Multilevel Sparse Kernel-Based Interpolation

A Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

Fazli Subhan
Department of Mathematics,
University of Leicester,
England, United Kingdom.

July 2011.

Abstract

Radial basis functions (RBFs) have been successfully applied for the last four decades for fitting scattered data in $\mathbb{R}^d$, due to their simple implementation for any d. However, RBF interpolation faces the challenge of keeping a balance between convergence performance and numerical stability. Moreover, to ensure good convergence rates in high dimensions, one has to deal with the difficulty of exponential growth of the degrees of freedom with respect to the dimension d of the interpolation problem. This makes the application of RBFs limited to few thousands of data sites and/or low dimensions in practice.

In this work, we propose a hierarchical multilevel scheme, termed sparse kernel-based interpolation (SKI) algorithm, for the solution of interpolation problem in high dimensions. The new scheme uses direction-wise multilevel decomposition of structured or mildly unstructured interpolation data sites in conjunction with the application of kernel-based interpolants with different scaling in each direction. The new SKI algorithm can be viewed as an extension of the idea of sparse grids/hyperbolic cross to kernel-based functions.

To achieve accelerated convergence, we propose a multilevel version of the SKI algorithm. The SKI and multilevel SKI (MLSKI) algorithms admit good reproduction properties: they are numerically stable and efficient for the reconstruction of large data in $\mathbb{R}^d$, for $d = 2, 3, 4$, with several thousand data. SKI is generally superior over classical RBF methods in terms of complexity, run time, and convergence at least for large data sets. The MLSKI algorithm accelerates the convergence of SKI and has also generally faster convergence than the classical multilevel RBF scheme.

Dedicated to:
my mother, my wife, my children
AND
in the memories of
my father and my brother.

Acknowledgements

In the Name of ALLAH, Most Gracious, Most Merciful. May ALLAH shower His countless blessing and peace upon all His messengers and prophets sent from time to time for the guidance of mankind and, in particular, the last prophet Hazrat Muhammad (peace be upon him (PBUH)), who has always been and will always be the main source of inspirations and guidance in all walks of life.

I would like to thank all the good people who made this thesis, in particular, and my stay in Leicester, in general, a success. The foremost amongst these has been my supervisor, Dr. Emmanuil Georgoulis. It has been an immense privilege to work with someone of the very highest intellectual calibre. I must record my gratitude to Dr. Emmanuil Georgoulis for his support, patience, enthusiasm, inspiration and critical thinking. His insight, precision, and distrust for gratuitous abstraction have enormously influenced my development as a mathematician. His friendly, caring and understanding nature always helped me to get through even at the times I was discouraged. In spite of his many commitments, he has always been generous with his time.

I am grateful to all the honourable staff members of the mathematics department for their help. I wish to thank Prof. Jeremy Levesley, the head of department, for his valuable ideas, suggestions and discussions throughout my studies in Leicester.

Certainly this work would not have been possible without the help of my family. I salute and express my deep appreciation to my caring and loving parents for their training, support, encouragement, prayers and dreams for my success during the whole of my life. My warmest thanks go to my beloved wife Nusrat Subhan and our lovely and fabulous daughters Javeria Subhan, Maria Subhan and Munazza Subhan. During the course of my stay in the UK, I remained completely unable to look after my family, and am indebted to my brilliant wife, who provided me with her continuous patience, encouragement and support by looking after herself and our children in the best possible way.

I am unable to thank my kind father Haji Saudagar and caring brother Faiz-ur-Rahman, this dissertation is dedicated to their memories. May Allah keep their souls in eternal rest and peace in heaven.

I wish to thank all my honourable teachers, from primary to the current level of my education, for the important impact they have had on my career. I would also like to extend my appreciation and thanks to all my relatives, friends and fellow research students for their support and helping hands.

My profound gratitude goes to my sponsor, the Higher Education Commission (HEC) of Pakistan for fully funding my studies of this degree in the UK. Finally, I wish to thank my employer, the Higher Education Department of Khyber Pukhtoonkhwa, Pakistan for granting me study leave to undertake this programme.

Fazli Subhan,
Leicester, England, UK.
July, 2011.

Contents

List of Figures

List of Tables

Notations and abbreviations

Notation

[$\kappa(A)$] The *condition number* of a matrix A with respect to a given matrix norm . . . 12

[σ_{max}] Maximum singular value of a given matrix 12

[σ_{min}] Minimum singular value of a given matrix 12

[λ_{max}] Maximum eigenvalue of a given matrix 12

[λ_{min}] Minimum eigenvalue of a given matrix 12

[π_m^d] The space of d-variate polynomials of degree not exceeding m 14

[$h_{X,\Omega}$] The *fill distance* of the data corresponding to the data sites X in Ω 14

[q_X] The *separation distance* of the data sites X in a given domain 14

[$\lfloor r \rfloor$] The largest integer that does not exceed r, where $r \in \mathbb{R}$ 17

[$\lceil r \rceil$] The smallest integer not less than r, where $r \in \mathbb{R}$ 17

[$C^{2k}(\mathbb{R}^d)$] Space of smooth functions on $\mathbb{R}^d$ of smoothness order $2k$ 19

[$\varphi_A(\|\cdot\|)$] Anisotropic radial basis function 24

[$X^{p,\gamma}(\Omega)$] The space of all functions with derivatives up to order γ in $L_p(\Omega)$ 28

[$\mathfrak{X}_{\mathbf{l}}$] Anisotropic d-dimensional grid with mesh size $2^{-\mathbf{l}}$, $\mathbf{l} \in \mathbb{N}^d$ 28

[$V_{n,d}$] The discrete space of piecewise d-linear basis functions associated with the equally spaced grid of level n 30

[$V_{n,d}^s$] The sparse grid space of level n corresponding to the discrete space $V_{n,d}$ 32

[$\mathfrak{Y}_{n,d}^s$] Sparse grid of level n in d-dimensions 32

[$S_{A_{\mathbf{l}}}(\mathbf{x})$] The *anisotropic RBF interpolant to the data on the sub-grid* $\mathfrak{X}_{\mathbf{l}}$ *corresponding to the ARBF* φ_A 40

Abbreviation

[FDM]	Finite Difference Method	115

Chapter 1

Introduction

Multivariate scattered data interpolation and approximation is an important area of science and engineering that has many applications such as mapping problems in geodesy, geophysics and meteorology [56], [57], [58], [59], solution of partial differential equations [28], [29], [66], [67], fitting of potential energy surfaces in chemistry, coupling of engineering models with sets of incompatible parameters, non uniform sampling, e.g., medical imaging [18], [96], mathematical finance such as option pricing [78], computer graphics, e.g., representation of surfaces from point information such as laser rang scan data and image warping [1], [48], learning theory, neural networks, data mining [47], [4] and optimization.

In practice, the number of parameters or dimensions can go up to hundreds or may be even up to several thousands in some cases. Even with the most modern and fast developing computing technology, it is difficult and, sometimes, impossible to perform direct computer simulations of many large problems, especially in high dimensional settings. The main reason is the limitation of the computer memory and long simulation time.

Hence it is of crucial importance to develop efficient techniques and tools for simulating in computational mathematics, starting from tasks like how to define sets of points to approximate, interpolate, or to integrate certain classes of functions as good as possible, up to the numerical solutions of differential equations. This work concerns the development of a new interpolation method for the interpolation of high dimensional (large) data sets.

Let $f|_X = [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]^T \in \mathbb{R}^N$, sampled from an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ at a finite point set $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subset \mathbb{R}^d$, $d \geq 1$. The interpolation problem consists of finding a suitable function $s : \mathbb{R}^d \to \mathbb{R}$, (called the *interpolant*), satisfying the interpolation conditions

$$s(\mathbf{x}_i) = f(\mathbf{x}_i), \qquad 1 \leq i \leq N. \tag{1.1}$$

The fact that $\mathbf{x}_j$'s (the *data sites*) are allowed to lie in d-dimensional space $\mathbb{R}^d$ means that the formulation of the above problem allows to cover many different types of applications. If $d = 1$, the data could be a series of measurements such as heat of a body, taken over a certain time period, thus the data sites would correspond to contain time instances, or it could be a property that change only with altitude. For $d = 2$, we can think of the data being obtained over a planar region, and so $\mathbf{x}_j$ correspond to the two coordinates in the plane. For instance, we might want to produce a map which shows the rainfall in the country we live in based upon the data collected at weather stations located through out the country. For $d = 3$, we might think of an extension of this situation into space over the globe. One of the possibilities is that we could be interested in the temperature/pressure or any other chemical or physical property inside some solid body. Higher-dimensional examples might not be that intuitive, but a multitude of them exist, e.g., in areas such as:

- financial mathematics, where the price of an option based on d underlying stocks can be computed through solving the d-dimensional $Black - Scholes$ equation.
- molecular biology, is the chemical balance in a cell. There, the dimensions are the different substances that interact and influence the system.
- quantum mechanics, the number of dimensions in a system is proportional to the number of particles within a molecule.

Radial basis function (RBF) interpolation is a powerful tool for approximating and interpolating multidimensional data and/or when the data is scattered in its domain. Over the last four decades RBFs have been a fast growing research area and found to be widely successful for interpolation of scattered data, not only in mathematics but also in the applied sciences community. RBFs interpolation originated in the 1970's [56], [57], [58] and since that time has been successfully used in a variety of applications. In RBF interpolation the space from which the interpolant is chosen changes with the data sites of interpolation. Thus, the Mairhuber counter-example (which states that any method for which the interpolant does not change with data sites is doomed to fail for at least one choice of $N \geq 2$ in $\mathbb{R}^d$, $d \geq 2$) does not apply. Despite the attractive properties of RBFs such as nice convergence, easy implementation and their spatial dimension independence, the direct computation of RBF interpolation can be cumbersome. The respective linear system of the global RBFs turns out to be full in most cases. Moreover, the interpolation system is usually ill-conditioned as data density increases [31], [81], [82]. This instability can be circumvented by using more peaked basis functions, or compactly supported basis functions but at the cost of accuracy; even then only for moderate number N. On the other hand, using RBFs depending on a scaling parameter called the shape parameter such as Gaussians and multiquadrics etc, good accuracy is achieved with small shape parameter (flat RBFs). However, in this case the interpolation matrix becomes ill-conditioned and this results in very large interpolation coefficients, which can cause cancelation of terms when they are combined to get the interpolant. This may result in loss of evaluation accuracy. So even for small data sets, we have to have a compromise on accuracy for achieving stability. This is known as *the principle of uncertainty* [81]. So, a balance between the stability and convergence of RBF schemes is an issue of concern. The compactly supported RBFs produce sparser linear system, but this sparsity is lost as higher convergence rates are sought.

Solving the RBF system by non-customized methods such as Gaussian elimination, (exploiting symmetry) requires $\mathcal{O}(N^3)$ flops and $\mathcal{O}(N^2)$ storage. Moreover, a single direct evaluation of the interpolant requires $\mathcal{O}(N)$ operations. Experience with RBF interpolants on small problems has been almost universally positive. However, their applications to large problems with 10,000 or more centers have been limited due to prohibitive computational costs. For example, a problem with data size 20,000 requires approximately $1.5GB$ of core memory, and 10^{13} flops, which is impractical. So, large N implies unacceptable computational cost. On the other hand, the bad conditioning of the interpolation matrix makes the results unreliable.

It is now accepted, that direct methods are inappropriate for problems with $N \geq 10,000$ [7].

The RBFs literature contains many comments on the desirable properties of RBF interpolants and their computational difficulties. Here we quote some of them:

- [27] “The global interpolation methods with Duchon’s “thin plate splines” and Hardy’s multiquadrics are considered to be of high quality; however, their application is limited, due to computational difficulties, to ∼150 data points.”

- [84] “Practical problems often arise with many more than 10,000 data sites; for example, in aeromagnetic survey work it is common to have 50,000 to 100,000 observations in a single data set. We believe that such problems will indefinitely remain beyond the scope of thin-plate splines.”

- [36] “The most accurate results of registration of images with local distortions were obtained by using the surface spline mapping functions. As shown below, their direct use has extreme computing complexity and is not suitable for practical applications.”

These limitations make the use of classical RBF methods restricted to only few thousands of data sites, but practically large scale problems often arise with tens of thousands of data sites. For example, in aero-magnetic survey work it is common to have more than 50,000 data sites. So RBFs suffer from instabilities on one hand and the complexity issue on the other. Therefore it is natural to design strategies to prevent such instabilities and limitations. To address the phenomenon of these practical difficulties attached with RBFs schemes, several techniques/remedies have been suggested. Some of these techniques are such as methods of domain decomposition [26], [75], the introduction and application of compactly supported radial basis functions (CSRBFs) [80], [82], [92], [95], [30], [23], [22], [24] etc., multilevel interpolation methods [35], [65], preconditioning strategies such as local approximate cardinal function, least square approximate cardinal functions and domain decomposition together with approximate cardinal functions [73], [10], [7], [75], [69], [72], [34] etc. Fast and efficient methods for RBF interpolation can be found in [26], [75], [7], [8]. In [75], [7], approximate cardinal function pre-conditioner, a fast multiply and GMRES iterative approaches are combined to give a fast fitting RBF method. A domain decomposition method is given in [8], whereby the domain is subdivided into smaller overlapping domains.

Using RBFs with a scaling parameter called the shape parameter such as Gaussians and multiquadrics etc, good accuracy is achieved with small shape parameter (flat RBFs), if the attached instability is addressed. So it is of importance to have stable algorithms for small shape parameters, one such method is the Contour-Pade [40] of Fornberg and Wright, while another one is the RBF-QR method [39] developed for the case when the nodes are distributed over the surface of a sphere by Fornberg and Piret. Very recently, two stable algorithm for the evaluation of Gaussian RBF interpolant with flat kernels have been introduced, one by Fasshauer and Mccourt [32] and one by Fornberg, Larsson and Flayer [38]. In [38], the RBF-QR approach of [39] has been implemented in cases of 2-dimensional domains. RBF-QR method works for tens of points in one dimension, hundreds of points in two dimensions, and thousands of points in three dimensions. In [32], the authors present stable Gaussian interpolation in $\mathbb{R}^4$. These methods address the stability issues and are in general still limited to small problems and/or lower dimensions.

In [75], [7], it was shown that the combination of a suitable approximate cardinal function preconditioner, a fast multiplication, and GMRES iterations, make the solution of large RBF interpolation problems orders of magnitude less expensive in storage and operations. These authors present a fast fitting method for large N and show that it is now possible to find fast fitting coefficients for a data set (e.g a geophysical data) containing 20,000 points in less than 2 minutes. Domain decomposition method is another approach to address the efficiency and complexity issues. For example, in [8], an algorithm based on overlapping domain decomposition is used to solve a problem of size 10,000 in 7 seconds and the authors claim that existing code has been successfully used to fit data sets of up to 5 million points in 2 dimensions, and up to 250,000 points in 3 dimensions.

The implementation of RBFs on a computer is quite simple and insensitive to the dimension d of the data set space. This makes RBFs a potentially effective tool for application to high dimensional problems, if the size and complexity challenges described above are addressed. This work aims at addressing these challenges.

1.1 Objectives, Outline and Results

1.1.1 Motivation and objectives

Numerical simulations on gridded data are generally speaking limited to low dimensions. This limitation is known as the *curse of dimensionality*, a term due to Bellmann [9]. The computational expense of representing an approximation with a given accuracy ϵ, depends exponentially on the dimension d of the space $\mathbb{R}^d$ of the problem considered. For instance, to solve an interpolation problem on a uniform grid over a bounded domain $\Omega \subset \mathbb{R}^d$, the complexity estimate translates $\mathcal{O}(N^d)$ degrees of freedom, where N is size of the input data on the full grid. This is the reason for restriction to very few dimensions as stated above, even on the most modern and efficient machines. For example, for the resolution of 33 points in each coordinate direction for 6-dimensional problem, we need more than 10^9 degrees of freedom (DOF).

The issue of the curse of dimensionality can be circumvented to some extent by imposing stronger assumptions on the smoothness of the functions under consideration. Hyperbolic cross spaces were introduced in early 1960's, for example, Smolyak [86] and Babenko [2] in the context of numerical integration. In [86] and [2] the extra smoothness assumptions were used to construct quadrature rules for high dimensional functions based on the so called hyperbolic cross products.

More recently, Zenger in 1991 [97] introduced sparse grid methods for the solution of PDEs. These methods have also been used for interpolation and approximation [3], [90], [85], [52], [70] and are closely related in spirit with the hyperbolic cross product idea. The *sparse grid method* is based on a hierarchical basis (a representation of a discrete function space which is equivalent to the conventional nodal basis) and a sparse tensor (hyperbolic cross) product construction. Sparse grid methods have been introduced in the context of numerical methods for the solution of partial differential equations (PDEs) for finite difference methods [51], [53] and finite volume methods [61].

Sparse grid methods allows for a massive reduction of the amount of storage to solve a problem with given accuracy, provided the underlying functions to be approximated are sufficiently smooth. In particular, instead of the standard Sobolev spaces, the solution lies in the anisotropic Sobolev spaces (also known as Mixed Sobolev spaces). The sparse grid methods have been extended to non smooth solutions by adaptive refinement [51].

However, hyperbolic cross/sparse grid spaces consist of shifts of more than one basis function. This makes the use of analogous to hyperbolic cross/sparse grid-type ideas in the context of RBF interpolation particularly cumbersome. This is because, apart from the technical difficulties in implementing such interpolation procedures, there is no underlying theory for the solvability of the resulting interpolation problem. We note that in [83] hyperbolic cross products in one dimensional RBFs have been considered.

