{|\alpha_j|, j = 1, ..., N\}$ and using the upper bound for the n-th Hermite polynomial (6) we obtain,

$$|s(x) - s_M^{\gamma}(x)| \leq \mathcal{A} \sum_{k=1}^{N} \sum_{n=M}^{\infty} \left(\exp\left(\varepsilon^2 (x_k^{\text{cen}})^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) \frac{\varepsilon^{2n}}{\gamma^n n!} |x_k^{\text{cen}}|^n \cdot \right. \\ \left. \cdot \mathrm{e}^{\left(\frac{\gamma^2}{2} - \varepsilon^2\right) x^2} c 2^{\frac{n}{2}} \sqrt{n!} \right). \tag{30}$$

To further estimate this expression, we use that $|x_k^{\text{cen}}| \leq L$, $k = 1, \ldots, N$, and $|x| \leq L$ and introduce the constants,

(31)
$$P_1 = \max\left\{\exp\left(\left(\frac{\gamma^2}{2} - \varepsilon^2\right) L^2\right), 1\right\}, \quad P_2 = \max\left\{\exp\left(\varepsilon^2 L^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right), 1\right\}.$$

Then, we obtain the bound

$$|s(x) - s_M^{\gamma}(x)| \leq \underbrace{c\mathcal{A} N P_1 P_2}_{C} \underbrace{\sum_{n=M}^{\infty} \frac{(\sqrt{2}\varepsilon^2 L)^n}{\gamma^n \sqrt{n!}}}_{T_M} \tag{32}$$

Consider the following series of positive terms,

$$\sum_{n=1}^{\infty} \underbrace{\frac{(\sqrt{2}\varepsilon^2 L)^n}{\gamma^n \sqrt{n!}}}_{t_n}. \tag{33}$$

Then T_M is the tail of the series. Therefore it is enough to prove that the series (33) converges in order to prove that $T_M \to 0$ [2, 6.11]. It can be shown that $\lim_{n\to\infty} t_{n+1}/t_n = 0$ and hence, the series $\{t_n\}$ converges by the ratio criterion [2, 6.17]. Therefore,

$$|s(x) - s_M^{\gamma}(x)| \leq C T_M \to 0 \quad \text{for} \quad M \to \infty, \tag{34}$$

where C depends only on the size of the interpolation interval L, the coefficients $\{\alpha_k\}$ of the RBF interpolant, the number of RBFs N, and the parameters ε, γ.

For $\gamma > \sqrt{2}\varepsilon^2 L$, T_M can be estimated by

$$T_M < \frac{1}{\sqrt{M!}} \sum_{n=M}^{\infty} q^n$$

with $q = \frac{\sqrt{2}\varepsilon^2 L}{\gamma} < 1$. Using the geometric series we obtain (28). ∎

REMARK 2.1. *Analogously, it can be proven that the HermiteGF interpolant $s_M^{\gamma}(x)$ converges to $s(x)$ in $L_2([-L, L])$. Moreover, the geometric bound (28) holds with a different constant for the $L_2([-L, L])$ norm.*

3. Stabilization of the RBF interpolation. In this section, we derive a numerical stabilization algorithm of the RBF interpolation based on the HermiteGF expansion. The main idea is to perform a basis transformation to a more stable basis $\{H_n^{\gamma,\varepsilon}\}$. For appropriately chosen parameter γ we expect the basis $\{H_n^{\gamma,\varepsilon}\}$ to be better conditioned. We can write the expansion (8) as an infinite matrix-vector product,

$$\underbrace{\begin{pmatrix}\phi_1(x), & \dots, & \phi_N(x)\end{pmatrix}}_{\Phi} = \underbrace{\begin{pmatrix}H_1^{\gamma,\varepsilon}(x), & \dots, & H_M^{\gamma,\varepsilon}(x), & \dots\end{pmatrix}}_{H^{\gamma,\varepsilon}} B(\varepsilon,\gamma,X^{\text{cen}}) \tag{35}$$

with

$$B(\varepsilon,\gamma,X^{\text{cen}})_{nk} = \exp\left(\varepsilon^2(x_k^{\text{cen}})^2\left(\frac{\varepsilon^2}{\gamma^2}-1\right)\right)\frac{\varepsilon^{2n}\sqrt{2^n}}{\sqrt{n!}}(x_k^{\text{cen}})^n. \tag{36}$$

The major part of the ill-conditioning is now confined in the matrix B. Since B is independent of the point x where the basis function is evaluated, both the evaluation and interpolation matrix can be expressed in the form (35) with the same matrix B. For this reason, a strategy of dealing with the ill-conditioning in B analytically can be developed.

To make the representation (35) usable for numerical computations, one has to cut the expansion (8) at some point M. This point has to be chosen such that the order of magnitude of the interpolation error is the same order as the error of the RBF interpolant.

We now consider two ways of dealing with the matrix B. One way is to eliminate the matrix B from the computation completely by choosing $M = N$. This case corresponds to an interpolation in the HermiteGF basis. Even though this method provides good results, it lacks the flexibility of choosing M. To allow $M > N$, an RBF-QR algorithm can be designed for the HermiteGF expansion analogously to the Chebyshev RBF-QR algorithm by Fornberg et al. [7].

3.1. HermiteGF interpolant. Let us write the RBF interpolant $s(x)$ in the matrix-vector form,

$$s(x) = \sum_{k=1}^{N}\alpha_k\phi_k(x) = \Phi(x,X^{\text{cen}})\alpha, \tag{37}$$

where $\Phi(x,X^{\text{cen}}) = \begin{pmatrix}\phi_1(x), & \dots, & \phi_N(x)\end{pmatrix}$, X^{cen} are the centering points of the basis functions and α is the coefficients vector. We now use the expansion (8):

$$s(x) = \Phi(x,X^{\text{cen}})\alpha \approx H^{\gamma,\varepsilon}(x)B(\varepsilon,\gamma,X^{\text{cen}})\alpha, \tag{38}$$

where the ill-conditioning related to varying powers of ε is confined in a matrix B.