The purpose of this thesis is to present a new kernel-based interpolation method which overcomes the computational complexity and conditioning difficulties presented by large RBFs interpolation problems in higher dimensions. The new scheme uses direction-wise multilevel decomposition of structured or mildly unstructured interpolation data sites in conjunction with the application of kernel-based interpolants with different scaling in each direction. The new algorithm, which we name *Sparse Kernel-based Interpolation* (SKI) algorithm can be viewed as extension of the idea of sparse grids/hyperbolic cross to kernel-based functions.

The main idea is the use of anisotropic RBF interpolation in conjunction with the hyperbolic crosses in the form of the, so called, sparse grid combination technique.

The combination technique [54] is a sparse grid representation where partial solutions are evaluated on a certain sequence of coarser grids and the solution is obtained by linearly combining these partial solutions. By combining several partial approximations based on the translates of the *same* basis function, it is possible to achieve good approximation of smooth functions while keeping the memory requirements low. The dependence of the combination technique on the translates of the *same* basis function, makes it attractive for application in the context of d-variate RBFs. Due to the parallel nature of the combination technique and the practical insensitivity of RBF to the dimension parameter d, blending the two ideas could potentially prove to be effective for large and/or high dimensional problems. This is the idea the main motivation for the proposed *sparse kernel-based interpolation* method stems from.

In the context of RBF interpolation, the property that in each partial interpolation problem the basis functions is translation of a single function is of crucial importance to ensure the provable well-posedness of the interpolation problem. Due to the anisotropic nature of the data in the partial approximation in the combination technique, we make use of anisotropic radial basis function (ARBF) interpolation. ARBFs have proven to be effective to the interpolation of anisotropic data sets or functions [19], [20], [6]. Due to the insensitivity of the ARBFs to the dimension parameter d and the elementary nature of each partial ARBF interpolation problem, the implementation of the new method is straightforward in $\mathbb{R}^d$ for any d. The naturally parallel nature of the new method by solving the partial problems on the coarse data sets independently, makes it perfect for implementation in parallel fashion on modern high performance computing systems. The SKI method has shown to be effective for the solution of large problems: on an ordinary laptop computer it can solve an interpolation problems of moderate size of order $13,000$ (on an ordinary computer) and even more than 60,000 (on a high performance computer) nodes in smaller times and nearly with the same accuracy when compared to the classical RBFs interpolation in two dimensions. The computational savings become more evident in high dimensions: we have also implemented the method to high dimensional interpolation problems in $\mathbb{R}^d$ for $d = 3$, 4 with very encouraging numerical results.

The method is further extended to its multilevel version termed *multilevel sparse kernel-based interpolation* (MLSKI). The classical multilevel interpolation scheme introduced by Iske and Floater [35] and further studied by Iske, Levesley and Hales in [63], [65] and [55], makes use of hierarchical nested decomposition of the data. In the context of SKI, the nestedness property of the nodes in the partial approximations up to the final level corresponding to the data to be interpolated, gives rise to a natural implementation of the MLSKI. In principle, we follow the scheme given in [35] to recover the global features of the data by evaluating the sparse kernel-based interpolant on the sparse grid of level one and then recovering its local behavior by interpolating the residuals on the subsequent sparse grids. The multilevel application accelerates the convergence of SKI. MLSKI is faster and superior in convergence than the multilevel RBF interpolation. In addition MLSKI has the same nearly one-dimensional complexity of SKI and is, therefore, capable of solving larger and/or high dimensional problems.

1.1.2 Main achievements

The SKI and MLSKI algorithms aiming at high dimensional interpolation are proposed. We perform extensive numerical experiments by implementing the proposed SKI and its multilevel version to interpolate several d-variate functions for $d \geq 2$. In our experiments, we have implemented the naïve RBF approach to solve the partial interpolation problems in the SKI scheme. Thus, it is fair to compare our method with the naïve RBF interpolation method, as we shall do through out all the experiments reported in this thesis. We remark that the fast RBF methods mentioned above can be applied to accelerate SKI. The implementation of the method in all our experiments, on ALICE (computing cluster of the University of Leicester) as well as on an ordinary computer, has been done in *Matlab* and in serial. We use a desktop computer "Core 2 Duo CPU @ 3.16GHz 3.17GHz and 3.24GB of RAM" for d=2, 3. For d=4, we use ALICE to be able to run many experiments at the same time by accessing as many nodes of ALICE.

Each experiment was run in serial on a single node (having a pair of quad-core 2.67GHz Intel Xeon X5550 CPUs and 12GB of RAM) which is equivalent to a more powerful desktop computer. Thus all the CPU timing and the problem size shown here relate to computations that can be run on an ordinary computer mentioned above. Moreover, we are going to report separately the largest size of the problem that SKI can solve on a single node of ALICE for d =2, 3, 4 in Sections 4.3.2, 5.4.1.1(a), and in Tables 6.1, 6.2, 6.3.

The 2-dimensional results with direct SKI, are presented in Chapter 4, and they confirm the generally superior performance of SKI over the classical RBFs interpolation. The SKI visits some nodes more than once, but is still faster when machine time is considered as a function of the sparse grid nodes irrespective of the fact that this redundancy is included or excluded. SKI outperforms RBF in terms of accuracy as function of machine time. As for as the computations remain stable, the convergence is also faster if considered as a function of the input data size. When implemented on an ordinary computer mentioned above, SKI can solve an interpolation problem on a sparse grid of level up to 10, having 13,313. On a single node of ALICE, SKI solves interpolation on sparse grids of level up to 12, having more than 60,000 nodes for d=2. A single node of ALICE corresponds to a more power full desktop computer. Henceforth, ALICE should be understood as a desktop computer with more RAM. SKI algorithm generally shows better efficiency in terms of accuracy, complexity and the run time over the classical RBF method when the computation is stable.

SKI having nearly one-dimensional complexity, is capable of solving even larger interpolation problems in higher dimensions. We continue implementing the algorithm for the interpolation of a range of d-variate functions for d=3, 4. These high dimensional numerical results reconfirms the, generally, superior performance of the scheme observed in the 2-dimensional experiments. On an ordinary machine, the scheme can solve interpolation problem on sparse grids of level 8 containing 21,249 nodes for d=3. When d=4, SKI can solve interpolation on a sparse grid containing 52,993 nodes. In our experiments on ALICE, the maximum size of a d-variate interpolation problem that SKI can solve is 114,690 for d=3 and 331,780 nodes for d=4. The corresponding full grids would have sizes of order 10^9 and 10^{11} respectively. On the other hand, due the d-dimensional complexity the maximum size of classical RBF interpolation problem in our experience on ALICE is only nearly 15,000 regardless of the dimension. Hence SKI has better complexity in particular when it comes into high dimensional interpolation problems. The numerical results presented in Chapter 4 and Chapter 6, show that SKI algorithm is not only faster but also often superior in terms of convergence (it might be slower in some cases if error is considered as a function of the input data size), stability and complexity to the classical RBF interpolation in $\mathbb{R}^d$ for $d \geq 2$.

The SKI algorithm has been implemented with a wide range of RBFs. The use of positive definite RBFs such as Gaussian and inverse multiquadrics as basis functions admit good convergence, and among them the best convergence is observed with Gaussian. The slower convergence of SKI with inverse multiquadrics as the basis function, agrees with the superior convergence of Gaussian over inverse multiquadrics in the context of direct RBF interpolation. Among the conditionally positive definite (CPD) RBFs such as multiquadrics (MQ), poly-harmonics and radial powers, MQ produces nearly same convergence as the inverse multiquadrics. For other CPD RBFs such as thin plate splines, the convergence is either very slow or the scheme does not converge at all.

Apart from the encouraging performance of SKI on structured grids, its convergence on mildly irregular sparse grids is also observed.

Following the ideas of Floater and Iske [35], the multilevel sparse kernel-based interpolation is capable of accelerating the convergence of SKI by several order with same complexity and linear time. A series of numerical experiments with a wide range of RBFs for MLSKI to interpolate a d-variate function for d=2, 3, 4 can be found in Chapter 5 and Chapter 6. We observe from its numerical performance that the proposed MLSKI interpolation method is superior in terms of convergence, run time, and complexity as compared to classical multilevel RBF interpolation. The implementations of the methods in all our experiments, on ALICE as well as on an ordinary computer, have been done in *Matlab* and in a serial way. Since the proposed method is perfect for implementation in parallel on modern high performance computing (HPC) systems, its generally superior performance in terms of run time and complexity could be more visible if implemented in parallel.

1.1.3 Outline

This dissertation is organized as follows.

In Chapter 2, we give an overview of the RBF interpolation. This family of basis functions provides us with the main tools for our methods in this work.

In Chapter 3, linear sparse grid spaces are discussed followed by an introduction to their indirect version the so called combination technique.

In Chapter 4, we introduce the *sparse kernel-based interpolation* (SKI) method. The SKI is computed by evaluating anisotropic RBF interpolants on subsets of the data set and then linearly combining following the combination technique ideas. Numerical results for a range of RBF interpolants are presented. We make a distinction between Gaussian RBF and others as, Gaussian RBF has tensor product nature. Sparse grid techniques [11], [97], [54], [14] are based on hierarchical tensor product multi-linear basis.

In Chapter 5, the multilevel version of the new sparse kernel-based interpolation MLSKI is presented. In high dimension, achieving a high degree of accuracy could be difficult but this might be required in some problems. The purpose of the multilevel sparse kernel-based interpolation is to accelerate the convergence of SKI algorithm. The implementation of MLSKI is somewhat analogous to the standard multilevel RBF interpolation [35], [63], [65], [55]. SKI is naturally fit for multilevel settings as the data sets used in each level of the SKI algorithm are nested.

In Chapter 6, we perform 3-variate and 4-variate interpolation of 4-dimensional and 5-dimensional data, respectively. We compare our results for direct and multilevel versions of the sparse kernel-based method to the classical and multilevel RBF interpolation (MLRBF).

In Chapter 7, we draw some conclusions on the new developments presented in this work and we discuss the overall performance of the proposed methods. Finally we briefly discuss our plan about future research and ideas on possible extension of the current results.

Chapter 2

Radial Basis Functions

2.1 The scattered data interpolation problem

The scattered data interpolation problem reads: given a set of measurements (called the *data sets*) obtained at certain locations (called *data sites*), we are interested in finding a rule (e.g., an unknown function) which allows us to deduce information about the process we are studying also at the locations different from those at which measurements are available. In other words, the objective is to find a function $s(\cdot)$ which is a good fit to the given data. One way of constructing such a function $s(\cdot)$ is by requiring that $s(\cdot)$ passes through the data, i.e., matches the given measurements at the corresponding locations. This approach is called *interpolation.*

Definition 2.1 (Scattered data interpolation) *Given data* $(\mathbf{x}_i, y_i)$*;* $i = 1, \dots, N$*, with* $\mathbf{x}_i = (x_1, \dots, x_d) \in \mathbb{R}^d$*,* $y_i \in \mathbb{R}$*, find a smooth function* $s : \mathbb{R}^d \to \mathbb{R}$ *such that* $s(\mathbf{x}_i) = y_i$*,* $i = 1, \dots, N$ *where* s *is a suitable function called interpolant to the data.*

A common approach to solve the scattered data interpolation problem is to assume that the interpolant $s(\cdot)$ is a linear combination of certain basis functions, $\beta_i(\mathbf{x})$; $i = 1, \dots, N$, i.e.,

$$s(\mathbf{x}) = \sum_{i=1}^{N} c_i \beta_i(\mathbf{x}) \qquad \mathbf{x} \in \mathbb{R}^d. \tag{2.1}$$

Then the interpolation conditions

$$s(\mathbf{x}_i) = y_i, \;\; i = 1, \dots, N, \tag{2.2}$$

lead to the linear system

$$A\mathbf{c} = \mathbf{y}, \tag{2.3}$$

with
$A = [\beta_j(\mathbf{x}_i)] \in \mathbb{R}^{N\times N}$, $1 \le i,j \le N$, $\mathbf{c} = [c_1,\dots,c_N]^T \in \mathbb{R}^N$, $\mathbf{y} = [y_1,\dots,y_N]^T \in \mathbb{R}^N$.

A unique solution of this problem exists, if and only if the interpolation matrix A is non singular and the well-posedness of the problem depends upon the choice of the basis functions $\{\beta_j : j = 1,\dots,N\}$. In the next section, we recall some of the basic concepts such as positive definiteness, matrix norm, condition number of a matrix, stability of a linear system etc, that will be useful for the understanding of the solution and implementation of the interpolation problem.

2.2 Basic concepts

Definition 2.2 (l_p-norm) *Let $A : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$. The $p-norm$ or $l_p - norm$, $p \ge 1$ of matrix A induced by the vector norm $\|\cdot\|_p$ is defined as*

$$\|A\|_p = \max_{\mathbf{v}\in\mathbb{R}^n\setminus\{0\}} \frac{\|A\mathbf{v}\|_p}{\|\mathbf{v}\|_p},$$

where, $\|\mathbf{v}\|_p = \{\sum_{i=1}^n |v_i|^p\}^{\frac{1}{p}}$.

Some of the commonly used norms are, $\|\mathbf{v}\|_2 = \{\sum_{i=1}^n |v_i|^2\}^{\frac{1}{2}}$, $\|A\|_2 = \sqrt{\rho(A^TA)}$, $\|\mathbf{v}\|_1 = \sum_{i=1}^n |v_i|$, $\|A\|_1 = \max_{1\le j\le n}\sum_{i=1}^m |A_{ij}|$, $\|A\|_\infty = \max_{1\le i\le m}\sum_{j=1}^n |A_{ij}|$ and $\|\mathbf{v}\|_\infty = \max_{1\le i\le n}|v_i|$.

Definition 2.3 (Condition number of a matrix) *The condition number of a matrix A with respect to any matrix norm $\|\cdot\|$ is defined as*

$$\kappa(A) = \|A\|\|A^{-1}\|.$$

In the 2-norm, the condition number $\kappa(A)$ is given by

$$\kappa(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_{max}}{\sigma_{max}},$$

and, in particular, when A is positive definite it becomes $\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}}$.

The condition number of a matrix or a linear system is of fundamental importance in computations. Conditioning is a measure of the sensitivity/stability to the perturbations made by the algorithm used to solve the problem on a computer. In practice perturbations in the input data are unavoidable, for instance caused by the rounding-off errors due to machine precision. A problem sensitive to small perturbations is termed as ill-conditioned and it suffers from instabilities when solved on a machine. A numerical algorithm is stable if the output depends continuously on the input data.

To see this, consider the linear system

$$A\mathbf{y} = \mathbf{b}. \tag{2.4}$$

Let us perturb the right side of (2.4) from $\mathbf{b}$ to $\tilde{\mathbf{b}}$ and let $\tilde{\mathbf{y}}$ be its new solution, then it can be shown ([62],[89],[87],[49],[91]) that

$$\frac{\delta \mathbf{y}}{||\mathbf{y}||} \leq \kappa(A)\frac{||\delta \mathbf{b}||}{||\mathbf{b}||}, \tag{2.5}$$

where $\delta\mathbf{y} = ||\mathbf{y} - \tilde{\mathbf{y}}||$ is the error corresponding to the perturbation $\delta\mathbf{b} = ||\mathbf{b} - \tilde{\mathbf{b}}||$ in the right side $\mathbf{b}$.

If the matrix A in (2.4) is perturbed from $\mathbf{A}$ to $\tilde{\mathbf{A}}$ then one gets [91]

$$\frac{\delta \mathbf{y}}{||\mathbf{y}||} \leq \kappa(A)\frac{||\delta A||}{||A||}, \tag{2.6}$$

where $\delta A = ||A - \tilde{A}||$ is the perturbation in the matrix A. So from inequalities (2.5) and (2.6), we see that if $\kappa(A) = \mathcal{O}(1)$, the system (2.4) is not sensitive to small perturbations in A and in $\mathbf{b}$ and is, therefore, well-conditioned. The bounds in (2.5) and (2.6) are sharp at least theoretically, that is there exist $\mathbf{b}$, $\delta\mathbf{b}$,$\mathbf{x}$, $\delta\mathbf{x}$, A, δA for which equality holds in (2.5) and (2.6). If $1 \lll \kappa(A) < \infty$, we might get large $\frac{\delta\mathbf{y}}{||\mathbf{y}||}$ for small $\frac{\delta\mathbf{b}}{||\mathbf{b}||}$. In this case (2.4) is called ill-conditioned. If $\epsilon_{machine}$ denotes the machine precision then the relative error is given by

$$\frac{||\mathbf{y} - \tilde{\mathbf{y}}||}{||\mathbf{y}||} = \mathcal{O}(\kappa(A)\epsilon_{machine}) \tag{2.7}$$

Equation (2.7) shows that we might be losing $\lfloor log_{10}(\kappa(A))\rfloor$ digits in computing the solution $\mathbf{y}$, where for any real number r, $\lfloor r \rfloor$ stands for the largest integer that does not exceed r. Practically due to the, so-called, *effective well conditioning* [21], the relative error might be substantially smaller than that predicted by the bounds given in (2.5) and (2.6); however due the error propagation, it may not be possible to always quantify or guarantee this effective well conditioning in the case of large $\kappa(A)$ when the interpolation step is only one part of a numerical algorithm.

Definition 2.4 (Multi-index notation) *Let $\mathbb{N}_0$ denote the set of non-negative integers. A d-dimensional multi-index is a d-tuple $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$. For $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we define $|\alpha| = \alpha_1 + \cdots + \alpha_d$, $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_d^{\alpha_d}$, and*

$$D^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d} = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}.$$

Let us denote by π_m^d the space of d-variate polynomials of degree not exceeding m. The polynomial space π_m^d has the following properties (see, e.g., [94]).