The system (3) then takes the form,

$$f_0(X^{\text{col}}) = H^{\gamma,\varepsilon}(X^{\text{col}})B(\varepsilon,\gamma,X^{\text{cen}})\alpha, \tag{39}$$

where X^{col} are the collocation points. Considering $M = N$ we arrive to the following expression for the coefficients α,

$$\alpha = B(\varepsilon,\gamma,X^{\text{cen}})^{-1}H^{\gamma,\varepsilon}(X^{\text{col}})^{-1}f_0(X^{\text{col}}). \tag{40}$$

If we now insert the expression (40) into (38), we get,

$$s(x) \approx s_M^{\gamma} = H^{\gamma,\varepsilon}(x)B(\varepsilon,\gamma,X^{\text{cen}})B(\varepsilon,\gamma,X^{\text{cen}})^{-1}H^{\gamma,\varepsilon}(X^{\text{col}})^{-1}f_0(X^{\text{col}}) \tag{41}$$

$$= H^{\gamma,\varepsilon}(x)H^{\gamma,\varepsilon}(X^{\text{col}})^{-1}f_0(X^{\text{col}}). \tag{42}$$

The only restriction that we put on the collocation points is that their number should be equal to the number of center points. Note that the obtained expression for the interpolant s *does not depend* on the *grid of centers* X^{cen}. This way of computing s is very easy to implement and allows to avoid ill-conditioning arising in B. However, it restricts us to $M = N$.

3.2. RBF-QR. In case we want to cut the expansion (8) at $M > N$, the interpolation algorithm gets more complicated. Since the matrix B is now rectangular, B^{-1} is not well defined. Therefore, it is necessary to come up with another way of dealing with the ill-conditioning contained in B. We follow the RBF-QR approach and further split B into a well-conditioned full matrix C and a diagonal matrix D, where all harmful effects are confined in D. In the case of expansion (8), the following setup follows naturally from the Chebyshev-QR theory [7, § 4.1.3],

$$C_{kn} = \exp\left(\varepsilon^2 (x_k^{\text{cen}})^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) (x_k^{\text{cen}})^n, \quad D_{nn} = \frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n \sqrt{n!}}.$$

A problem is arising when we take *center points* with an absolute value greater than 1. That can lead to an ill-conditioning in C. One of the ways to treat this effect is to divide each coefficient by the width of the domain L containing the centering points. That might be dangerous when the domain is too large, however, it still extends the range of available domains. The coefficients then look as follows,

$$C_{kn} = \exp\left(\varepsilon^2 (x_k^{\text{cen}})^2 \left(\frac{\varepsilon^2}{\gamma^2} - 1\right)\right) \frac{(x_k^{\text{cen}})^n}{L^n}, \quad D_{nn} = \frac{\varepsilon^{2n}\sqrt{2^n}}{\gamma^n \sqrt{n!}} L^n.$$

Another possible source of ill-conditioning in C is the exponential if $\gamma < \varepsilon$, however, this is usually not the case.

The goal is to find a basis $\{\psi_j\}$ spanning the same space as $\{\phi_k\}$ but yielding a better conditioned collocation matrix. In particular, we need an invertible matrix X such that $X^{-1}\Phi^T$ is better conditioned. Let us perform a QR-decomposition on $C = QR$. Then, we get,

$$\Phi(x)^T = CDH^{\gamma,\varepsilon}(x)^T = Q \begin{pmatrix} R_1 & R_2 \end{pmatrix} \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} H^{\gamma,\varepsilon}(x)^T. \tag{43}$$

Consider $X = QR_1D_1$. The new basis $\Psi := X^{-1}\Phi(x)^T$ can be formed as,

$$\Psi(x)^T = D_1^{-1}R_1^{-1}Q^{\text{H}}\Phi(x)^T = D_1^{-1}R_1^{-1}Q^{\text{H}}Q \begin{pmatrix} R_1D_1 & R_2D_2 \end{pmatrix} H^{\gamma,\varepsilon}(x)^T \tag{44}$$

$$= \begin{pmatrix} \text{Id} & D_1^{-1}R_1^{-1}R_2D_2 \end{pmatrix} H^{\gamma,\varepsilon}(x)^T. \tag{45}$$

To avoid under/overflow in the computation of $D_1^{-1}R_1^{-1}R_2D_2$, we form the two matrices $\tilde{R} = R_1^{-1}R_2$ and $\tilde{D} \in \mathbb{R}^{N\times M-N}$ with elements

$$\tilde{d}_{i,j} = \gamma^{j_1-j_2}\varepsilon^{2(j_2-j_1)}L^{j_2-j_1}\sqrt{\frac{j_1!}{j_2!}}\sqrt{2^{j_2-j_1}}. \tag{46}$$

and compute their Hadamard product. That is why despite the harmful effects contained in D, the term $D_1^{-1}R_1^{-1}R_2D_2$ does not lead to ill-conditioning.

3.3. Truncation value M. The major question arising for RBF-QR methods is the truncation value M. For $M = N$ we have a cheap and straightforward way of stably computing the interpolant without doing a costly QR-decomposition. Moreover, this ansatz allows for a tensor approach (cf. subsection 4.1) where forming full matricies for high dimensions can be avoided which is of great computational advantage.

Using $M \leq N$, the relation (43) becomes rank-deficient, since $\operatorname{rank}(CD) < \min(M, N) = M$. Such a low-rank approximation was tested by Fasshauer and McCourt [5, § 6.1] and showed rather good results. However, it still requires the assembly of a global matrix, which could be rather expensive in higher dimensions. Adding more expansion functions to reach $M = N$ significantly simplifies the structure of the method and does not harm the quality of the solution. That is why we will not be focusing on the rank-deficient case.

As for the case $M \geq N$ the values of coefficients D decay so rapidly that for a large enough value N the additional terms of the expansion (8) are negligible. The error is dominated by the error coming from the underlying RBF interpolation. This has also been confirmed numerically for various examples. In Table 1, we provide the results obtained with $\varepsilon = 0.1$ for one of the test functions from [18],

$$f_2(x) = \sin\left(\frac{x}{2}\right) - 2\cos(x) + 4\sin(\pi x), \quad x \in [-4, 4]. \tag{47}$$

Table 1: The L_2 interpolation error on the Chebyshev grid for the function f_2 with N basis functions, $M = N + j_{\text{add}}$ expansion functions, and 100 equidistant evaluation points.

j_{add} \ N_{bf}	10	20	25	30
0	8.6629010	0.0029523	0.1937075×10^{-4}	0.1827378×10^{-8}
1	8.6629010	0.0029523	0.1944307×10^{-4}	0.1827378×10^{-8}
2	8.6648555	0.0029609	0.1944307×10^{-4}	0.1836897×10^{-8}
3	8.6648555	0.0029609	0.1944291×10^{-4}	0.1836897×10^{-8}
4	8.6648569	0.0029609	0.1944291×10^{-4}	0.1836864×10^{-8}
5	8.6648569	0.0029609	0.1944291×10^{-4}	0.1836864×10^{-8}
30	8.6648569	0.0029609	0.1944291×10^{-4}	0.1836865×10^{-8}

4. Multivariate interpolation. In this section, we address the question of how to apply our stabilization algorithm to multivariate interpolation problems. First of all, we notice that the Gaussian basis is separable, i.e. the multivariate Gaussian basis $\psi_k(\mathbf{x})$ (with $\mathbf{x} \in \mathbb{R}^d$) can be written as a product of one dimensional Gaussians,

$$\psi_k(\mathbf{x}) = \exp\left(-\varepsilon^2 \|\mathbf{x} - \mathbf{x}_k^{\text{cen}}\|^2\right) = \prod_{i=1}^{d} \phi_k(x_i). \tag{48}$$

One possibility is to derive an RBF-QR algorithm that truncates the multivariate expansion on a hyperbolic cross. If we use a tensor product grid of centering and collocation points, on the other hand, a very simple generalization of the stabilization algorithm can be designed by applying the HermiteGF expansion separately in each dimension. This ansatz yields a memory-sparse algorithm since it relies on Kronecker products of one dimensional matrices as we will derive in Section 4.1. Therefore,

it is particularly suitable for high-dimensional problems, even though it comes with the drawback that we loose the uniformity in all directions. Hagedorn generating functions [3, 13] provide a truly multi-dimensional generalization of the HermiteGF expansion that additionally allows for anisotropic RBFs. This will be discussed in Section 4.2.

4.1. Tensor product approach. For dimension d, let X_ℓ^{cen}, $\ell = 1, \dots, d$, be the centering points along each coordinate direction. Then, we can index the d variate basis by a multi-index $\mathbf{k} = (k_1, \dots, k_d)$ and write the multivariate interpolant $s(\mathbf{x})$ as

$$
\begin{aligned}
s(\mathbf{x}) = & \sum_{k_1=1}^{N_1} \dots \sum_{k_d=1}^{N_d} \alpha_{\mathbf{k}} \phi_{\mathbf{k}}(\mathbf{x}) = \sum_{k_1=1}^{N_1} \dots \sum_{k_d=1}^{N_d} \alpha_{\mathbf{k}} \prod_{\ell=1}^{d} \phi_{k_\ell}(x_\ell) \\
= & \quad (\Phi(x_d, X_d^{\text{cen}}) \otimes \dots \otimes \Phi(x_1, X_1^{\text{cen}}))\, \text{vec}(\alpha),
\end{aligned} \tag{49}
$$

where we denote by $\text{vec}(\alpha)$ the vectorization of the coefficient tensor α. Now, we can replace $\Phi(x_\ell, X_\ell^{\text{cen}})$ by $H^{\gamma,\varepsilon}(x_\ell)B(\varepsilon, X_\ell^{\text{cen}})$ transforming the individual one-dimensional Gaussian bases to the HermiteGF basis with $M_\ell = N_\ell$ expansion coefficients. This yields the following expression for the interpolant,

$$
s_M^\gamma(\mathbf{x}) = (H^{\gamma,\varepsilon}(x_d)B(\varepsilon, X_d^{\text{cen}}) \otimes \dots \otimes H^{\gamma,\varepsilon}(x_1)B(\varepsilon, X_1^{\text{cen}}))\, \text{vec}(\alpha). \tag{50}
$$

Introducing a second tensor product grid for the collocation points X_ℓ^{col}, $\ell = 1, \dots, d$, we analogously get a Kronecker product representation of the collocation matrix yielding the following expression for the expansion coefficients α,

$$
\text{vec}(\alpha) = \left(H^{\gamma,\varepsilon}(X_d^{\text{col}})B(\varepsilon, X_d^{\text{cen}}) \otimes \dots \otimes H^{\gamma,\varepsilon}(X_1^{\text{col}})B(\varepsilon, X_1^{\text{cen}})\right)^{-1} \text{vec}(f_0(X_1^{\text{col}}, \dots, X_d^{\text{col}})). \tag{51}
$$

Putting everything together, we get

$$
\begin{aligned}
s_M^\gamma(\mathbf{x}) &= (H^{\gamma,\varepsilon}(x_d)B(\varepsilon, X_d^{\text{cen}}) \otimes \dots \otimes H^{\gamma,\varepsilon}(x_1)B(\varepsilon, X_1^{\text{cen}})) && (52) \\
&\left(H^{\gamma,\varepsilon}(X_d^{\text{col}})B(\varepsilon, X_d^{\text{cen}}) \otimes \dots \otimes H^{\gamma,\varepsilon}(X_1^{\text{col}})B(\varepsilon, X_1^{\text{cen}})\right)^{-1} \text{vec}(f_0(X_1^{\text{col}}, \dots, X_d^{\text{col}}) && (53) \\
&= \left(H^{\gamma,\varepsilon}(x_d)H^{\gamma,\varepsilon}(X_d^{\text{col}})^{-1} \otimes \dots \otimes H^{\gamma,\varepsilon}(x_1)H^{\gamma,\varepsilon}(X_1^{\text{col}})^{-1}\right) \text{vec}(f_0(X_1^{\text{col}}, \dots, X_d^{\text{col}})). && (54)
\end{aligned}
$$

Hence, we can compute the matrices $H^{\gamma,\varepsilon}(x_\ell)H^{\gamma,\varepsilon}(X_\ell^{\text{col}})^{-1}$ separately for each dimension $\ell = 1, \dots, d$, and then apply them mode-wise to the tensor $f_0(X_1^{\text{col}}, \dots, X_d^{\text{col}})$ of function values. The memory requirements for the interpolation matrices is hence limited to dN^2 which is much smaller than the memory requirement for the full d dimensional interpolation matrix of N^{2d}.

4.2. Anisotropic approximation. Until now we considered only interpolations with the same shape parameter ε in both directions. Given the HermiteGF-tensor structure one could also easily use different values of ε in different directions. Finding a stable interpolant for anisotropic multidimensional RBFs of type $\exp(-(x-x_k)^T E(x-x_k))$ is a more challenging task. A similar question was raised in [5, § 8.5], however, without further investigation. It turns out that generating function theory provides a convenient toolbox for deriving a stable basis that spans the same space, but doesn't lead to ill-conditioning related to small elements in E. Adapting the result of [3, Lemma 5] we derive the HagedornGF expansion that is very similar to the HermiteGF expansion.