Theorem 2.5 *The mononomials* $\mathbf{x} \mapsto \mathbf{x}^\alpha \quad \mathbf{x} \in \mathbb{R}^d$, $\alpha \in \mathbb{N}_0^d$ *are linearly independent and* $dim\pi_m^d = \begin{pmatrix} m+d \\ d \end{pmatrix}$.

Definition 2.6 (Condition of unisolvancy) *The data sites* $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subseteq \mathbb{R}^d$ *with* $N \geq M = dim\pi_m^d$ *are called* π_m^d*-unisolvent if the zero polynomial is the only polynomial from the space* π_m^d *that vanishes on all of them.*

For instance, $\pi_1^2 = span(1, x_1, x_2)$, $x_1, x_2 \in \mathbb{R}$, is a bivariate linear space and $\dim\pi_1^2 = 3$. Every polynomial in π_1^2 describes a plane in the three dimensional space and this plane is uniquely determined by three points if and only if these points are not collinear. Thus three points in $\mathbb{R}^2$ are π_1^2 -unisolvent if and only if they are not collinear.

A generalization of this fact, that can be found in [94], [31], is the following theorem.

Theorem 2.7 *Suppose that* $\{l_0, l_1....l_m\}$ *is a set of* $m+1$ *distinct lines in* $\mathbb{R}^2$ *and that* $X = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ *is a set of* $M = dim\pi_m^2 = (m+1)(m+2)/2$ *distinct points such that the first point* $\mathbf{x}_1$ *lies on* l_0*, the next two points* $\mathbf{x}_2$ *and* $\mathbf{x}_3$ *lie on* l_1 *but not on* l_0*, next three points* $\mathbf{x}_4$*,* $\mathbf{x}_5$ *and* $\mathbf{x}_6$ *lie on* l_2 *but not on* l_0 *and* l_1*, and so on, so that last* $m+1$ *points lie on* l_m *but not on any of the previous lines* $l_0, l_1....l_{m-1}$*. Then* X *is* π_m^2*-unisolvent.*

Definition 2.8 (Fill distance) *We define the* fill distance *of the data corresponding to the data sites* X *in* Ω $h_{X,\Omega}$ *by*

$$h_{X,\Omega} = \sup_{\mathbf{x}\in\Omega} \min_{\mathbf{x}_j\in X} ||\mathbf{x} - \mathbf{x}_j||_2.$$

The *fill distance* is used as a measure of the data distribution. A geometric interpretation of the *fill distance* is given by the radius of the largest open ball inside Ω that does not contain a data site. This decreases by increasing the number of uniformly distributed data sites.

Definition 2.9 (Separation distance) *We define the* separation distance *of the data sites* X *in a given domain* q_X *by*

$$q_X = \frac{1}{2} \min_{i\neq j} ||\mathbf{x}_i - \mathbf{x}_j||_2.$$

The *separation distance* is sometimes also referred to as the packing radius. Physically the *separation distance* is the maximum $r > 0$ such that all the open spheres $\{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x} - \mathbf{x}_j||_2 < r\}$ do not overlap as shown in Figure 2.1.

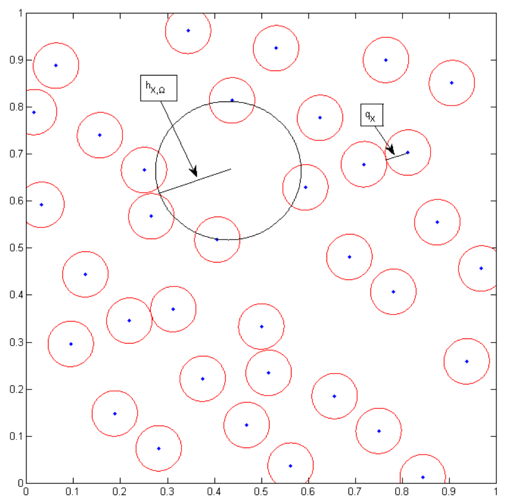


Figure 2.1: The separation and the fill distance for 35 Halton points.

Radial basis functions (which we are going to discuss later in this chapter) are the basic tools used in this work. All popular choices of these basis functions are not of the same kind in the context of the well-posedness of the the associated interpolation system. For the clarity of our later discussion about the well posedness of the resulting linear system, we briefly discuss the notions of positive definite function, conditionally positive definite function and the associated matrices such as positive definite and positive semi-definite.

Definition 2.10 (Symmetric positive definite matrix) *A real symmetric matrix A is called positive semi-definite if its associated quadratic form is non negative, i.e.,*

$$\mathbf{c}^T A\mathbf{c} = \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k A_{jk} \geq 0$$

for $\mathbf{c} = [c_1, ..., c_N]^T \in \mathbb{R}^N$. If the above quadratic form is zero only for $\mathbf{c} \equiv \mathbf{0}$, then A is called positive definite.

Definition 2.11 (Positive definite (PD) functions) *A continuous real function Φ :*

$\mathbb{R}^d \to \mathbb{R}$ is called positive semi-definite on $\mathbb{R}^d$ if and only if

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \Phi(\mathbf{x}_i - \mathbf{x}_j) \geq \mathbf{0}$$

for any N pairwise different points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^d$, and $\mathbf{c} = [c_1, ..., c_N]^T \in \mathbb{R}^N$. The function Φ is called positive definite on $\mathbb{R}^d$ if the above quadratic form is zero only when $\mathbf{c} \equiv \mathbf{0}$.

Definition 2.12 (Conditionally positive definite (CPD) functions) *A continuous real function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is said to be conditionally positive definite of order m (CPDm) on $\mathbb{R}^d$ if and only if*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \Phi(\mathbf{x}_i - \mathbf{x}_j) > \mathbf{0}$$

holds for all possible pairs $(\mathbf{c}, X)$ of choices for $\mathbf{c} = [c_1, ..., c_N] \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ and $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subseteq \mathbb{R}^d$ satisfying the vanishing moments conditions

$$\sum_{j=1}^{N} c_j P(\mathbf{x}_j) = 0,$$

for all $P \in \pi^d_{m-1}$.

Some important properties of positive definite/positive semi-definite/conditionally positive definite functions are list below:

- A conditionally positive definite function of order $m = 0$ on $\mathbb{R}^d$, is positive definite on $\mathbb{R}^d$
- If $\phi_1(.), \cdots, \phi_d(.)$ are positive definite and integrable on $\mathbb{R}$. Then the tensor product function

  $$\Phi(\mathbf{x}) = \phi_1(x_1) \cdots \phi_d(x_d), \quad \mathbf{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d,$$

  is also positive definite.
- A function that is conditionally positive definite of order m on $\mathbb{R}^d$ is also conditionally positive definite of any higher order. In particular, a positive definite function is always conditionally positive definite of any order.
- The product of two (and hence a finite number) of positive definite functions is also positive definite.

- If $\Phi_1(.), \cdots, \Phi_d(.)$ are positive semi definite, $c_j \geq 0$, $j = 1, \cdots, n$, then $\Phi(\cdot) = c_1\Phi_1(\cdot) + \cdots + c_n\Phi_n(\cdot)$ is also positive semi definite and is positive definite if one of the $\Phi_j(\cdot)'s$ is positive definite together with the corresponding positive c_j.

We have listed here few elementary properties of positive definite functions. More details can be found in the review article [88].

We continue with restricting the discussion to basis functions with radial symmetry.

Definition 2.13 *A function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is called radial provided there exist a univariate function $\varphi : [0, \infty) \to \mathbb{R}$ such that $\Phi(\mathbf{x} - \mathbf{x}_j) = \varphi(r)$, where $r = \|\mathbf{x} - \mathbf{x}_j\|$ and $\| \cdot \|$ is some norm on $\mathbb{R}^d$ (usually the Euclidean norm).*

This definition says that Φ at any point $\mathbf{x} \in \mathbb{R}^d$ at certain distance from a fixed point $\mathbf{x}_j \in \mathbb{R}^d$ (or the origin) is constant. The fixed point $\mathbf{x}_j$ is called its center. Hence Φ is radially symmetric about its center. An interesting property of radial functions for applications is the fact that the interpolation problem becomes insensitive to the dimension d of the space in which the data sites lie. Instead of having to deal with a multivariate function Φ (whose complexity will increase with increasing space dimension d), we can work with the same univariate function φ for all choices of d. One only needs to evaluate the distances of the points from the given center of the basis function.

Here the radial basis functions $\varphi(|| \cdot -\mathbf{x}_j||)$ are translates/shifts of a conditionally positive definite radial function $\varphi : [0, \infty) \to \mathbb{R}$ of order m. While there may be circumstances that suggest choosing the centers $\mathbf{x}_j$ different from the data sites, in most cases we pick the centers to coincide with the data sites. Of course, one can envision many other ways to construct an N-dimensional data-dependent basis for the purpose of interpolation. The use of shifts of one single basic function $\varphi(||\cdot||)$ makes the radial basis function approach elegant in practice, and the respective mathematical theory tractable.

In Table 2.1, we list some of the most commonly used globally supported radial basis functions (GSRBFs) [64]. In Table 2.1 m is the order of the conditionally positive definite RBF on $\mathbb{R}^d$, $K_\nu(r)$ is a modified Bessel function of order $\nu > 0$, and for any real number r, $\lfloor r \rfloor$ stands for the largest integer that does not exceed, while $\lceil r \rceil$ denotes the smallest integer not less than r.

Among the GSRBFs listed in Table 2.1 the Gaussian, multiquadrics and inverse multiquadrics have a parameter c, which adjust the shape of the RBF and is therefore usually called its *shape parameter*. A smaller value of c causes the function to become flatter, while increasing c leads to a more peaked RBF and, therefore, localises its influence. The radial symmetry and shape dependence of some radial basis functions is shown in Figure 2.2.

Table 2.1: Globally supported Radial Basis Functions.

Radial Basis Function	$\varphi(r) =$	Parameters	Order m
Gaussians	$e^{-(cr)^2}$	$c > 0$	$m \geq 0$
Matern	$r^\nu K_\nu(r)$	$\nu > 0$	$m \geq 0$
Exponential	e^{-cr}	$c > 0$	$m \geq 0$
Radial powers	r^ν	$\nu > 0,\ \nu \notin 2\mathbb{N}$	$m \geq \lceil \frac{\nu}{2} \rceil$
Polyharmonic Splines	$(-1)^{k+1} r^{2k} log(r)$	$k \in \mathbb{N}$	$m \geq k+1$
Multiquadrics(MQs)	$(1+c^2r^2)^{\frac{\nu}{2}}$	$\nu > 0,\ \ \nu \notin 2\mathbb{N},\ \ c > 0$	$m \geq \lceil \frac{\nu}{2} \rceil$
Inverse Multiquadrics(IMQs)	$(1+c^2r^2)^{\frac{\nu}{2}}$	$v < 0,\ c > 0$	$m \geq 0$

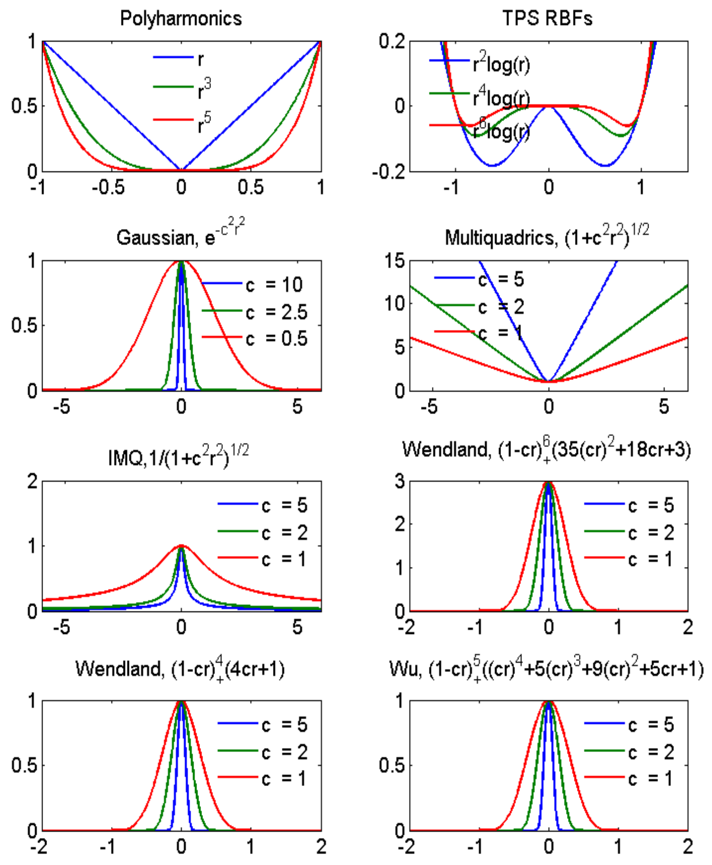


Figure 2.2: Radial Basis Functions, in the compactly supported RBFs $1/c$, is their support radius.

We note that the interpolation matrix in (2.13) associated to GSRBFs is always full and therefore the computational efficiency becomes unaffordable as the size of the problem grows. There also exist families of radial basis function with compact support. In order to have a sparser linear system for the interpolation problem, compactly supported radial basis functions (CSRBFs) have been introduced. These compactly supported function were introduced by Wu [95] and further developed and modified by Wendland [92, 94]. The CSRBFs are positive definite (CPD) of order $m = 0$ on $\mathbb{R}^d$ but only for fixed maximal d-value given in Table 2.2.

Wendland's functions, as defined in [92], are of the form

$$\varphi_{d,k}(r) = \begin{cases} P_{d,k} & for\ r\ \in\ [0,1], \\ 0 & for\ r \geq 1, \end{cases} \tag{2.8}$$

with a univariate polynomial $P_{d,k}$ of degree $\lfloor d/2 \rfloor + 3k + 1$. Clearly support of this function is normalized to the unit interval $[0, 1]$. Moreover $\varphi_{d,k}(r) \in C^{2k}(\mathbb{R}^d)$. Some of the Wendland compactly supported RBFs are given in Table 2.2

Table 2.2: Wendland's compactly supported Radial Basis Functions.

Dimension d	Radial Basis Function	C^{2k}
$d = 1$	$\varphi_{1,0}(r) = (1 - cr)_+$	C^0
	$\varphi_{1,1}(r) = (1 - cr)_+^3(3cr + 1)$	C^2
	$\varphi_{1,2}(r) = (1 - cr)_+^5(8(cr)^2 + 5cr + 1)$	C^4
$d \leq 3$	$\varphi_{3,0}(r) = (1 - cr)_+^2$	C^0
	$\varphi_{3,1}(r) = (1 - cr)_+^4(4cr + 1)$	C^2
	$\varphi_{3,2}(r) = (1 - cr)_+^6(35(cr)^2 + 18cr + 3)$	C^4
	$\varphi_{3,3}(r) = (1 - cr)_+^8(32(cr)^3 + 25(cr)^2 + 8cr + 1)$	C^6
$d \leq 5$	$\varphi_{5,0}(r) = (1 - cr)_+^3$	C^0
	$\varphi_{5,1}(r) = (1 - cr)_+^5(5cr + 1)$	C^2
	$\varphi_{5,2}(r) = (1 - cr)_+^7(16(cr)^2 + 7cr + 1)$	C^4

Wu's compactly supported positive definite function are constructed in [95]. As in the case of Wendland's functions, Wu's function are also positive definite on $\mathbb{R}^d$ but only for fixed maximal d-value given in Table 2.3. Some of the Wu's function are given in Table 2.3.

Table 2.3: Wu's compactly supported Radial Basis Functions.

Dimension	Radial Basis Function	C^{2k}
$d = 1$	$\psi_{2,0} = (1-cr)_+^5(1+5cr+9(cr)^2+5(cr)^3+r^4)$	C^4
$d \leq 3$	$\psi_{2,1} = (1-cr)_+^4(4+16cr+12r^2+3(cr)^3)$	C^2
$d \leq 5$	$\psi_{2,2} = (1-cr)_+^3(8+9cr+3(cr)^2)$	C^0
$d = 1$	$\psi_{3,0} = (1-cr)_+^7(5+35cr+101(cr)^2+147(cr)^3+101(cr)^4+35(cr)^5+5(cr)^6)$	C^6
$d \leq 3$	$\psi_{3,1} = (1-cr)_+^6(6+36cr+82(cr)^2+72(cr)^3+30(cr)^4+5(cr)^5)$	C^4
$d \leq 5$	$\psi_{3,2} = (1-cr)_+^5(8+40cr+48(cr)^2+25(cr)^3+5(cr)^4)$	C^2
$d \leq 7$	$\psi_{3,3} = (1-cr)_+^4(16+29cr+20(cr)^2+5(cr)^3)$	C^0
$d \leq 5$	$\psi_{4,2} = (1-cr)_+^7(48+336cr+928r^2+1120r^3+720r^4+245r^5+35r^6)$	C^4
$d \leq 7$	$\psi_{4,3} = (1-cr)_+^6(64+384cr+640r^2+515r^3+210r^4+35r^5)$	C^2
$d \leq 9$	$\psi_{4,4} = (1-cr)_+^5(128+325cr+345r^2+175r^3+35r^4)$	C^0

The symbol $(\cdot)_+$ denotes the *cutoff* function defined as follows

$$(x)_+ = \begin{cases} x & for\ x \geq 0, \\ 0 & for\ x < 0. \end{cases} \tag{2.9}$$

2.3 RBF interpolation

Consider a data vector $f|_X = [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]^T \in \mathbb{R}^N$ of function values, sampled from an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ at a finite point set $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subset \mathbb{R}^d$, $d \geq 1$.