LEMMA 4.1. *HagedornGF expansion*
For all positive definite $E \in \mathbb{R}^{d\times d}$, $\mathbf{x}_k \in \mathbb{R}^d$ the following relation holds,

(55)
$$\exp(-(\mathbf{x}-\mathbf{x}_k)^T E(\mathbf{x}-\mathbf{x}_k)) = \exp(-\mathbf{x}_k^T E\mathbf{x}_k + \mathbf{x}_k^T E^T E\mathbf{x}_k) \sum_{\boldsymbol{\ell}\in\mathbb{N}^d} \frac{(E\mathbf{x}_k)^{\boldsymbol{\ell}}}{\boldsymbol{\ell}!} h_{\boldsymbol{\ell}}(\mathbf{x}) \exp(-\mathbf{x}^T E\mathbf{x}),$$

where x_k is the center of the function $\phi_{\boldsymbol{k}}$, E is a shape matrix and $h_{\boldsymbol{\ell}}(\mathbf{x})$ are tensor product of physicists' Hermite polynomials,

$$h_{\boldsymbol{\ell}}(\mathbf{x}) = h_{\ell_1}(x_1) \cdot \ldots \cdot h_{\ell_d}(x_d). \tag{56}$$

Proof. The general Hagedorn polynomial's generating function is given by [3, Lemma 5], [13, Theorem 3.1],

$$\sum_{\boldsymbol{\ell}\in\mathbb{N}^d} \frac{t^{\boldsymbol{\ell}}}{\boldsymbol{\ell}!} q_{\boldsymbol{\ell}}(\mathbf{x}) = \exp(2\mathbf{x}^T\mathbf{t} - \mathbf{t}^T M\mathbf{t}), \tag{57}$$

where $q_{\boldsymbol{\ell}}^M(\mathbf{x})$ are generalized Hagedorn polynomials [3, § 3] that are given for any symmetric unitary matrix $M \in \mathbb{C}^{d\times d}$ by the following three-term recurrence,

$$(q_{\boldsymbol{\ell}+e_j}^M(\mathbf{x}))_{j=1}^d = 2\mathbf{x} q_{\boldsymbol{\ell}}^M(\mathbf{x}) - 2M \cdot (\ell_j q_{\boldsymbol{\ell}-e_j}^M(\mathbf{x}))_{j=1}^d, \tag{58}$$

with boundary conditions $q_0^M = 1$, $q_\ell^M = 0$ for all $\ell \notin \mathbb{N}^d$.

Consider $M = \mathrm{Id}$, $t = E\mathbf{x}_k$, then

$$\sum_{\boldsymbol{\ell}\in\mathbb{N}^d} \frac{(E\mathbf{x}_k)^{\boldsymbol{\ell}}}{\boldsymbol{\ell}!} q_{\boldsymbol{\ell}}(x) = \exp(2\mathbf{x}^T E\mathbf{x}_k - \mathbf{x}_k^T E^T E\mathbf{x}_k). \tag{59}$$

Note that for the case of $M = \mathrm{Id}$, Hagedorn polynomials turn into a tensor product of Hermite polynomials,

$$q_{\boldsymbol{\ell}}^{\mathrm{Id}}(\mathbf{x}) = h_{\ell_1}(x_1) \cdot \ldots \cdot h_{\ell_d}(x_d) = h_\ell(\mathbf{x}). \tag{60}$$

Hence, we get,

$$\exp(-(\mathbf{x}-\mathbf{x}_k)^T E(\mathbf{x}-\mathbf{x}_k)) = \exp(-\mathbf{x}^T E\mathbf{x} + 2\mathbf{x}^T E\mathbf{x}_k - \mathbf{x}_k^T E\mathbf{x}_k) \tag{61}$$

$$= \exp(-\mathbf{x}_k^T E\mathbf{x}_k) \cdot \exp(\mathbf{x}_k^T E^T E\mathbf{x}_k) \cdot \exp(2\mathbf{x}^T E\mathbf{x}_k - \mathbf{x}_k^T E^T E\mathbf{x}_k) \cdot \exp(-\mathbf{x}^T E\mathbf{x}) \tag{62}$$

$$= \exp(-\mathbf{x}_k^T E\mathbf{x}_k + \mathbf{x}_k^T E^T E\mathbf{x}_k) \sum_{\boldsymbol{\ell}\in\mathbb{N}^d} \frac{(E\mathbf{x}_k)^{\boldsymbol{\ell}}}{\boldsymbol{\ell}!} h_\ell(\mathbf{x}) \exp(-\mathbf{x}^T E\mathbf{x}). \tag{63}$$

∎

An RBF-QR method can then be naturally derived based on the HagedornGF expansion. This expansion provides a new powerful tool of dealing with anisotropic approximation. However, the computational costs of that method are way higher than for the HermiteGF-tensor approach.

Note that HermiteGF-tensor interpolation considered before corresponds to the following matrix E,

$$E_{\mathrm{tensor}} = \begin{pmatrix} \varepsilon^2 & 0 \\ 0 & \varepsilon^2 \end{pmatrix}. \tag{64}$$

5. Numerical results. In this section, we first discuss the implementation of the new method. Then, we compare the HermiteGF-based algorithm with the existing stabilization methods. We also look closer into the role of the parameter γ in conditioning and discuss the scaling of the method with dimensions. For all 1D tests we look at the L_2 error of the interpolant evaluated at 100 uniformly distributed points. For the multidimensional case less evaluation points have been used and will be specified separately below.

5.1. Stable implementation. We have implemented the HermiteGF interpolation both in `MATLAB` and `Julia`. The code can be downloaded from https://gitlab.mpcdf.mpg.de/clapp/hermiteGF. The `MATLAB` implementation has shown more stable results in some cases, on the other hand, `Julia` yields better performance (cf. section 6), especially in high dimensions where `Julia` enables easy and efficient parallelization.

Even though the described approach allows to reduce the ill-conditioning of the collocation and evaluation matrices, the HermiteGF-based matrices still become increasingly ill-conditioned for growing number of basis functions. On the other hand, the product of the evaluation matrix $H^{\gamma,\varepsilon}(X^{\text{eval}})$ and the inverse of the collocation matrix $H^{\gamma,\varepsilon}(X^{\text{col}})$ is still well-conditioned. For this reason, it is crucial to take special care when building these matrices and inverting the collocation matrix. The following configurations have proven to be preferable:

- For all the dimensions $\ell = 1, \dots, d$ compute $H^{\gamma,\varepsilon}(X_\ell^{\text{eval}})H^{\gamma,\varepsilon}(X_\ell^{\text{col}})^{-1}$ first, which allows to cancel out the ill-conditioning.
  Using the built-in operator `/` for the inversion yields good results both in `MATLAB` and `Julia`. However, `MATLAB` proved superior in the severely ill-conditioned case.
- The HermiteGF basis functions can be stabely evaluated by formulating them in terms of the Hermite functions ψ_n,

$$H_n^{\gamma,\varepsilon}(x) = \pi^{1/4}\psi_n(\gamma x)\exp(-\varepsilon^2 x^2 + (\gamma x)^2/2). \tag{65}$$

  Hermite functions can be stabely evaluated based on their three-term recurrence.
  `MATLAB` offers the built-in function `hermiteH` for the evaluation of Hermite polynomials. An evaluation based on this function yields somewhat better results, however, the evaluation is also considerably more costly.