We define the RBF interpolant s by

$$s(\cdot) = \sum_{j=1}^{N} c_j \varphi(|| \cdot - \mathbf{x}_j ||) + P(\cdot), \qquad P \in \pi_{m-1}^d, \tag{2.10}$$

Further, we denote by $M := dim\pi_{m-1}^d$ and by $p_1, ..., p_M$ a basis of π_{m-1}^d.

In the case of a positive definite radial basis function, we can choose $m = 0$. Nevertheless, addition of a polynomial P of total degree at most $m - 1$, guarantees the exactness of the interpolation for $f \in \pi_{m-1}^d$.

To specify c_j and P, we impose the interpolation conditions

$$s|_X = f|_X,$$

on the interpolant s to get

$$\sum_{j=1}^{N} c_j \varphi(||\mathbf{x}_k - \mathbf{x}_j||) + \sum_{l=1}^{M} d_l p_l(\mathbf{x}_k) = f(\mathbf{x}_k), \quad for\ 1 \le k \le N. \tag{2.11}$$

Conditions (2.11) form a linear system of N equations with $N+M$ unknown variables: the coefficient vector $\mathbf{c} = [c_1, ..., c_N]^T \in \mathbb{R}^N$ of the major part and $\mathbf{d} = [d_1, ..., d_M]^T \in \mathbb{R}^M$.

To eliminate the additional M degrees of freedom, we add the following M constraint conditions:

$$\sum_{k=1}^{N} c_k p_l(\mathbf{x}_k) = \mathbf{0}, \quad for\ 1 \le l \le M. \tag{2.12}$$

The linear system of $M + N$ equations given by (2.11) and (2.12) can be written as

$$\begin{pmatrix} A_{\varphi,X} & P_X \\ P_X^T & \mathbf{O} \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} f|_X \\ \mathbf{0} \end{pmatrix}, \tag{2.13}$$

where

$$A_{\varphi,X} = [\varphi(||\mathbf{x}_k - \mathbf{x}_j||)]_{1 \le j,k \le N} \in \mathbb{R}^{N \times N}, \tag{2.14}$$

$$P_X = [p_l(\mathbf{x}_k)]_{1 \le k \le N, 1 \le l \le M} \in \mathbb{R}^{N \times M}, \tag{2.15}$$

$$\mathbf{O} = \text{Null matrix} \in \mathbb{R}^{M \times M}, \tag{2.16}$$

$$\mathbf{0} = \text{Null vector} \in \mathbb{R}^{M}. \tag{2.17}$$

We now discuss the existence of a unique solution to the above interpolation problem.

Suppose $m = 0$, i.e., φ is positive definite. In this case, appending a polynomial to interpolant s as given in (2.10) is not a requirement and this interpolant reduces to the simpler form,

$$s(\cdot) = \sum_{j=1}^{N} c_j \varphi(|| \cdot - \mathbf{x}_j||). \tag{2.18}$$

The interpolation conditions

$$s|_X = f|_X$$

give the following simpler system (as opposed to (2.13))

$$A_{\varphi,X}\mathbf{c} = f|_X, \tag{2.19}$$

where $A_{\varphi,X}$, $\mathbf{c}$, and $f|_X$ have the same meaning as discussed above.
Since φ is positive definite, the matrix $A_{\varphi,X}$ is positive definite, hence (2.19) has a unique solution.

Suppose now that $m \geq 1$, i.e., investigate the solvability of the augmented problem given by (2.13).

Before discussing the solution of (2.13), we consider its corresponding homogenous system:

$$A_{\varphi,X}\mathbf{c} + P_X d = \mathbf{0}, \tag{2.20}$$

$$P_X^T \mathbf{c} = \mathbf{0}. \tag{2.21}$$

Multiplying (2.20) by $\mathbf{c}^T$, we get

$$\mathbf{c}^T A_{\varphi,X}\mathbf{c} + (P_X^T\mathbf{c})^T\mathbf{d} = \mathbf{0},$$

and using (2.21), we deduce

$$\mathbf{c}^T A_{\varphi,X}\mathbf{c} = \mathbf{0}. \tag{2.22}$$

The fact that φ is CPD of order $m \geq 1$ and $\mathbf{c}$ satisfies the constraints implies that $\mathbf{c} \equiv \mathbf{0}$ and so equation (2.20) becomes $P_X\mathbf{d} = \mathbf{0}$.
This implies $\mathbf{d} = \mathbf{0}$, provided that P_X is injective, i.e., if the point set X is π^d_{m-1}-unisolvent (see Definition (2.6)). So the corresponding homogenous system has unique solution $[\mathbf{c}, \mathbf{d}]^T = \mathbf{0}$, if X is π^d_{m-1}-unisolvent. This guarantees the non-singularity of the matrix A.

Hence the well-posedness of the RBF problem is guaranteed by imposing the condition that the set of centers has to contain π^d_{m-1}-unisolvent subset. This is a mild condition that can be easily met in most cases of practical interest.

Among the RBFs given in Table 2.1, the most popular in applications are the multiquadric (MQ) $\varphi(r) = \sqrt{c^2 + r^2}$ with $m = 1$, the thin plate splines (TPS2) $\varphi(r) = r^2 log(r)$ (which is a member of the polyharmonic spline family) with $m = 2$, the Gaussians $\varphi(r) = e^{-c^2r^2}$ with $m = 0$, and the inverse multiquadrics (IMQ) $\varphi(r) = \frac{1}{\sqrt{c^2+r^2}}$

with $m = 0$. Although MQ is CPD of order $m = 1$, the associated interpolation problem without the additional constant polynomial part has unique solution [74]. If $c = 0$ the MQ boils down to the *RBF* $\varphi(r) = r$ which also produces an interpolation system which remains non singular when no polynomial term is included in the the interpolant.

It is worth noting that we have only shown the well-posedness of the problem when one uses the *Euclidean* distances in defining s. The case for 1-norm and $d = 2$, can produce singular interpolation matrix, if the data points are vertices of a closed polygon [71]. This was further studied in [5], where the well-posedness in more general form with p-norm, $p \in (0, 1)$ was proven. In [5] it was also shown that, if both p and the dimension d exceed 2, then it is possible to construct sets of distinct points which generate a singular interpolation matrix.

The interpolation matrix in (2.13) is usually full, and, moreover when the number of interpolation points N is large, then the corresponding linear system is usually ill-conditioned. This numerical instability may not come from the interpolation problem but from the basis used being nearly linearly dependent as the separation distance decreases. Solving this system by ordinary methods such as Gaussian elimination exploiting symmetry requires $\mathcal{O}(N^3)$ flops and $\mathcal{O}(N^2)$ storage. These computational costs are unacceptable for large N. So RBFs suffer from instabilities on one hand and from complexity on the other.

We have few stable algorithms for RBFs interpolation with shape parameters resulting flatter basis functions. One of such methods is the *Contour-Pade* [40] and another one is the *RBF-QR method* [39]. Very recently, two stable algorithm for the evaluation of Gaussian RBF interpolant with flat kernels have been introduced by Fasshauer and Mccourt [32] and by Fornberg, Larsson and Flayer [38]. In [38], the *RBF-QR* method presented in [39] has been implemented for two dimensional data. *RBF-QR* works for tens of points in one dimension, hundreds of points in two dimensions, and thousands of points in three dimensions. These method address the stability issues, but are in general limited to small problems and/or lower dimensions.

In [75], [7] it was shown that the combination of a suitable approximate cardinal function pre-conditioner, a fast multiplication, and GMRES iteration, make the solution of large RBF interpolation problems orders of magnitude less expensive in storage and operations. Domain decomposition method is another approach to address the efficiency and complexity issues [8].

2.4 Anisotropic RBF interpolation

It is common to have data sets whose variation in one direction is much slower or faster than that in the other directions. It is also possible that the distribution of the data sites in the domain is anisotropic. The RBFs are radially symmetric having hyper-spheres as their level surfaces. To interpolate/approximate data with anisotropy of any of the kinds discussed above, anisotropic radial basis functions have been introduced and used in practice [6], [19], [20]. Anisotropic radial basis functions have hyper-ellipsoids as their level surfaces and are therefore also known as elliptic basis functions. The application of anisotropic basis functions have proved to be numerically effective for local fitting [19], [20].

Definition 2.14 *Let $\varphi(\|\cdot - \mathbf{x}_j\|)$ be any given RBF with its center at $\mathbf{x}_j \in \mathbb{R}^d$ and $A \in \mathbb{R}^{d\times d}$ be a suitably chosen invertible matrix, then the* anisotropic radial basis function *(ARBF) $\varphi_A(\|\cdot\|)$ is defined as*

$$\varphi_A(\|\cdot - \mathbf{x}_j\|) \; = \; \varphi(\|A(\cdot - \mathbf{x}_j)\|).$$

The *ARBF* $\varphi_A(\|\cdot\|)$ coincides with $\varphi(\|\cdot\|)$ when A is the $d\times d$ identity matrix. When A is non scalar matrix, (i.e., $A \neq kI_d$, $k \in \mathbb{R}$), the basis function $\varphi_A(\|\cdot\|)$ is not radially symmetric, but it is still symmetric about its centre $\mathbf{x}_j$. The axes of symmetry of the level sets of φ_A depends upon the construction of the non singular matrix A. In general, the level sets of φ_A are hyper-ellipsoids with their centers coincident with the center of the anisotropic RBF. In the case in which A is a scaling matrix (i.e., a diagonal matrix with positive diagonal entries), these hyper-ellipsoids will have their axes parallel to the co-ordinate directions. Gaussian RBF and the corresponding anisotropic Gaussian RBF centred at (0.5, 0.5) for $A = \begin{pmatrix} 2^5 & 0 \\ 0 & 2^2 \end{pmatrix}$ are exemplified in Figure 2.4.

We now discuss the solution of an anisotropic interpolation problem. Suppose $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \Omega \subset \mathbb{R}^d$ and $\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in X, \; i = 1, \ldots, N\}$ is the given data. Let φ be a conditionally positive definite radial function of order m, and let $\{p_1, \ldots, p_M\}$ be a basis of the polynomial space π^d_{m-1} and finally $A \in \mathbb{R}^{d\times d}$ be a suitably chosen invertible matrix.

The *anisotropic radial basis functions interpolant S_A* is defined as:

$$S_A(\mathbf{x}) = \sum_{j=1}^{N} c_j \varphi_A(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{M} d_k p_k(A\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{2.23}$$

If we impose the interpolation conditions

$$S_A(\mathbf{x}_i) \; = \; y_i, \;\; i = 1, \ldots, N, \tag{2.24}$$

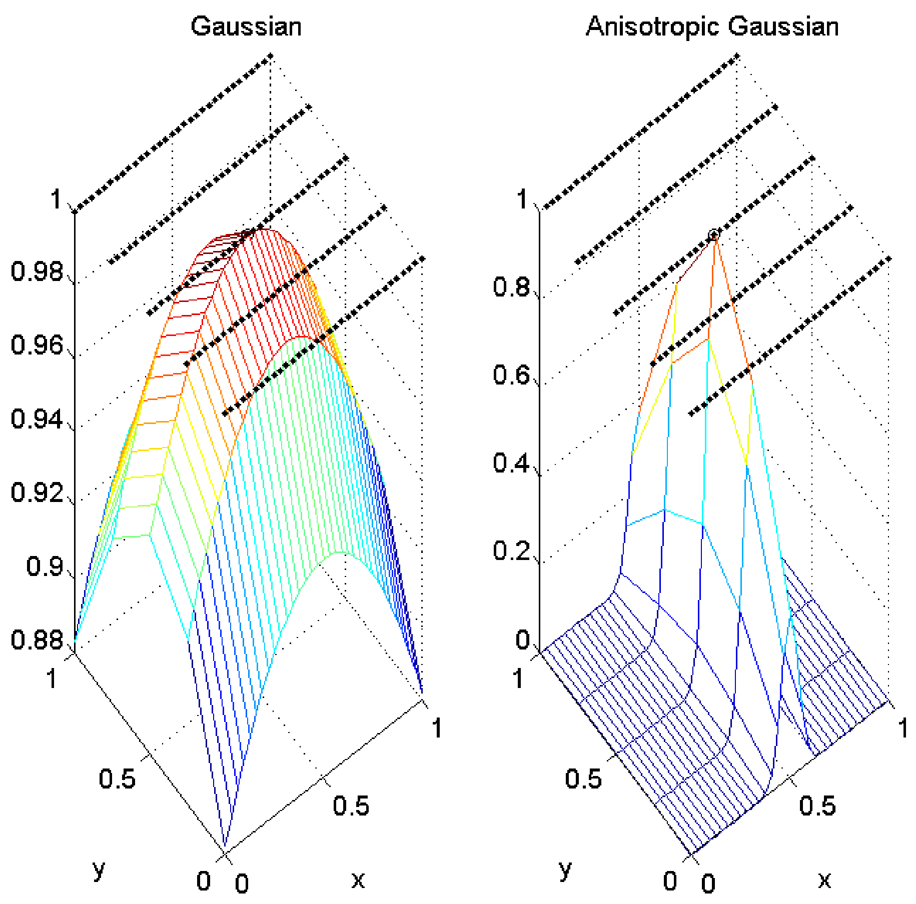


Figure 2.3: Gaussian RBF and the corresponding anisotropic Gaussain RBF, with shape parameter 0.5 and center at (0.5, 0.5)

the additional constraint conditions

$$\sum_{j=1}^{N} c_j p_k(A\mathbf{x}_j) \;=\; 0, \quad k = 1, \dots, M, \tag{2.25}$$

and, finally, definig the set of transformed data sites $\{\mathbf{u}_1, \dots, \mathbf{u}_N\}$, where $\mathbf{u}_i \;=\; A\mathbf{x}_i,\ i = 1, \cdots, N$, we get the following linear system.

$$\sum_{j=1}^{N} c_j \varphi(\|\mathbf{u}_i - \mathbf{u}_j\|) + \sum_{k=1}^{M} d_k p_k(\mathbf{u}_i) \;=\; y_i, \quad i = 1, \dots, N. \tag{2.26}$$

$$\sum_{j=1}^{N} c_j p_k(\mathbf{u}_j) \;=\; 0, \;\; k = 1, \dots, M. \tag{2.27}$$

The non singularity of the transformations matrix A and the π^d_{m-1}-unisolvency of the data sites $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ imply that the set $\{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ of transformed data sites is also π^d_{m-1}-unisolvent. This guarantees the welposedness of the above linear system. Thus the existence of a unique solution to the linear system given by (2.24) and (2.25) of the anisotropic RBFs interpolation problem is guaranteed by the same π^d_{m-1}-unisolvency condition on the data sites as required for the classical RBFs interpolation problem, as long as the transformation matrix A chosen to be non singular (discussed in detail in Section 2.3). We refer to the recent paper [6] for the error analysis of anisotropic RBF interpolation.

Chapter 3

Hyperbolic Crosses and Sparse Grids

We review the linear hyperbolic cross product spaces and the related sparse grid method based on multi-linear basis functions. The key idea is the use of hyperbolic cross product spaces. The use of such approximation spaces is known since the early 1960's [2], [86]. In [86], Smolyak studied quadrature formulas based on a tensor product of lower-dimensional operators. More recently the sparse grid methods, introduced by Zenger [97] in 1991 have been applied to the numerical solutions of partial differential equations in the context of finite difference and finite element methods.

3.1 Piecewise linear interpolation

We discuss the interpolation of smooth functions by using piecewise d-linear hierarchical bases based on tensor products. Let $\Omega := [0,1]^d$ and $u : \Omega \to \mathbb{R}$. Let $\alpha \in \mathbb{N}_0^d$ be a d-dimensional multi-index, i.e., $\alpha = (\alpha_1, \dots, \alpha_d)$ where $\alpha_j \in \mathbb{N}_0,\ j = 1, \dots, d$, with its 1-norm and ∞-norm defined respectively, as:

$$|\alpha|_1 \;=\; \sum_{j=1}^{d} \alpha_j \quad \text{and} \quad |\alpha|_\infty \;=\; \max_{1 \le j \le d} \alpha_j.$$

3.1.1 Hierarchical multilevel subspaces

For $p \in [2, \infty]$, we define the following spaces:

$$X^{p,\gamma}(\Omega) \;=\; \{u : \Omega \to \mathbb{R} \;:\; D^\alpha u \in L_p(\Omega), |\alpha|_\infty \le \gamma\}$$

and

$$X_0^{p,\gamma}(\Omega) \;= \{u \in X^{p,\gamma}(\Omega) \;:\; u|_{\partial\Omega} \;=\; 0\},$$

i.e., $X^{p,\gamma}(\Omega)$ is the space of all functions with derivatives up to order γ in $L_p(\Omega)$ and $X_0^{p,\gamma}(\Omega)$ is the subspace of $X^{p,\gamma}(\Omega)$ consisting of those functions u that vanish on the boundary $\partial\Omega$ of the domain Ω.

For $u \in X^{p,\gamma}(\bar{\Omega})$ and for a multi index α, with $|\alpha|_\infty \leq \gamma$, we define

$$|u|_{\alpha,\infty} \;=\; \|D^\alpha u\|_\infty$$

and

$$|u|_{\alpha,2} \;=\; \left(\int_{\bar{\Omega}} |D^\alpha u|^2 dx\right)^{\frac{1}{2}}.$$

We define the family of anisotropic d-dimensional grids $\{\mathfrak{X}_{\mathbf{l}} : \mathbf{l} \in \mathbb{N}_0^d\}$ *on* Ω, where $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}_0^d$ with mesh size $h_{\mathbf{l}} = (h_{l_1}, \ldots, h_{l_d}) = 2^{-\mathbf{l}} = (2^{-l_1}, \ldots, 2^{-l_d})$, that is, the grid $\mathfrak{X}_{\mathbf{l}}$ has different, in general, but equidistant nodes in each individual coordinate direction. The multi-index $\mathbf{l}$ with zero components are required only for defining hierarchical multi-linear basis corresponding to the boundary when the solution u is not zero on the boundary. Hence, $\mathfrak{X}_{\mathbf{l}}$ consists of the following points:

$$\mathbf{x}_{\mathbf{l},\mathbf{i}} \;=\; (x_{l_1,i_1}, \ldots, x_{l_d,i_d}) \;=\; \mathbf{i} \cdot h_{\mathbf{l}}, \;\; \mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}},$$

or

$$x_{l_j,i_j} \;=\; i_j h_{l_j} \;=\; i_j 2^{-l_j}, \quad \textit{where}\;\; 0 \leq i_j \leq 2^{l_j}, \;\; 1 \leq j \leq d.$$

In what follows, the multi index $\mathbf{l}$ will be the index associated with the grid $\mathfrak{X}_{\mathbf{l}}$, a point $\mathbf{x}_{\mathbf{l},\mathbf{i}}$ or a basis $\phi_{\mathbf{l},\mathbf{i}}$, while the multi-index $\mathbf{i}$ will be associated with the location of $\mathbf{x}_{\mathbf{l},\mathbf{i}}$ in $\mathfrak{X}_{\mathbf{l}}$.

We consider the standard one dimensional linear function (also known as the hat function) $\phi : \mathbb{R} \to \mathbb{R}$ with

$$\phi(x) = (1-x)_+ = \begin{cases} 1 - |x| \;\; \textit{if} \;\; x \in [-1,1]; \\ \quad 0 \quad , \textit{otherwise}, \end{cases} \tag{3.1}$$

and we define the scaled piecewise linear basis functions $\phi_{l_j,i_j}(x_j)$ by

$$\phi_{l_j,i_j}(x_j) := \phi(\frac{x_j - i_j h_{l_j}}{h_{l_j}}) = \phi(2^{l_j} x_j - i_j). \tag{3.2}$$

The support of $\phi_{l_j,i_j}(x_j)$ is given by

$$\textit{supp}(\phi_{l_j,i_j}) = [x_{l_j,i_j} - h_{l_j}, x_{l_j,i_j} + h_{l_j}] \;=\; [(i_j-1)h_{l_j}, (i_j+1)h_{l_j}].$$

We define the piecewise d-linear basis function $\phi_{\mathbf{l},\mathbf{i}}$ as the tensor product of the scaled linear basis functions $\phi_{l_j,i_j}(\cdot)$ in each dimension, viz.,

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j) \tag{3.3}$$

An example of piecewise bilinear basis functions for d=2 is shown in Figure 3.1.

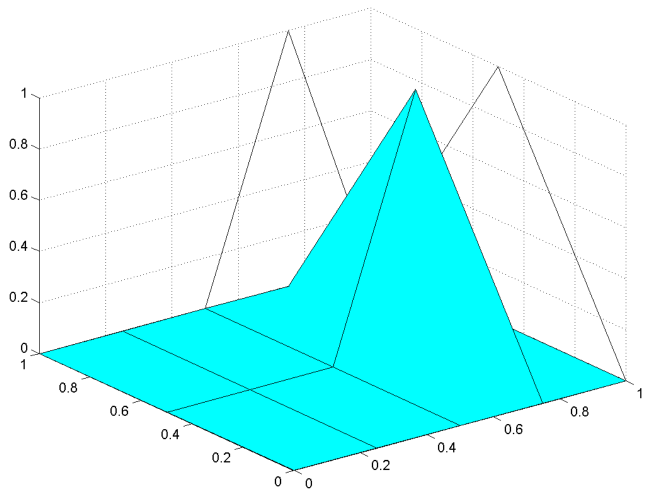


Figure 3.1: Piecewise bilinear basis function $\phi_{\mathbf{l},\mathbf{i}}$ on $\mathfrak{X}_{\mathbf{l}}$ with $\mathbf{l} = (2, 1)$, $\mathbf{i} = (3, 1)$.

We define the space $V_{\mathbf{l}}$ of piecewise d-linear basis functions associated with the grid $\mathfrak{X}_{\mathbf{l}}$ by

$$V_{\mathbf{l}} = \text{span}\{\phi_{\mathbf{l},\mathbf{i}} \;\; : \;\; \mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}}\}. \tag{3.4}$$

Finally we define the index set

$$I_{\mathbf{l}} = \left\{ \begin{array}{ll} \mathbf{i} \in \mathbf{N}_0^d : & i_j = 1, \ldots, 2^{l_j} - 1, \; i_j \; \textit{is odd}, \; \forall \, j = 1, \ldots, d, \; l_j > 0 \\ \quad \textit{and} & i_j = 0, \; 1, \; i_j \; \textit{is odd}, \; \forall \, j = 1, \ldots, d, \; l_j = 0 \end{array} \right\}. \tag{3.5}$$

In the case in which the function to be interpolated is zero on the boundary, $I_{\mathbf{l}}$ is given instead by

$$I_{\mathbf{l}} = \left\{ \; \mathbf{i} \in \mathbf{N}_0^d : \; i_j = 1, \ldots, 2^{l_j} - 1, \; i_j \; \textit{is odd}, \; \forall \, j = 1, \ldots, d, \; l_j > 0 \; \right\}. \tag{3.6}$$

We define the space $W_{\mathbf{l}}$ as follows

$$W_{\mathbf{l}} = \text{span}\,\{\phi_{\mathbf{l},\mathbf{i}} : \;\; \mathbf{i} \in I_{\mathbf{l}}\} \tag{3.7}$$

Note that $W_{\mathbf{l}}$ is equivalent to the difference space:

$$W_{\mathbf{l}} = V_{\mathbf{l}} \setminus \bigoplus_{j=1}^{d} V_{\mathbf{l}-\mathbf{e}_j}, \tag{3.8}$$

where $\mathbf{e}_j$ is the j-th unit vector in $\mathbb{R}^d$, using the convention that $V_{\mathbf{l}} = \{0\}$ if $\mathbf{l}_j = -1$ for at least one $j \in \{1, \ldots, d\}$. Then, from (3.7) and (3.8) we deduce

$$V_{\mathbf{l}} = \bigoplus_{\mathbf{k} \leq \mathbf{l}} W_{\mathbf{k}} = \bigoplus_{k_1=0}^{l_1} \ldots \bigoplus_{k_d=0}^{l_d} W_{\mathbf{k}} \tag{3.9}$$

for $\mathbf{k} = (k_1, \ldots k_d)$. This suggests a different grouping

$$\{\phi_{\mathbf{l},\mathbf{i}} \ : \ \mathbf{i} \in I_{\mathbf{k}}, \ \ \mathbf{k} \leq \mathbf{l}\} \tag{3.10}$$

for the hierarchical basis of $V_{\mathbf{l}}$.

For any positive integer $n \in \mathbb{N}$, such that $l_i = n$ for $i = 1, \cdots, d$, then $V_{\mathbf{l}}$ is the space of piecewise d-linear basis functions associated with the equally spaced grid of level n and we denote it by $V_{n,d}$. Hence

$$V_{n,d} = \bigoplus_{|\mathbf{k}|_\infty \leq n} W_{\mathbf{k}}, = \bigoplus_{k_1=0}^{n} \ldots \bigoplus_{k_d=0}^{n} W_{\mathbf{k}}, \tag{3.11}$$

for $\mathbf{k} = (k_1, \ldots k_d)$, so that we have

$$\{\phi_{\mathbf{l},\mathbf{i}} \ : \ \mathbf{i} \in I_{\mathbf{k}}, \ \ |\mathbf{k}|_\infty \leq n\},$$

as the *hierarchical basis* of the space $V_{n,d}$. This space for n=3 and d=2 is exemplified in Figure 3.2. Hence we can represent each $u \in V_{n,d}$ as follows

$$u(\mathbf{x}) = \sum_{|\mathbf{l}|_\infty \leq n} u_{\mathbf{l}}, \quad u_{\mathbf{l}} = \sum_{\mathbf{i} \in I_{\mathbf{l}}} c_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}), \text{ with } \phi_{\mathbf{l},\mathbf{i}} \in W_{\mathbf{l}} \tag{3.12}$$

where the $c_{\mathbf{l},\mathbf{i}}$ are the coefficients of the hierarchical tensor product basis representation.

The number of basis required is thus $(2^n + 1)^d$ which is the same as for the nodal (multilinear spline) basis.

Now we state an important result related to hierarchical representation of $V_{n,d}$ by the spaces $W_{\mathbf{l}}$, taken from [11].

Lemma 3.1 *For $u \in X_0^{2,2}(\Omega)$, the estimate for its component $u_{\mathbf{l}} \in W_{\mathbf{l}}$ is given by*

$$\|u_{\mathbf{l}}\|_2^2 \leq C(d) 2^{-4|\mathbf{l}|_1} |u|_{2,2}^2,$$

where, $|u|_{2,2}^2 = \|D^2 u\|_2^2 = \int_{\bar{\Omega}} |D^2 u|^2 dx$

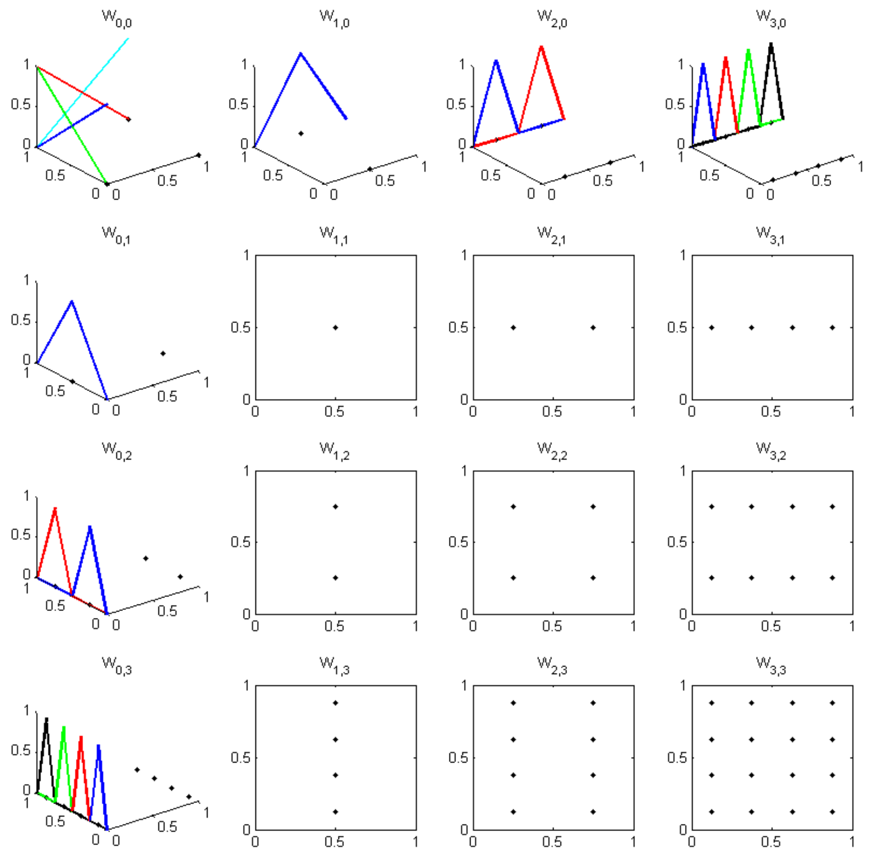


Figure 3.2: Subspaces $W_{\mathbf{l}}$ for space $V_{3,2}$

3.2 Hyperbolic cross products/sparse grids

We now use the hierarchical representation of the d-linear functions space $V_{\mathbf{l}}$ to define the hyperbolic cross product/sparse grid spaces. The representation of u as given in (3.12) and Lemma 3.1 shows that the upper bound of $\|u_{\mathbf{l}}\|_2$ is proportional to $2^{-2|\mathbf{l}|_1}$ and the influence that u gets from the support of the basis functions $\phi_{\mathbf{l},\mathbf{i}}$ in $W_{\mathbf{l}}$. The idea of hyperbolic cross product/sparse grid methods is to exclude the basis functions with comparatively smaller influence in the discrete space $V_{n,d}$ of level n to get the corresponding sparse grid space.

If we replace $|\mathbf{l}|_\infty$ by $|\mathbf{l}|_1$ in (3.11-3.12), we obtain the sparse grids space $V^s_{n,d} \subset V_{n,d}$ defined by

$$V^s_{n,d} = \bigoplus_{|l|_1 \leq n} W_{\mathbf{l}}, \tag{3.13}$$

whose basis is given by $\{\phi_{\mathbf{l},\mathbf{i}},\ \mathbf{i} \in I_k,\ |k|_1 \leq n\}$. Thus, every $u \in V^s_{n,d}$ has the following representation

$$u(\mathbf{x}) = \sum_{|l|_1 \leq n} u_{\mathbf{l}}, \text{ where } u_{\mathbf{l}} = \sum_{\mathbf{i} \in I_{\mathbf{l}}} c_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}. \tag{3.14}$$

The basis for $V_{3,d}$, $d = 2$, is given in Figure 3.2, while the basis for the sparse grid space $V^s_{3,d}$, $d = 2$, is given in Figure 3.3.

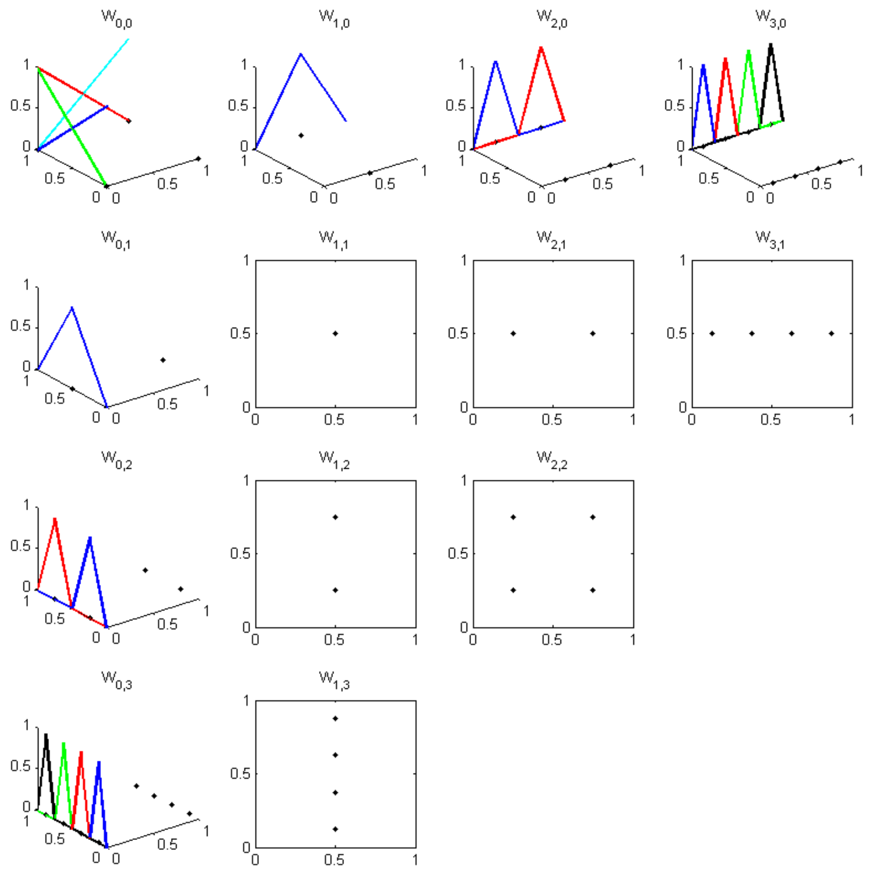


Figure 3.3: Subspaces $W_{\mathbf{l}}$ for space $V^s_{3,d}$, d=2

The grid created as a result of the approximation space $V^s_{n,d}$ is the sparse grid of level n in d-dimensions, which we denote by $\mathfrak{Y}^s_{n,d}$, corresponding to a full grid of level n in the domain $\Omega \in \mathbb{R}^d$. The full grid and sparse grid corresponding to $V_{n,d}$ and $V^s_{n,d}$ for the cases n=3, 4 and d=2 are given in Figures 3.4 and 3.5.