All experiments were performed with `MATLAB` if not stated otherwise.

5.2. Comparison with existing RBF-QR methods. In this section, we compare the performance of the above described method with the Chebyshev-QR method[1] and the Gauss-QR method[2]. We use the two test functions that were studied in [18], namely

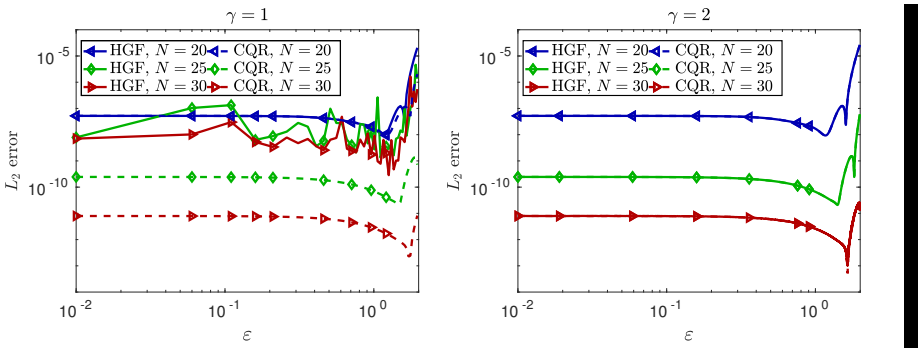
$$f_1(x) = e^x \sin(2\pi x) + \frac{1}{x^2+1}, \quad x \in [-1, 1], \tag{66}$$

$$f_2(x) = \sin\left(\frac{x}{2}\right) - 2\cos(x) + 4\sin(\pi x), \quad x \in [-4, 4]. \tag{67}$$

Note that for the Chebyshev-QR method we must always scale the interpolated func-

[1] Code downloaded from http://www.it.uu.se/research/scientific_computing/software/rbf_qr on November 28, 2016.

[2] Code downloaded from http://math.iit.edu/~mccomic/gaussqr/ on May 29, 2017.

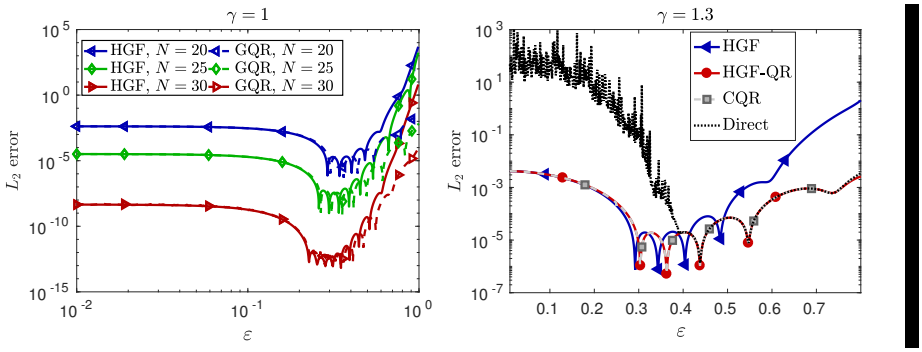


(a) $\gamma = 1$, $\varepsilon =$ `0.01:0.05:1.99` (b) $\gamma = 2$, $\varepsilon =$ `0.01:0.001:1.99`

Fig. 1: For the function f_1 HermiteGF-tensor algorithm (HGF) tends to be unstable for $\gamma = 1$ the for larger number of Chebyshev nodes unlike the Chebyshev-QR (CQR). However the error magnitude is still reasonable. Increasing the value of γ to 2 stabilizes the method and brings the interpolation quality in agreement with other methods.

tion to the unit disk. This also implies a scaling of the value of the shape parameter which we account for in Figure 2. We use Chebyshev collocation points here but discuss the case of uniform points in subsection 5.4

We look at the performance of the methods for different values of ε. The Chebyshev-QR error curves turned out to lay exactly on top of the Gauss-QR ones with the optimal values of α from [18, § 5.1], that is why we only present one of them at a time. For all setups in the flat limit our HermiteGF-tensor method performs in a stable way



(a) $\varepsilon =$ `0.01:0.001:0.99` (b) $\varepsilon =$ `0.01:0.001:0.8`, $N = 20$. For HGF-QR $M = 30$.

Fig. 2: Dependence of the L2 error for the function f_2 on the value of ε with different number of Chebyshev nodes shown for the HermiteGF-tensor (HGF), the Chebyshev-QR (CQR), and the Gauss-QR (GQR) method. With the natural choice of $\gamma = 1$ the HermiteGF-tensor method is in a good agreement with the RBF-QR methods.

.

unlike RBF-Direct. However, function f_1 is more sensitive to the parameter γ. For a natural choice of $\gamma = 1$ the algorithm gets unstable for increasing number of basis functions. However, the magnitude of the error still stays around 10^{-8} (see Figure 1a). If we increase the value of γ to 2, we get a full resemblance to the Chebyshev-QR results (see Figure 1b).

For the function f_2 with $\gamma = 1$, i.e. $\gamma L = 4$, HermiteGF-tensor and Gauss-QR show comparable results (see Figure 2): For small values of ε the results are identical but they start to differ slightly for the optimal ε range before clearly diverging when the error starts to grow. The curve for $N = 20$ where this effect is most pronounced is further investigated in Figure 2a. For larger ε the RBF-Direct method produces stable results that are in agreement with the Chebyshev-QR method. In the figure, we also show the results of the HermiteGF-QR method with an expansion of $M = 30$ points and $\gamma = 1.3$, again agreeing with Chebyshev-QR. From these experiments, we conclude that $M > N$ can be necessary in the optimal ε range (especially for small N) to exactly reproduce the Gaussian RBF interpolant. On the other hand, the HermiteGF method with $M = N$ gives results of the same quality while being cheaper. For larger values of ε, the method seems to be more sensitive to the parameter choice. However, in this range the RBF-Direct algorithm would anyway be preferable. Further information regarding the choice of γ can be found in subsection 5.3.

5.3. Scaling and conditioning. Let us take a look at the behavior of the condition number of the interpolation matrix for different values of γ. We consider an interpolation matrix on an interval $[-1, 1]$ as a function of the number N_{col} of Chebyshev points. Note that the interpolation matrix that has to be inverted is independent of the interpolated function.

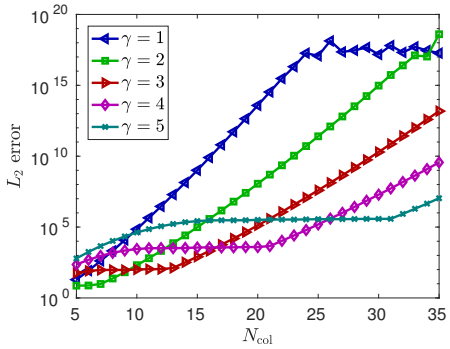


Fig. 3: Conditioning of the interpolation matrix on the interval $[-1, 1]$ for varying number of Chebyshev points. The condition number grows slower for larger values of the parameter γ.

As we can see in Figure 3 the condition number gets smaller for larger values of γ. There are two counteracting effects influencing conditioning. On the one hand, the larger the value of γ the larger the evaluation interval for the Hermite polynomials becomes. Therefore, for larger value of γ the points are further away from each other for the same values of N_{col}, which leads to improved condition numbers. On the other hand, Hermite polynomials take very large values on big domains, which can lead to an overflow.

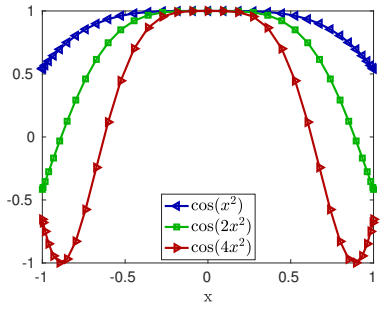


Fig. 4: Test functions with different gradients.

Recall that for the function f_2 from the previous section $\gamma = 1$ provided good results. However, in that case the interpolation interval was $[-4, 4]$. Therefore, the evaluation interval for Hermite polynomials is also $[-4, 4]$ which corresponds to smaller condition number (equivalent to $\gamma = 4$ in Figure 3). Since for the function f_1, the interpolation interval was $[-1, 1]$, increasing $\gamma = 2$—and hence the evaluation interval to $[-2, 2]$—reduced the condition number and allowed for stable computations for higher values of N_{col}.

As mentioned before, the conditioning of the interpolation matrix does not depend on the interpolated function itself. However, the impact on the result can be different for different functions. Consider the following functions on the interval $[-1, 1]$ (see Figure 4):

$$f_1^{\text{c}} = \cos(x^2), \quad f_2^{\text{c}} = \cos(2x^2), \quad f_4^{\text{c}} = \cos(4x^2). \tag{68}$$

We expect that the faster the function change, especially near the boundaries, the more sensitive the interpolation quality should be towards the condition number. Indeed, looking at the L_2-error (see Figure 5) we see that for f_1^{c} the quality is good for all integer values of γ between 1 and 5. On the other hand, for the function f_4^{c} the result for $\gamma = 1$ is considerably worse than for other values. Note that the values of γ are not fixed to integers but any $\gamma > 0$ can be chosen. On the other hand, the stability is not sensitive to minor changes of γ which is why we use a rough integer estimation of the desired evaluation interval.

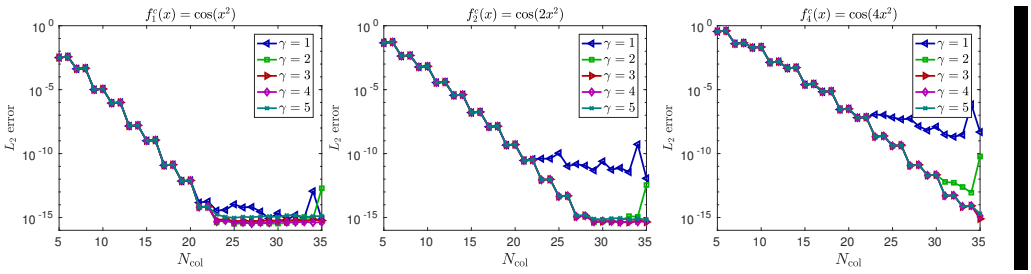


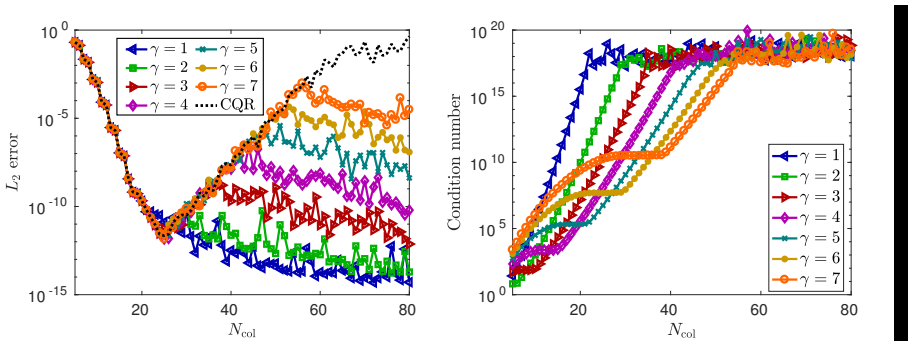
Fig. 5: The L_2 error for the testing functions for different values of γ. For the more flat function f_1^{c} all values of γ suit equally good. For the other two functions it is preferable to use $\gamma \geq 3$.

Even though using large γ appears to be advantageous one should not forget that Hermite polynomials take very large values on big domains. That can lead to cancellations and overflow. From our experience, the range $[3, 5]$ seems to be optimal for γL for most of the cases.

5.4. Interpolation on a uniform grid. Spectral interpolation on uniform grids is known to be intrinsically ill-conditioned causing large errors close to the boundary [17]. This ill-conditioning persists after a basis transformation so that the number of basis functions N_{col} needs to be chosen small enough in applications where uniform nodes are of interest. We consider the following function for our tests,

$$f_{\text{u}}(x) = \sin(2x) + \cos(4x) + \frac{1}{2+x}, \quad x \in [-1, 1]. \tag{69}$$

Figure 6a shows the L_2 error in the interpolation of function f_{u} for $\varepsilon = 0.1$ as a function of the number of collocation points. The Chebyshev-QR algorithm and the HermiteGF algorithm for various values of γ are considered. First, we note that we again need to choose γ large enough to get results of the same quality as with the Chebyshev-QR algorithm. The results of the Chebyshev-QR algorithm also clearly show the increase of the error that is typical for uniform points (starting at $N_{\text{col}} = 24$). For the HermiteGF method, the error starts to decrease again as soon as the condition number of the interpolation matrix stagnates at a very large value (cf. Figure 6b).



(a) L_2 error of HermiteGF (γ given in the legend) and Chebyshev-QR methods (CQR).

(b) HermiteGF, conditioning.

Fig. 6: Error of in interpolation of the function f_{u} on a uniform grid.