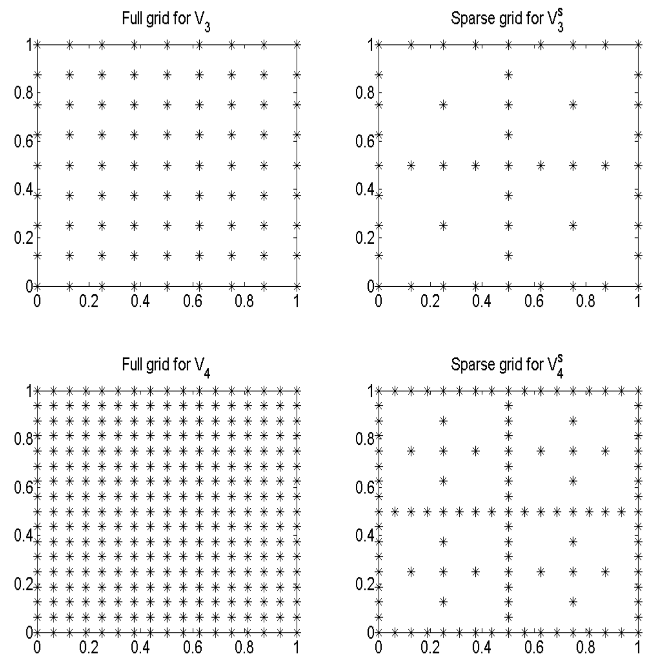


Figure 3.4: The full and sparse grids

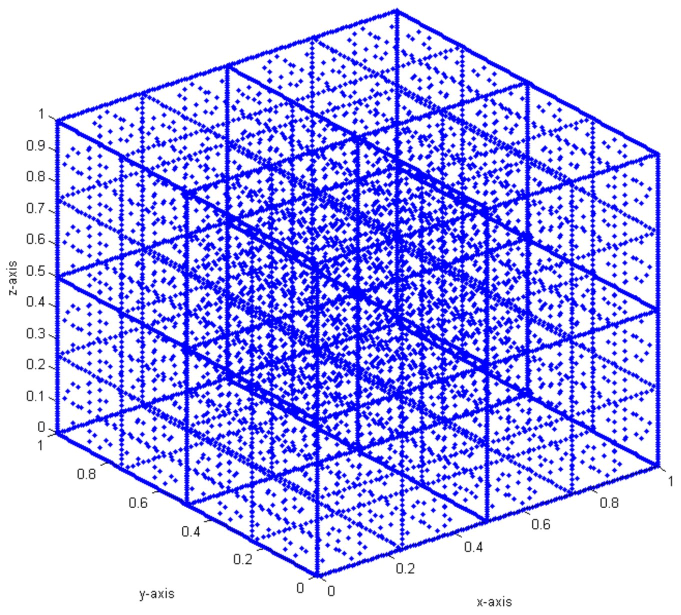


Figure 3.5: Sparse Grid $\mathfrak{Y}^s_{7,3}$, of level 7 in 3D

When the function is zero on the boundary, we consider the sparse grid space $V^s_{0,n,d}$ instead of $V^s_{n,d}$, which is defined by

$$V^s_{0,n,d} = \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}}.$$

3.2.1 Approximation order and size of sparse grid spaces

The hyperbolic cross/sparse grid method substantially reduces the computational complexity at a moderate cost in terms f accuracy, allowing for the numerical treatment of problems with $d \geq 2$. We reiterate that the approximation theory for sparse grids requires stronger smoothness conditions in comparison to the full tensor product approximation spaces. In particular, the so called anisotropic Sobolev norms are used instead of the standard Sobolev norms [11]. In [11], it is shown that the dimension of $V_{n,d}$ is given by $|V_{n,d}| = \mathcal{O}(2^{nd})$, while the dimension of $V^s_{n,d}$ is given by

$$\begin{aligned} |V^s_{n,d}| &= \mathcal{O}(2^n n^{d-1}) \\ &= \mathcal{O}(h_n^{-1}(\log(h_n^{-1}))^{d-1}), \ for \quad h_n = 2^{-n}. \end{aligned}$$

The full grid multi-linear approximation using the space $V_{n,d}$ has $\mathcal{O}(h^2) = \mathcal{O}(2^{-2n})$ accuracy in the case of sufficiently smooth functions, while with sparse grids based on multi-linear basis functions, this accuracy decreases only by a logarithmic factor to

$$\mathcal{O}(h^2(\log h^{-1})^{d-1}) \;=\; \mathcal{O}(2^{-2n}n^{d-1}).$$

These results are summarized in Table 3.1. The dimension of the sparse grid space is far less than the full grid space, while the accuracy decreases only by a logarithmic factor. The small number of degrees of freedom results in the reduction of computation and storage requirements. This advantage becomes more prominent as the dimension of the problem increases.

Table 3.1: Approximation and Complexity of Sparse grids

	Full grid	Sparse grid
degrees of freedom	$\mathcal{O}(2^{nd})$	$\mathcal{O}((2^n n^{d-1}))$
Approximation order	$\mathcal{O}(2^{-2n})$	$\mathcal{O}(2^{-2n}n^{d-1})$

3.3 Sparse grid combination technique

As we saw, the sparse grid/hyperbolic cross product spaces consist of translations of different basis functions belonging to the spaces $W_{\mathbf{l}}$, defined above. This renders the use of these ideas in the context of RBF interpolation particularly cumbersome. This is because, apart from the technical difficulties in implementing such interpolation procedures, there is no underlying theory for the solvability of the resulting interpolation problem. Moreover, the implementation of direct sparse grid/hyperbolic cross methods requires the construction of new computer codes.

The sparse grid combination technique is an indirect way of constructing the sparse grid discretisations by making use of the known full grid methods, hence allowing for the use of existing full grid codes. The sparse grid combination technique for linear interpolation was introduced by Griebel, Zenger and Bungartz [54], [13], [14]. A sparse grid, $\mathfrak{Y}^s_{n,2}$, can be be represented as superposition of several much coarser but full grids $\mathfrak{X}_{\mathbf{l}}$, with $|\mathbf{l}|_\infty \leq n$. This combination is described in Figures 3.6 and 3.7. By combining several smaller approximations based on the translates of the *same* basis function, it is possible to achieve an acceptable approximation and keep the memory requirements low. Hence, this is potentially a very attractive property in the context of RBF interpolation problem.

To this end, we consider the interpolation problem of a function u on a certain sequence of anisotropic grids $\mathfrak{X}_{\mathbf{l}}$, $\mathbf{l} \;=\; (l_1, \ldots, l_d)$, with different but equidistant mesh

$$\begin{array}{rlllll} \mathfrak{Y}^s_{4,2} = & \mathfrak{X}_{4,1} & & & & \\ & \oplus & \mathfrak{X}_{3,2} & \ominus & \mathfrak{X}_{3,1} \\ & \oplus & \mathfrak{X}_{2,3} & \ominus & \mathfrak{X}_{2,2} \\ & \oplus & \mathfrak{X}_{1,4} & \ominus & \mathfrak{X}_{1,3} \end{array}$$

Figure 3.6: Sparse grid as superposition of coarser full grids

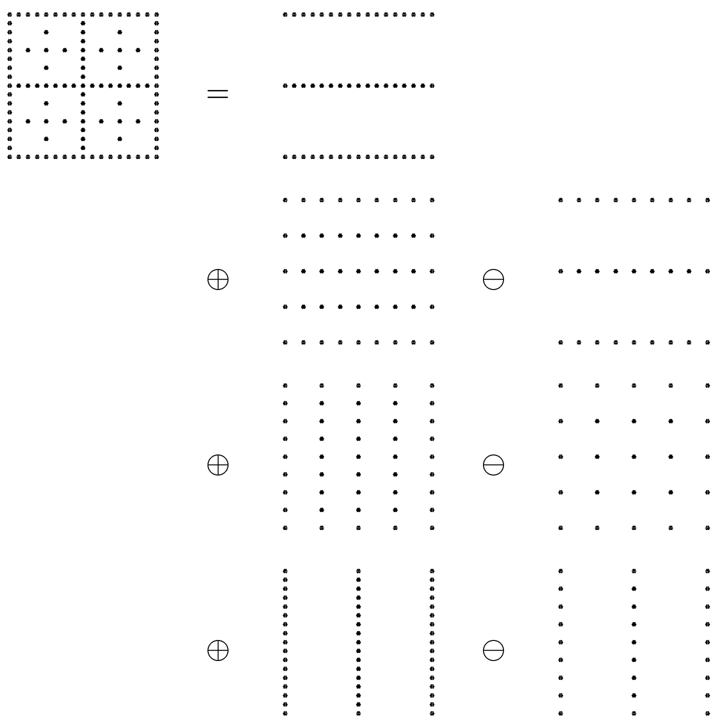


Figure 3.7: Sparse grid $\mathfrak{Y}^s_{4,2}$ as a combination of coarser full grids

size in each coordinate direction. For this purpose we consider all grids $\mathfrak{X}_{\mathbf{l}}$ such that

$$|\,\mathbf{l}\,|_1 = \; l_1 + \ldots + l_d \; = \; n - q, \;\; q = 0, \ldots, d-1, \;\; l_j \; \geq \; 0,$$

The discrete partial interpolant on $\mathfrak{X}_{\mathbf{l}}$ with respect to the nodal basis function $\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x})$ is given by

$$u_{\mathbf{l}} \; = \; \sum_{i_1=0}^{2^{l_1}} \cdots \sum_{i_d=0}^{2^{l_d}} c_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}).$$

These partial interpolants $u_{\mathbf{l}}$ obtained on the different coarser full grids $\mathfrak{X}_{\mathbf{l}}$ are then linearly combined according to the following combination formula [25], [54], [44]:

$$u_n(\mathbf{x}) = \sum_{q=0}^{d-1} (-1)^q \begin{pmatrix} d-1 \\ q \end{pmatrix} \sum_{|\mathbf{l}|_1 = n+(d-1)-q} u_l(\mathbf{x}). \qquad (3.15)$$

For instance, the last formula has the following structure

for $d = 2$:

$$u_n = \sum_{|\mathbf{l}|_1 = n+1} u_{\mathbf{l}} - \sum_{|\mathbf{l}|_1 = n} u_{\mathbf{l}} \tag{3.16}$$

and, for $d = 3$:

$$u_n = \sum_{|\mathbf{l}|_1 = n+2} u_{\mathbf{l}} - 2 \sum_{|\mathbf{l}|_1 = n+1} u_{\mathbf{l}} + \sum_{|\mathbf{l}|_1 = n} u_{\mathbf{l}}. \tag{3.17}$$

Thus, only interpolation problems on standard grids need to be treated and the sparse grid interpolant is obtained by a simple linear combination.

Remark 3.2 *Note that each full grid space contains translations of the* same *basis function. This is of crucial importance for the construction of the new fast algorithm presented in Chapter 4.*

The sparse grid combination technique has mainly two advantages over direct classical as well as direct sparse grid methods. Firstly, obtaining the partial solution $u_{\mathbf{l}}$ on the full grids $\mathfrak{X}_{\mathbf{l}}$ of small resolution allows the possibility of using the existing codes. Secondly, the discretisation on the small (but full and anisotropic in general) grids can be done in a completely parallel fashion. This makes the combination technique perfectly suited for implementation on the modern high performance computing systems [50], [44], [46], [42]. The combination technique makes use of some nodes several times but still it is making use of the same data sites as required for the direct sparse grid approach. However, compared to the full grids, it still requires very small amount of memory storage and it performs well for large data, especially in high dimensions. The combination technique has been successfully applied to give better efficiency in terms of stability, complexity and the run time at the cost of negligible loss of accuracy. Some recent developments on the combination technique can be found in [45], [60], while application to a number of different problems has appeared, such as solution of multidimensional options pricing PDEs [42]and machine learning [43].

Chapter 4

Sparse kernel-based interpolation

Due to their simple structure and insensitivity of their implementation to the dimension, RBFs are potentially a strong tool for high dimensional interpolation problems. The convergence rates of RBF interpolation are directly related to a measure of the data density, for example the fill/seperation distance of the full grid. By keeping the fill and separation distance fixed, the grid size and, hence the size of the interpolation problem grows exponentially with dimension d of the problem space. For example, a uniform grid in $[0,1]^d \subset \mathbb{R}^d$ with fill distance $h = 1/N$, where N is the number of points in each direction, has size $(h^{-1}+1)^d$. This makes RBFs useful in practice only for low dimensional problems and feasible in high dimensions only for small grids.

As discussed in Chapter 3, sparse grid/hyperbolic cross methods address the restrictions on the data size for linear interpolation at the cost of losing only a logarithmic factor in convergence when applied to high dimensional interpolation. Direct sparse grid methods based on the tensor product of one-dimensional standard RBFs have been considered in [83], where the error analysis has been given based on the resulting tensor-product nature of the basis functions, but no numerical assessment of the resulting methods was given there in. Unlike the approach in [83], the scheme we are going to propose is based on the standard d-variate RBFs in conjunction with an indirect sparse grid method, the so called, combination technique.

The direct sparse grid/hyperbolic cross spaces are spanned by translates of different basis functions. In the context of d-variate RBFs, this makes the method impractical, as the solution of the resulting interpolation problem may not be guaranteed. The combination technique for multilinear sparse grid interpolation described in Section 3.3 is based on translations of the *same* basis function. Motivated by this observation, we propose a new interpolation method termed *Sparse Kernel-based Interpolation* (SKI) method. The method is based on anisotropic radial basis function interpolation on partial grids in conjunction with a combination technique inspired by the corresponding ideas for linear interpolation described above. The new SKI method turns out to be very efficient and stable even for the interpolation of large data in higher dimensions.

4.1 Sparse kernel-based interpolation

Let $\Omega := [0,1]^d$, and let $u : \Omega \to \mathbb{R}$ i.e. $u(\mathbf{x}) \in \mathbb{R}$, for $\mathbf{x} = (x_1, \ldots, x_d) \in \Omega$. Let $\{(\mathbf{x}_i, u_i),\ u_i = u(\mathbf{x}_i),\ i = 1, \cdots, N\}$, be the data to be interpolated sampled from the unknown function u at a finite point set $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subset \Omega$. The interpolation problem consists of finding a suitable function (called the *interpolant*), $s : \mathbb{R}^d \to \mathbb{R}$ satisfying the interpolation conditions

$$s(\mathbf{x}_i) = u(\mathbf{x}_i), \qquad 1 \le i \le N. \tag{4.1}$$

We define a family of standard anisotropic d-dimensional grids. For a multi-index $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$, we define the family of grids:

$$\{\mathfrak{X}_\mathbf{l} \ : \ \mathbf{l} \in \mathbb{N}^d\} \subset \Omega,$$

with uniform mesh size $h_\mathbf{l} = (h_{l_1}, \ldots, h_{l_d}) = 2^{-\mathbf{l}} := (2^{-l_1}, \ldots, 2^{-l_d})$. Henceforth, we will be using the multi-index $\mathbf{l}$ with non zero components. This is because the use of zero components in $\mathbf{l}$ is not a requirement for the combination technique, in particular, in the context of solving interpolation problem. Hence $\mathfrak{X}_\mathbf{l}$ consists of the points:

$$\mathbf{x}_{\mathbf{l},\mathbf{i}} = (x_{l_1,i_1}, \ldots, x_{l_d,i_d}),$$

with

$$x_{l_j,i_j} = i_j h_{l_j} = i_j 2^{-l_j}, \quad \text{where } 0 \le i_j \le 2^{l_j}, \ 1 \le j \le d.$$

In particular, if $h_{l_i} = 2^{-n}$, $i = 1, \cdots, d$, $\mathfrak{X}_\mathbf{l}$ becomes a uniform full grid of level n, having size N=$(2^n + 1)^d$ and is denoted by X_n. Next we consider the sequence of sub-grids of X_n

$$\{\mathfrak{X}_\mathbf{l} : |\mathbf{l}|_1 = n, \cdots, n + (d-1)\},$$

where n is the level of the grid. $\mathfrak{X}_\mathbf{l}$ is in general anisotropic with its mesh size given by

$$h = 2^{-l} = (2^{-l_1}, \cdots, 2^{-l_d}),$$

and the number of nodes $N^\mathbf{l}$ in $\mathfrak{X}_\mathbf{l}$ is given by

$$N^\mathbf{l} = \prod_{i=1}^{d} (2^{l_i} + 1).$$

As discussed in Chapter 3, the sparse grid $\mathfrak{Y}^s_{n,d}$ of level n in $\Omega \subset \mathbb{R}^d$ is obtained by the superposition of these sub-grids (e.g., Figures 3.6, 3.7).

We want to evaluate the interpolant at the constituent sub-grids $\mathfrak{X}_\mathbf{l}$. Due to the anisotropy of the grids, we use anisotropic RBFs, $\varphi_A(\|\cdot\|)$, of the kind discussed in Chapter 2. The transformation matrix, $A = A_\mathbf{l}$, associated with the basis function $\varphi_A(\|\cdot\|)$ controls its anisotropy. Using $\varphi_A(\|\cdot\|)$, resulting anisotropic RBF interpolation problem is guaranteed to be well-posed for any choice of $A_\mathbf{l}$ as long as it remains non singular. Corresponding to each anisotropic grid $\mathfrak{X}_\mathbf{l}$, $A_\mathbf{l}$ is specially constructed so that it scales (stretches or contracts) the level sets of the anisotropic RBF in order to counteract the anisotropy of the grid $\mathfrak{X}_\mathbf{l}$. In other words, the anisotropic grids $\mathfrak{X}_\mathbf{l}$ and the transformation matrix $A_\mathbf{l}$ should be correlated in the sense that the two anisotropies cancel each other effect to an optimal extent. For the purpose of using the combination formula in the implementations of our algorithm, for the choice of each small grid $\mathfrak{X}_\mathbf{l}$: $\mathbf{l} = (l_1, \ldots, l_d)$ in the sequence of the anisotropic full grids, we construct the transformation matrix $A_\mathbf{l} \in \mathbb{R}^{d\times d}$ to be a diagonal matrix with its main diagonal given by $(2^{l_1}, \ldots, 2^{l_d})$, i.e.,

$$\begin{pmatrix} 2^{l_1} & 0 & \cdots & 0 \\ 0 & 2^{l_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 2^{l_d} \end{pmatrix}$$

To this end, the *anisotropic RBF interpolant* $S_{A_\mathbf{l}}(\mathbf{x})$ corresponding to the sub-data on the sub-grid $\mathfrak{X}_\mathbf{l}$ is given by.

$$S_{A_\mathbf{l}}(\mathbf{x}) = \sum_{j=1}^{N^\mathbf{l}} c_j \varphi(\|A_\mathbf{l}(\mathbf{x} - \mathbf{x}_j)\|) + \sum_{k=1}^{M} d_k p_k(A_\mathbf{l}\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{4.2}$$

and it is such that

$$S_{A_\mathbf{l}}|_{\mathfrak{X}_\mathbf{l}} = u|_{\mathfrak{X}_\mathbf{l}}.$$

The sub-grid interpolants $S_{A_\mathbf{l}}(\mathbf{x})$ are then linearly combined according to the technique [54], [44] given by the following formula to get the sparse kernel-based interpolant $S^c_n(\cdot)$ of level n. More specifically the *Sparse Kerenel-based Interpolant* is defined by

$$S^c_n(\mathbf{x}) = \sum_{q=0}^{d-1} (-1)^q \begin{pmatrix} d-1 \\ q \end{pmatrix} \sum_{|\mathbf{l}|_1 = n+(d-1)-q} S_{A_\mathbf{l}}(\mathbf{x}). \tag{4.3}$$

The combination formula (4.3) can also be found in the earlier work of Delvos [25], where it was used in the context of Lagrange interpolation.