5.5. Multivariate interpolation. In this section we take a look at high dimensional interpolation. As mentioned before, RBF-QR methods require forming a full collocation matrix which leads to very high computational costs in 3+ dimensions. Consider the function,

$$f_3(x) = \cos(\|x\|^2) \tag{70}$$

We now look at the behavior of HermiteGF-tensor for different dimensions. With the use of simple parallelization via built-in `Julia` tools, it was possible to run tests for 1–5D. The largest simulation run contained $1.5 \cdot 10^6$ points. Due to the computational

complexity for 5D only 5–18 points per dimension have been considered. One can see in Figure 7 that even though the error increases for larger values of N_{col} with the dimension, the rate of decay of the error is the same for all dimensions.

Note that the underlying RBF expansion used in Chebyshev-QR could be used in a similar fashion to construct a tensor based algorithm. However, the restriction to the unit domains still holds.

In order to demonstrate the potential for the HagedornGF expansion for anisotropic basis functions, we consider the following function,

$$f_{\text{a}}(x,y) = \cos\left(\frac{(x+y)^2}{2.88} + \frac{(y-x)^2}{4.5}\right), \quad x,y \in [-1,1] \tag{71}$$

This is an anisotropic modification of a two-dimensional function f_3 used for the tests earlier. We expect that anisotropic interpolation should suit better in this case than a regular HermiteGF-tensor. For testing purposes only matrices E of the following form were considered,

$$E = \begin{pmatrix} \varepsilon^2 & \xi^2 \\ \xi^2 & \varepsilon^2 \end{pmatrix}, \quad \xi < \varepsilon < 1. \tag{72}$$

Indeed, as one can see in Figure 8 there exists a matrix E for which the error is smaller than for the HermiteGF interpolants with equal values of ε in both directions.

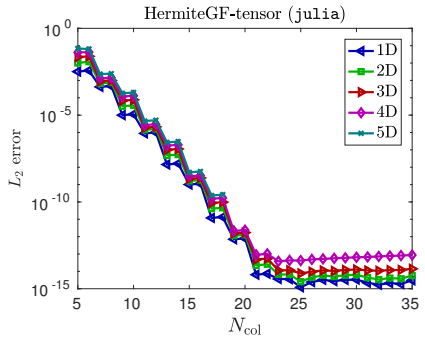


Fig. 7: Dependence of the L_2 error of the HermiteGF-tensor ($\gamma = 3$) interpolation on the number N_{col} of Chebyshev nodes per dimension. The value of ε is set to 0.1. The interpolation quality of the HermiteGF-tensor algorithm is almost dimension independent. The error has been computed on a uniform grid with 53 points per dimension.

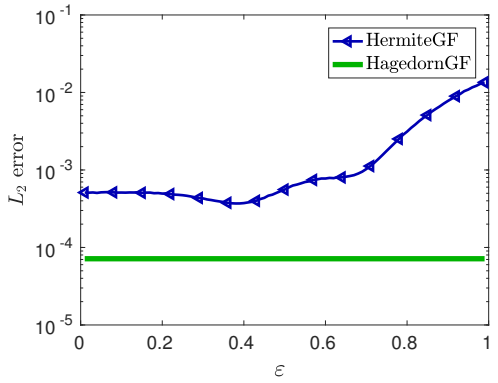


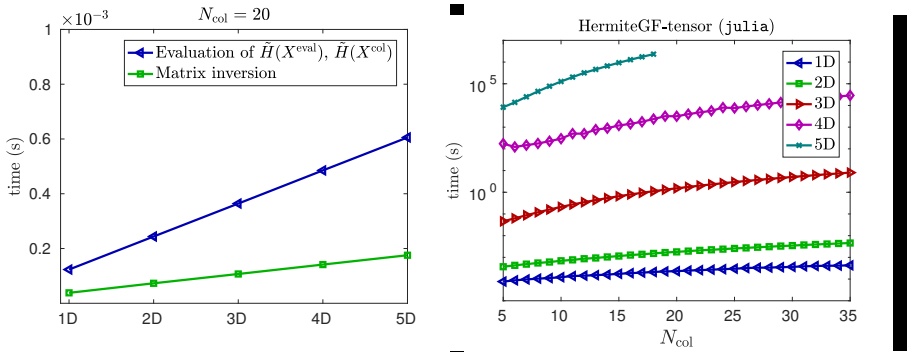
Fig. 8: Anisotropic interpolation of the function f_{A} for the positive-definite E of type $E = \begin{pmatrix} \varepsilon^2 & \xi^2 \\ \xi^2 & \varepsilon^2 \end{pmatrix}$ with $\varepsilon = 0.4$, $\xi = 0.2$ (121 Halton nodes in 2D). The result is better than all HermiteGF interpolants with the same values of the shape parameter in both directions. The error has been evaluated at $289 (= 17^2)$ uniformly distributed evaluation points.

6. Performance tests. To assess the computational complexity and the performance of our HermiteGF-tensor code, we report here the run times of the `Julia` code. Compared to `MATLAB`, `Julia` is faster for 3–5D, while the performance difference is negligible in 1–2D. The experiments where performed on the DRACO cluster of the Max Planck society. A DRACO node is equipped with `Intel 'Haswell' Xeon E5-2698v3` processors with 32 cores @2.3 GHz and 128 GB of memory. The parameters of the basis functions are set to $\varepsilon = 0.1$ and $\gamma = 3$. The number of evaluation points per dimension was fixed to $N_{\mathrm{eval}} = 53$ in all tests. In a first test, we have split the timings to the following three essential parts of the algorithm:

- Forming of the interpolation matrix $\tilde{H}(X^{\mathrm{col}})$ and evaluation matrix $\tilde{H}(X^{\mathrm{eval}})$;
- Inversion of the interpolation matrix;
- Evaluation of the interpolant s.

The timings for the first two tasks are shown in Figure 9a as a function of the problem dimension for $N_{\mathrm{col}} = 20$ collocation points per dimension. Due to the tensor formulation of the algorithm these first two parts do not impose significant costs. Indeed, we only need to evaluate and invert small one dimensional matrices. The costs grow linearly in the dimensionality, since we have two evaluations and one inversion of one-dimensional matrices per dimension. The evaluation of the interpolant s, on the contrary, gets exponentially more expensive with increase of the dimensionality. That is due to the fact that we need to evaluate our interpolant in every point of the multidimensional tensor grid and the domain size grows exponentially with the dimension if we keep the amount of points per dimension constant. This can be seen from the run times reported in Figure 9b for the total simulation times which show an exponential increase in the problem dimension. Note that the total CPU time reported in Figure 9b steems from serial simulations in 1–3D and from parallel runs on 32 nodes for 4 and 5D. In order to minimize the influence of disturbances, we have run all serial simulations 100 times and report the minimum time. For the parallel runs, the disturbances are negligible.

As for the wall clock time, in 1–3D dimensions with moderately low amount of



(a) Time of the matrices evaluation and inversion with 20 Chebyshev collocation points per dimension.