We summarize the sparse kernel-based interpolation method in the following algorithm.

Algorithm 4.1 (Sparse kernel-based interpolation algorithm)

Input: *Sparse grid data,* $\{(\mathbf{x}_i, u_i),\ u_i = u(\mathbf{x}_i),\ \text{and}\ \mathbf{x}_i \in \Omega \subset \mathbb{R}^d\},\ \text{for}\ i = 1, \cdots, N$.

1: Obtain the constituent anisotropic sub-grids

1a: Construct a sequence of indices $\mathfrak{S}_n$:

$$\mathfrak{S}_n = \left\{\mathbf{l} = (l_1, \cdots, l_d) \in \mathbb{N}^d : |\mathbf{l}|_1 = n, \cdots, n + (d-1)\right\}.$$

1b: sequence of sub-grids $\mathfrak{A}_n$: *Construct the sequence of anisotropic sub-grids of* $X_\mathbf{n}$

$$\mathfrak{A}_n = \{\mathfrak{X}_\mathbf{l} : \mathbf{l} \in \mathfrak{S}_n\}.$$

such that $\mathfrak{X}_\mathbf{l}$ *has mesh size given by*

$$h = 2^{-l} = (2^{-l_1}, \cdots, 2^{-l_d}).$$

and the superposition of these sub grids is the sparse grid $\mathfrak{Y}^s_{n,d}$.

2: Iterpolate the sub problems *Evaluate the anisotropic interpolant* $S_{A_\mathbf{l}}(\cdot)$ *on each* $\mathfrak{X}_\mathbf{l} \in \mathfrak{A}_n$.

3: Combine the partial interpolants, $S_{A_\mathbf{l}}$ **:** *The interpolant* $S^c_n(\cdot)$ *on the sparse grid* $\mathfrak{Y}^s_{n,d}$ *of level* n *is given by the formula:*

$$S^c_n(\cdot) = \sum_{q=0}^{d-1} (-1)^q \begin{pmatrix} d-1 \\ q \end{pmatrix} \sum_{|\mathbf{l}|_1 = n+(d-1)-q} S_{A_\mathbf{l}}(\cdot).$$

Output: *The interpolant* S^c_n *on* $\mathfrak{Y}^s_{n,d}$, *which approximates the sparse grid data.*

The proposed scheme uses a dimension-wise multilevel decomposition of interpolation data sites in conjunction with the application of kernel-based interpolants $S_{A_\mathbf{l}}(\cdot)$ with different scaling in each direction. The new SKI algorithm can be viewed as an extension of the idea of sparse grids/hyperbolic crosses to kernel-based functions. The SKI approximant S^c_n, being combination of ARBF-interpolants, makes the implementation of the algorithm on a computer quite straightforward: an existing RBF computer code with relatively small modification can be used. We feel that ill-conditioning problem of the naïve RBF approach still remains. However, due to the grided configuration of the sub-problems, this instability is somewhat mild and can be circumvented by choosing suitable scaling of the anisotropic kernel used in the algorithm. The stability of SKI will be be discussed separately in Section 4.2. Due to the d-dimensional complexity, the size of the largest naïve RBF interpolation problem which can be solved on a given computer stays the same regardless of the dimension, while SKI has the advantage of nearly one-dimensional complexity. Hence the capability of SKI of addressing the complexity may be more attractive in higher dimension. Moreover, the parallel nature of SKI for solving independent sub-problems together with the dimensional insensitivity of RBFs makes the two seemingly different approaches attractive for large and/or high dimensional problems. SKI algorithm having the properties of a less expensive interpolation method, could provide prominent computational savings in practice. Since the SKI method is very suitable to be implemented in parallel, it could prove to be more effective for high dimensional discretisation specially when implemented in parallel on high precession modern computing systems.

4.2 Stability of SKI

The numerical stability of SKI method can be measured by the maximum condition number of the constituent partial interpolation problems. The error estimates of the RBF interpolation are usually expressed in terms of the fill distance $h_{X,\Omega}$ of the data sites X in its domain Ω [94], [31]. The fill distance $h_{X,\Omega}$ itself is not a direct measure of the instability, because only two close enough points in X can produce a badly conditioned system irrespective of $h_{X,\Omega}$ being quite big. For a positive definite matrix, the condition number is given by, $\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}}$. The upper bound for the maximum eigenvalue λ_{max}, given in [31] in terms of the problem size N is $\lambda_{max} \leq N\varphi(0)$, where φ is a positive definite RBF. Hence for a uniform grid X, λ_{max} grows proportional to $h_{X,\Omega}^{-d}$ and hence λ_{max} is not the main cause of the badly conditioned interpolation system. On the other hand, the lower bound of minimum eigenvalue λ_{min} decreases exponentially with the separation distance q_X of the data set in the given domain Ω and also changes with the support radius in the case of CSRBFs or the shape parameter in the case of GSRBFs [94], [31]. A list of Lower bounds for λ_{min} for a range of RBFs in terms of the separation distance q_X can be found in Table 12.1 in [94]. So the problem size, the separation distance and the shape parameter (which adjust the support) of the RBF cause the growth in the condition number. Among the mentioned causes, shape parameter (or the support radius) of the RBF and q_X need more attention, while the problem size is of secondary importance. In [56], [41], [15], [17], [16], [68], the authors suggest several ways for choosing the shape parameter. The dependence of the stability on the shape has been addressed in [40], [39], [32], [38], where stable algorithms for RBF interpolation with small shape parameters have been proposed. In [79], an algorithm called, "leave-one-out" cross validation, for choosing a good shape parameter was proposed by Rippa. According to this algorithm, the optimal shape parameter is chosen by minimizing the least squares error of the fit by removing part of the data and then comparing with the known values at the removed data points. This is done by finding interpolants to the data by leaving one data out, and hence the dependence of the error on the data function is also taken into account in Rippa's algorithm.

In SKI, we first adjust the support of the ARBF $\varphi_{A_{\mathfrak{l}}}(\cdot) = \varphi(\|A_{\mathfrak{l}}(\cdot - \mathbf{x}_j)\|)$ used in the partial interpolant $S_{A_{\mathfrak{l}}}(\cdot)$ by choosing $A_{\mathfrak{l}}$ according to the anisotropy of the constituent anisotropic sub-grid $\mathfrak{X}_{\mathfrak{l}}$. The anisotropic interpolant $S_{A_{\mathfrak{l}}}(\cdot)$, discussed in Section 2.4, is equivalent to the standard RBF interpolant on the equally spaced grid $A_{\mathfrak{l}}\mathfrak{X}_{\mathfrak{l}}$, provided we choose the transformation matrix $A_{\mathfrak{l}} = \mathrm{diag}(2^{\mathfrak{l}})$ corresponding to anisotropic sub-grid $\mathfrak{X}_{\mathfrak{l}}$. Hence, the ratio of the fill distance $h_{A_{\mathfrak{l}}\mathfrak{X}_{\mathfrak{l}},\Omega}$ to the separation distance $q_{A_{\mathfrak{l}}\mathfrak{X}_{\mathfrak{l}}}$ is $\sqrt{2}$. Eventually, for the shape dependent RBF such as Gaussian, MQ, GIMQ etc we also need to adjust its shape parameter c. This is done by choosing

$$c = \frac{2^{h_r}}{K_d},$$

where,

$$h_r = \frac{h_{\mathfrak{Y}^s_{n+1,d},\Omega}}{h_{\mathfrak{Y}^s_{n,d},\Omega}}, \text{and } K_d \text{ is a constant,}$$

for SKI on level n. For example, for the sparse grids of level 1 to 10 and $d = 2$, the numbers h_r up to 4 places of decimal are given in the following array: $h_r = \{0.707, 0.5872, 0.8514, 0.5872, 0.8514, 0.5872, 0.8514, 0.5872, 0.8514, 0.5872\}$. With these parameters, the condition number do grow relatively slowly, thus suggesting that the method is non stationery. The choice $K_d = 1$, produces nearly ideal condition numbers ($\mathcal{O}(1)$) but with resulting low accuracy, while the choice $K_d = 3$, produces safe condition numbers ($\leq 10^{10}$) and admit good convergence. For $K_d \geq 3$, condition number tend to become bigger and result in unstable computations. In our experiments, we find that a suitably chosen constant value for c can also ensure stable implementation of SKI. For example, $c = 0.45$ for Gaussian, $c = 0.4$ for MQ, $c = 0.21$ for GIMQ, $c = 0.25$ for IMQ and $c = 0.25$ for IQ results in stable computations with a slowly growing condition number in the range of our experiments.

4.3 Numerical Experiments for d=2

A list of of 2D test functions, we have used for generating data to be interpolated in the numerical experiments to follow is given below.

- **Franke2D:**

$$
\begin{aligned}
F_1(x_1,x_2) &= \frac{3}{4}e^{(-(9x_1-2)^2-(9x_2-2)^2} \\
&+\frac{3}{4}e^{-((9x_1+1)^2)/49-((9x_2+1)^2)/10} \\
&+\frac{1}{2}e^{-((9x_1-7)^2)/4-(9x_2-3)^2} \\
&-\frac{1}{5}e^{-((9x_1-4)^2)/4-(9x_2-7)^2}.
\end{aligned}
$$

- **Test1-2D:** $G_1(x_1,x_2) = 4^2 x_1(1-x_1)x_2(1-x_2)$.
- **Sinc2D**: $H(x_1,x_2) = \prod_{i=1}^{2} \frac{sin(\pi x_i)}{\pi x_i}$.
- **Test2-2D:** $G_2(x_1,x_2) = (r^2+r^4)\log r,\ r = \sqrt{x_1^2+x_2^2}$.
- **Test3-2D:** $G_3(x_1,x_2) = \frac{1}{2}x_2 \cos^4\left(4(x_1^2+x_2-1)\right)$.
- **Test4-2D:** $G_4(x_1,x_2) = \sqrt{\frac{18}{\pi}}e^{-(x_1^2+81x_2^2)}$.

The function Franke2D is generally known as Franke's test function and can be found in RBF literature, e.g.[94]. Test1-dD and Sinc$d$D for any $d \in \mathbb{N}$ can be found in [31]. Test2-2D is sum of the polyharmonic splines $r^2 \log r$ and $r^4 \log r$. Test3-2D and Test4-2D can be found in [20] and [6] respectively. These function are exemplified in Figures 4.1.

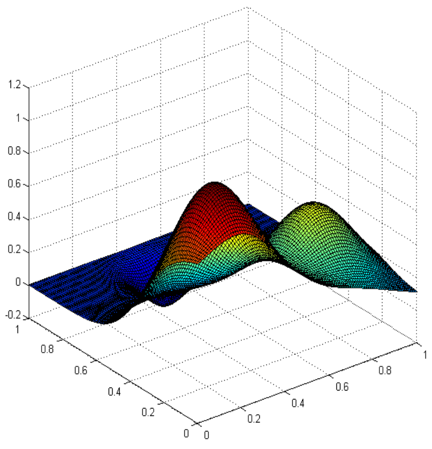
In all our numerical experiments, SKI interpolants are evaluated at either 160×160 equally spaced points or 25,600 Halton points (these are uniformly distributed random points) in the the domain $[0,1]^2 \subset \mathbb{R}^2$. We evaluate on the Halton points, because the data is on a grid. In our experiments, we have implemented the naïve RBF approach to solve the partial interpolation problems in the SKI scheme. Thus, it is fair to compare our method with the naïve RBF interpolation method, as we shall do through out all the experiments reported in this thesis. We remark that the fast RBF methods mentioned above can be applied to accelerate SKI. This is also worth mentioning that we compare SKI on sparse grids with naïve RBF on full grid. This is because the convergence of the naïve RBF approach on sparse grids is not very encouraging in our experience.

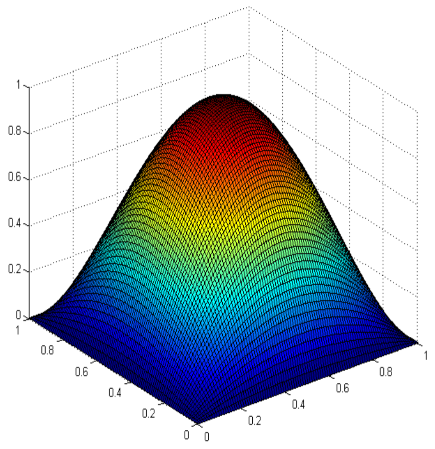
Recalling the dependence of relative error on the condition number described in (2.7), we define the condition numbers is *safe*, when it is $\leq 10^{10}$ and we define the condition numbers is large if it is $\geq 10^{10}$. In the RBF interpolation, we choose the shape parameters proportional to the data density to so that the stability can be ensured. If "N" is the size of a full grid of level $n \in \mathbb{N}$ in the bounded domain $[0,1]^d$, we have $n = N^{1/d} - 1$. Henceforth, we use $c = C \times 2^n$ with a suitably chosen constant C for RBF and MLRBF interplation. We use a range of shape parameters , resulting in *safe* condition numbers as well as in large condition numbers. In the numerical examples, "L_∞-err" is the maximum modulus error, i.e., $\|u - \tilde{u}\|_\infty$, "RMS-error" is the standard root mean squared (RMS) error, i.e., $\sqrt{\frac{\sum_{k=1}^{Neval} |u_k - \tilde{u}_k|^2}{Neval}}$, where u is the exact solution, $\tilde{u}$ is the approximate solution and "$Neval$" is the size of the evaluation grid.

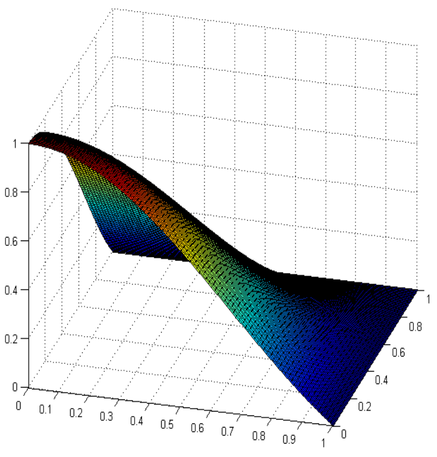
In SKI results, "SGnodes" denote the number of SKI nodes used, "DOFs(SG)" denotes the total number of times the SGnodes are visited (as some of them are re-visited several times) and "Cond no" stands for the maximum 2-norm condition numbers among the interpolation sub-problems required for combination to evaluate the SKI interpolant $S_A^c(\cdot)$. "N" stands for the number of centers used in RBF interpolation. "Err at nodes" is the L_∞-error at the RBF centres (in the case of RBF interpolation) or it is the L_∞-error at the *SGnodes* (in the case of SKI interpolation). We evaluate "Err at nodes" in order to check whether it is zero (as it should be zero in a stable interpolation) or not.

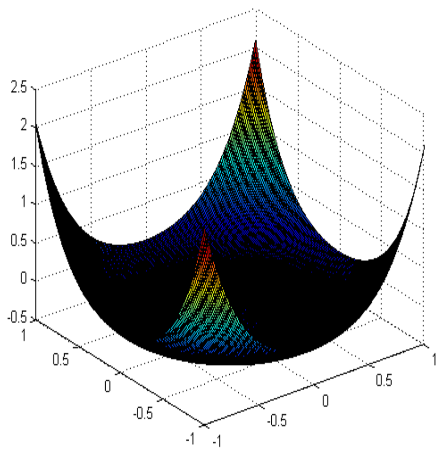
The *Sparse Kernel-based Interpolation* can be implemented by relatively easy modification to an existing RBF code. For the choice of each sub-grid $\mathfrak{X}_{\mathbf{l}} : \mathbf{l} = (l_1, \ldots, l_d)$, we construct the transformation matrix $A_{\mathbf{l}} \in \mathbb{R}^{d \times d}$ to be a diagonal matrix with its main diagonal given by $(2^{l_1}, \cdots, 2^{l_d}) \in \mathbb{R}^d$. While choices for chosing c in SKI have been discussed in Section 4.2.

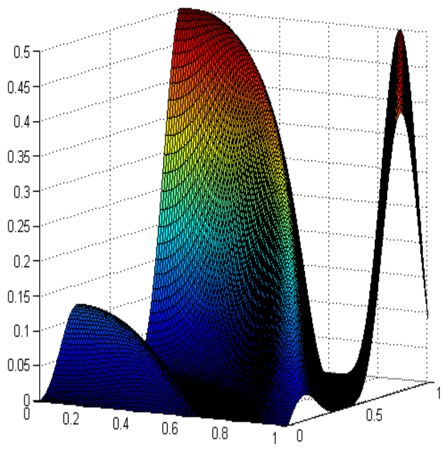
We perform extensive numerical experiments for d=2, by interpolating data generated from the following test functions, Test1-2D, Test2-2D, Test3-2D, Test4-2D, Franke2D and Sinc2D. Note that the test function Test2-2D is not smooth. In these experiments SKI has been implemented with a wide range of RBFs. The implementation of the method in all our experiments, on ALICE as well as on an ordinary computer, has been done in *Matlab* and in serial. We use a desktop computer "Core 2 Duo CPU @ 3.16GHz 3.17GHz and 3.24GB of RAM" for d=2, 3. For d=4, we use ALICE to be able to run many experiments at the same time by accessing as many nodes of ALICE. Each experiment was run in serial on a single node (having a pair of quad-core 2.67GHz Intel Xeon X5550 CPUs and 12GB of RAM) which is equivalent to a more powerful desktop computer. Thus all the CPU timing and the problem size shown here relate to computations that can be run on an ordinary computer mentioned above. Moreover, we are going to report separately the largest size of the problem that SKI can solve on a single node of ALICE for d =2, 3, 4 in Sections 4.3.2, 5.4.1.1(a), and in Tables 6.1, 6.2, 6.3.