(b) Total CPU time.

Fig. 9: The timings of the HermiteGF-tensor interpolation for 1–5D.

points (up to 35 per dimension) the interpolation can be run in less than a minute without parallelization (see Figure 9b). For 4–5D, parallelization is required. The largest simulation (18^5 points in 5D) takes slightly less than a day on a full node of the DRACO cluster.

7. Conclusion. In this paper we derived a new stabilization algorithm for the RBF interpolation in the flat limit ($\varepsilon \to 0$). The main idea of "isolating" the ill-conditioning in a special matrix is the same as in [9, 5, 7]. On the other hand, we use a novel expansion of RBFs through Hermite polynomials based on the generating functions theory. Even though a standard RBF-QR approach is possible, we follow the road of choosing the number M of expansion functions to be equal to N. This simplifies the algorithm greatly and enables an efficient implementation for up to millions of points in 5D. Compared to the existing RBF-QR stabilization methods (Chebyshev-QR and Gauss-QR) the 1D HermiteGF-based method features the same accuracy while having a simpler structure. The structure of the HermiteGF method is very similar to Gauss-QR, however, the structure of the parameters ε, γ of basis functions is simpler: ε is the original shape parameter of the RBF basis and γ stands for the size of the evaluation domain of the Hermite polynomials. The interpolation quality is not sensitive to a specific value of γ.

Two ways to generalize the algorithm to the multivariate case were discussed. When tensor grids can be used, the HermiteGF-tensor method provides a very efficient embarassingly parallel solution. A similar solution could be also possible with the underlying RBF expansions of Chebyshev-QR and Gauss-QR algorithms. A combination with compression techniques as e.g. proposed by Zhao [21] will be explored in future work. As for the RBF-QR technique, we make a step forward by providing an opportunity for anisotropic approximations. The next steps in that direction is to develop an algorithm of choosing an optimal shape matrix E and to use fast multipole methods to speed up the computation [20].

The HermiteGF-tensor algorithm has been implemented both in `MATLAB` and `Julia`. The `MATLAB` code showed to be less sensitive to floating point arithmectics with large numbers. The `Julia` implementation, on the other hand, features more

efficient computation. Moreover, the `Julia` built-in parallelization toolbox enabled an implementation of $5D$ interpolation with up to 18 points per dimension. With `Julia` being open source, it is possible to run it on any cluster. HermiteGF-tensor is currently the only available stable implementation of the RBF interpolation in the flat limit with millions of points.

Acknowledgments. The authors would like the thank Caroline Lasser (Technische Universität München) for constant support during the project. Fruitful discussions with Elisabeth Larsson (Uppsala University) are gratefully acknowledged.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, vol. 55, Courier Corporation, 1964.

[2] K. G. Binmore, *Mathematical Analysis: a straightforward approach*, Cambridge University Press, 1982.

[3] H. Dietert, J. Keller, and S. Troppmann, *An invariant class of wave packets for the wigner transform*, Journal of Mathematical Analysis and Applications, 450 (2017), pp. 1317–1332, https://doi.org/10.1016/j.jmaa.2016.12.041.

[4] G. E. Fasshauer, F. J. Hickernell, and H. Woniakowski, *On dimension-independent rates of convergence for function approximation with gaussian kernels*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 247–271, https://doi.org/10.1137/10080138X.

[5] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of Gaussian radial basis function interpolants*, SIAM Journal on Scientific Computing, 34 (2012), pp. A737–A762, https://doi.org/10.1137/110824784.

[6] B. Fornberg and N. Flyer, *Solving pdes with radial basis functions*, Acta Numerica, 24 (2015), p. 215258, https://doi.org/10.1017/S0962492914000130.

[7] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions*, SIAM Journal on Scientific Computing, 33 (2011), pp. 869–892, https://doi.org/10.1137/09076756X.

[8] B. Fornberg, E. Lehto, and C. Powell, *Stable calculation of gaussian-based rbf-fd stencils*, Computers & Mathematics with Applications, 65 (2013), pp. 627–637, https://doi.org/http://dx.doi.org/10.1016/j.camwa.2012.11.006.

[9] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM Journal on Scientific Computing, 30 (2007), pp. 60–80, https://doi.org/10.1137/060671991.

[10] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Computers & Mathematics with Applications, 48 (2004), pp. 853 – 867, https://doi.org/http://dx.doi.org/10.1016/j.camwa.2003.08.010.

[11] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Computers & Mathematics with Applications, 54 (2007), pp. 379–398, https://doi.org/http://dx.doi.org/10.1016/j.camwa.2007.01.028.

[12] G. A. Hagedorn, *Raising and lowering operators for semiclassical wave packets*, Annals of Physics, 269 (1998), pp. 77–104, https://doi.org/https://doi.org/10.1006/aphy.1998.5843.

[13] G. A. Hagedorn, *Generating function and a rodrigues formula for the polynomials in d-dimensional semiclassical wave packets*, Annals of Physics, 362 (2015), pp. 603–608, https://doi.org/10.1016/j.aop.2015.08.030.

[14] E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg, *Stable computation of differentiation matrices and scattered node stencils based on gaussian radial basis functions*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2096–A2119, https://doi.org/10.1137/120899108.

[15] E. Larsson, V. Shcherbakov, and A. Heryudono, *A least squares radial basis function partition of unity method for solving PDEs*, arXiv preprint 1702.07148, (2017).

[16] C. Lubich, *From quantum to classical molecular dynamics: reduced models and numerical analysis*, European Mathematical Society, 2008, https://doi.org/10.4171/067.

[17] R. B. Platte, L. N. Trefethen, and A. B. Kuijlaars, *Impossibility of fast stable approximation of analytic functions from equispaced samples*, SIAM review, 53 (2011), pp. 308–318, https://doi.org/10.1137/090774707.

[18] J. Rashidinia, G. Fasshauer, and M. Khasi, *A stable method for the evaluation of Gaussian radial basis function solutions of interpolation and collocation problems*, Computers &

Mathematics with Applications, 72 (2016), pp. 178–193, https://doi.org/10.1016/j.camwa.2016.04.048.

[19] A. E. Tarwater, *Parameter study of Hardys multiquadric method for scattered data interpolation*, Tech. Report UCRL-53670, Lawrence Livermore National Lab., CA (USA), 1985.

[20] C. D. Yu, W. B. March, B. Xiao, and G. Biros, *Inv-askit: A parallel fast direct solver for kernel matrices*, in 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), May 2016, pp. 161–171, https://doi.org/10.1109/IPDPS.2016.12.

[21] Y. Zhao, *Multilevel sparse grid kernels collocation with radial basis functions for elliptic and parabolic problems*, PhD thesis, University of Leicester, 2016.