(a) Franke2D


(b) Test1-2D


(c) Sinc2D


(d) Test2-2D


(e) Test3-2D

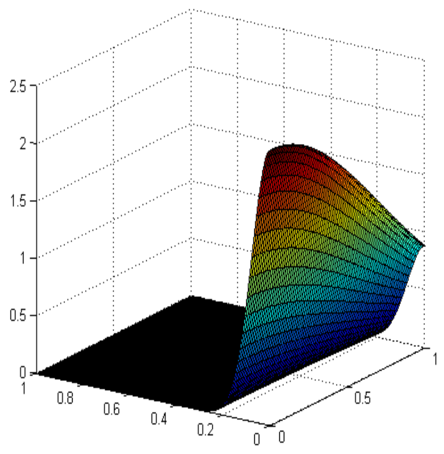

(f) Test4-2D

Figure 4.1: Test functions

4.3.1 Examples with Gaussian RBF

As discussed in Chapter 3, the sparse grids were originally created based on a hierarchical basis and sparse tensor product construction of multi-linear basis functions. Among the RBFs, the Gaussian has tensor product structure as it can be written as

$$e^{-c^2\|\mathbf{x}_i-\mathbf{x}_j\|^2} = e^{-c^2|x_1^i-x_1^j|^2}e^{-c^2|x_2^i-x_2^j|^2}\cdots e^{-c^2|x_d^i-x_d^j|^2},$$

where

$$\mathbf{x}_i = (x_1^i, x_2^i, \cdots, x_d^i) \in \mathbb{R}^d.$$

Motivated by its tensor-product nature, we first implement the sparse kernel-based interpolation with Gaussians.

Examples with safe condition number

Data from some of our experiments with safe condition numbers is given in Tables 4.1, 4.3 for sparse kernel based interpolation and in Tables 4.2, 4.4 for classical full grid RBF interpolation, while more details are give in the figures. It is observed that, as for as the condition number stays safe, the error at the sparse grid nodes is nearly the machine zero and this is in complete agreement with a similar behaviour observed for the classical RBFs interpolation. This observation is given as the "Err at nodes" in Tables 4.1, 4.3, 4.2, 4.4. This confirms that we are not losing any accuracy and one should expect reliable computation.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	6.2203e-1	1.8260e-1	1.1727e-15	2.6912e+3	9.6437e-4
21	39	3.3070e-1	7.5595e-2	3.3973e-14	2.5325e+4	2.6042e-3
490	109	1.1070e-1	3.8513e-2	4.0863e-13	2.8184e+5	5.1803e-3
113	271	4.0603e-2	1.1233e-2	8.6423e-13	2.6522e+6	9.2529e-3
257	641	1.3883e-2	2.6363e-3	1.4270e-12	2.9516e+7	2.2223e-2
577	1475	8.1981e-3	5.1787e-4	1.0622e-12	1.7591e+8	1.0507e-1
1281	3333	2.9843e-3	1.3879e-4	1.1879e-13	1.0484e+9	5.3998e-1
2817	7431	2.1240e-3	6.6385e-5	4.1966e-14	2.3229e+9	3.5107e+0
6145	16393	5.5459e-4	3.0491e-5	5.1847e-14	5.1468e+9	3.4347e+1
13313	35851	2.5892e-4	7.3961e-6	5.2847e-14	6.5016e+9	3.1038e+2

Table 4.1: SKI results using Gaussian RBF with shape parameter $c = 0.45$ for each level, test function Franke2D, on 160×160 equally spaced evaluation grid.

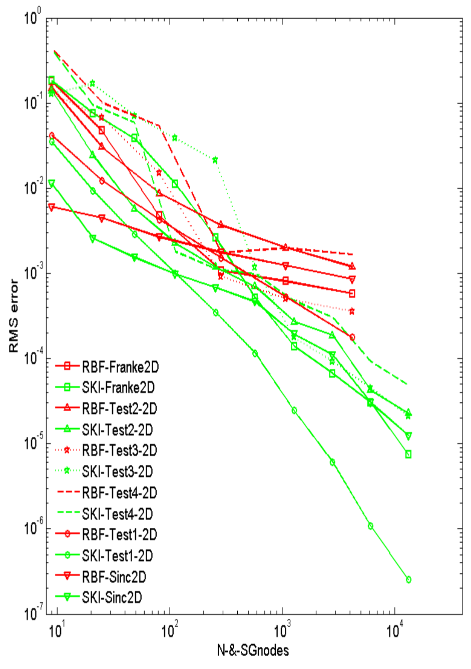
The SKI algorithm is faster than direct RBF-method, we observe that the time taken by SKI is smaller than that taken by standard RBF interpolation for the same number of nodes used. The run time of SKI remains smaller even if the re-visits of the algorithm to the same node (DOFs(SG)) are taken in to account. The plots describing this observation about run time in terms of the number of nodes used is omitted for brevity. We present error in terms of the number of nodes used as well as in terms of run time. In Figures 4.2(c), 4.2(d), RMS-error is plotted versus the machine time. We express the error in terms of the size (N or SGnodes) of the input data in Figures 4.2(a), 4.2(b), In either case performance of the new proposed algorithm is superior than the direct RBF method.

N	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	6.1200e-1	1.7975e-1	4.1257e-16	1.1687e+3	7.5539e-2
25	1.5084e-1	4.7240e-2	6.8737e-14	4.9415e+4	4.2162e-3
81	2.5120e-2	4.7206e-3	4.7240e-14	2.0309e+6	2.0519e-3
289	1.3917e-2	1.0754e-3	1.4433e-15	2.3576e+7	3.3288e-2
1089	1.4517e-2	8.0590e-4	2.4425e-15	6.3673e+7	1.4637e+0
4225	1.4740e-2	5.7989e-4	1.9984e-15	8.4800e+7	1.4658e+2

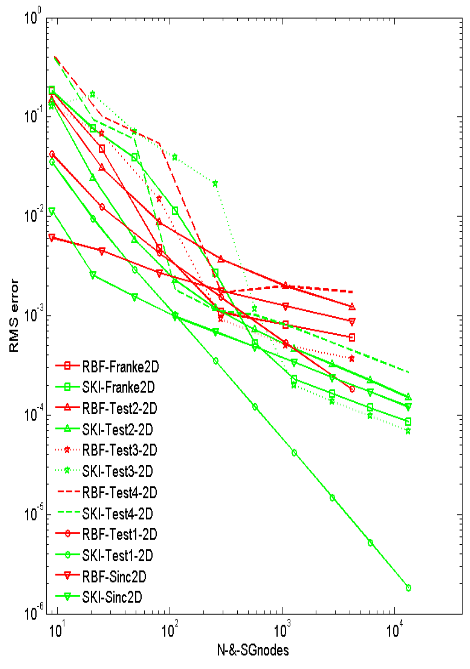
Table 4.2: RBF interpolation results using Gaussian RBF with shape parameter $c = 0.5 \times 2^n$, test function Franke2D, on equally spaced 160×160 evaluation grid.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	6.2215e-1	1.8363e-1	1.1727e-15	2.6912e+3	1.0851e-3
21	39	3.3061e-1	7.6066e-2	3.3973e-14	2.5325e+4	2.6534e-3
49	109	1.1072e-1	3.8767e-2	4.0863e-13	2.8184e+5	5.1141e-3
113	271	4.0627e-2	1.1304e-2	8.6423e-13	2.6522e+6	9.6881e-3
257	641	1.3900e-2	2.6530e-3	1.4270e-12	2.9516e+7	2.3347e-2
577	1475	7.6365e-3	5.2642e-4	1.0622e-12	1.7591e+8	1.0289e-1
1281	3333	5.7691e-3	2.2933e-4	1.1879e-13	1.0484e+9	5.2904e-1
2817	7431	4.4410e-3	1.6392e-4	4.1966e-14	2.3229e+9	3.5082e+0
6145	16393	4.3974e-3	1.1790e-4	5.1847e-14	5.1468e+9	3.4310e+1
13313	35851	4.4684e-3	8.5654e-5	5.2847e-14	6.5016e+9	3.0990e+2

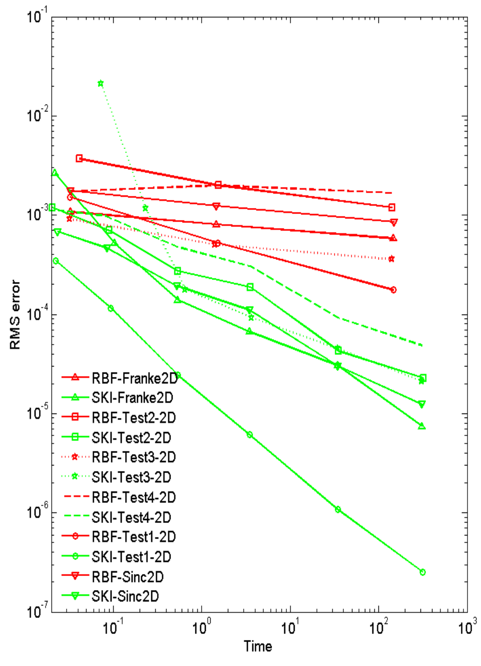
Table 4.3: SKI results using Gaussian RBF with shape parameter $c = 0.45$ for each level, test function Franke2D, evaluated at $25,600$ Halton points.



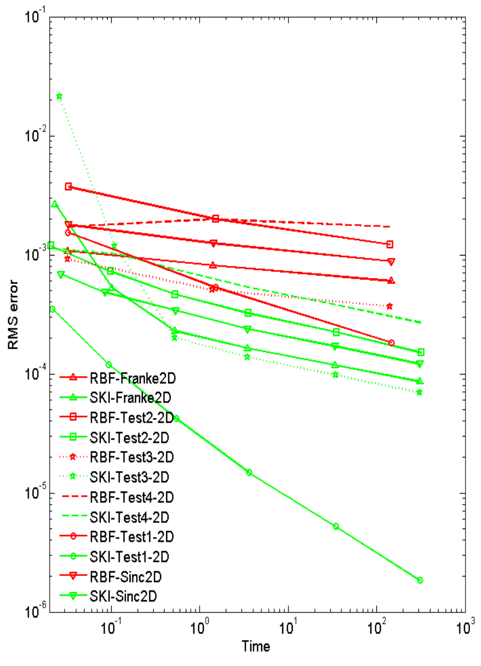
(a) On equally spaced evaluation grid



(b) On Halton points evaluation grid



(c) On equally spaced evaluation grid



(d) On Halton points evaluation grid

Figure 4.2: RMS-error versus N(for RBFs) and SGnodes (Top row) and RMS-error versus CPU-Time (Bottom row): Gaussian SKI (Green), RBF interpolation (Red) with safe condition numbers

N	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	6.1207e-1	1.8077e-1	4.1257e-16	1.1687e+3	6.6746e-2
25	1.5070e-1	4.7532e-2	6.8737e-14	4.9415e+4	4.1533e-3
81	2.5128e-2	4.7504e-3	4.7240e-14	2.0309e+6	2.2908e-3
289	1.4005e-2	1.0796e-3	1.4433e-15	2.3576e+7	3.2607e-2
1089	1.4103e-2	8.1099e-4	2.4425e-15	6.3673e+7	1.4265e+0
4225	1.3740e-2	5.9901e-4	1.9984e-15	8.4800e+7	1.4649e+2

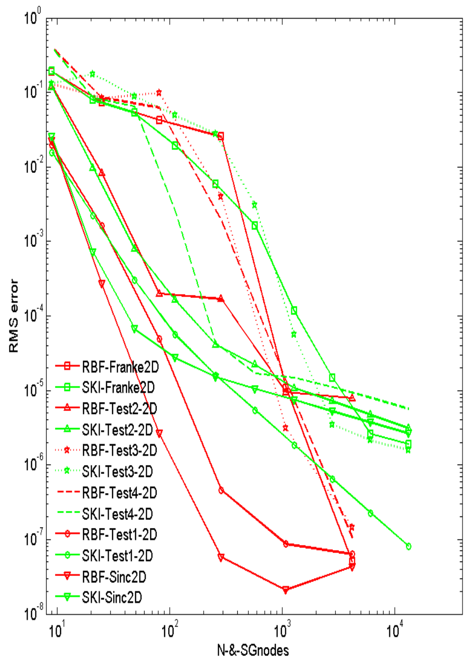
Table 4.4: RBF interpolation results using Gaussian RBF with shape parameter $c = 0.5 \times 2^n$, test function Franke2D, evaluated at $25,600$ Halton points.

Examples with large condition numbers

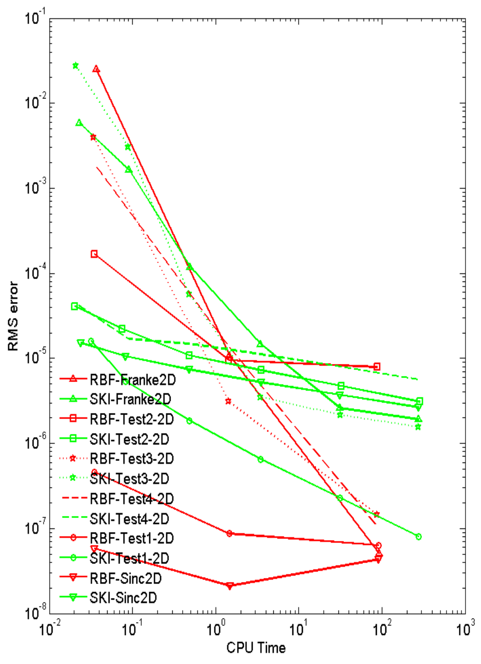
We continue our experiments with Gaussian SKI with shape parameters resulting the interpolation system to be badly conditioned. We observe that the convergence is faster in both cases and direct RBF shows slightly better convergence than SKI as shown in Figure 4.3(a). But in this case, error at the nodes in Tables 4.6, 4.5 and Figures 4.3(c), 4.3(d) is not machine procession and it looks we are losing some accuracy, this makes the experiment with large condition number unreliable. This visible loss of accuracy at the nodes is observed in the RBF as well as sparse kernel-based interpolation.

N	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	2.9730e-2	1.9785e-2	5.6843e-13	3.4263e+4	1.2393e-2
25	3.2376e-3	1.6338e-3	1.7280e-11	8.4607e+8	4.4140e-4
81	1.4367e-4	4.9394e-5	7.8883e-7	9.0547e+17	2.3645e-3
289	2.7305e-6	4.5956e-7	1.0937e-7	8.0684e+20	3.4577e-2
1089	1.0011e-6	8.6589e-8	1.3720e-7	4.3148e+20	1.4688e+0
4225	2.1251e-7	6.3437e-8	2.0267e-7	8.2015e+20	9.0569e+1

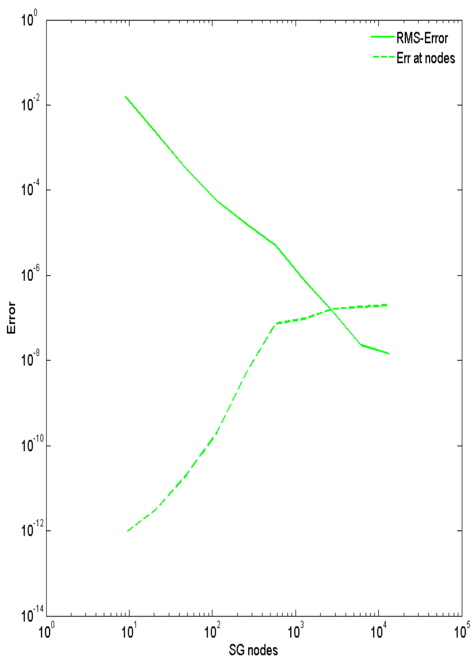
Table 4.5: RBF interpolation results using Gaussian RBF with shape parameter c=$0.4 \times 2^{((5/7)n)}$, test function Test1-2D, evaluated at $25,600$ Halton points.



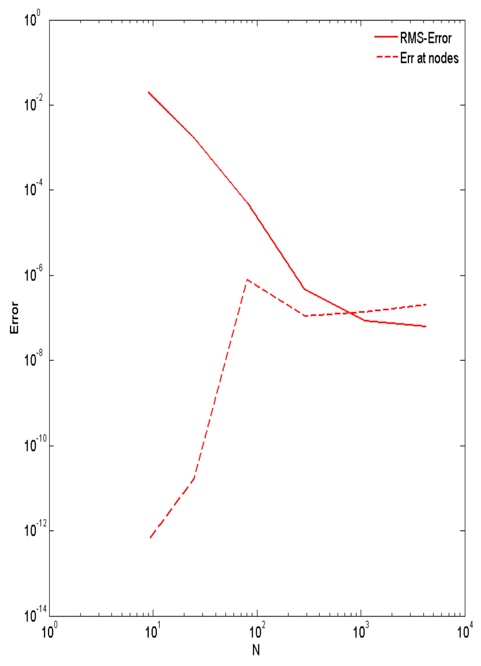
(a) RMS-error versus N (for RBFs) and SGnodes(for SKI): on Halton points evaluation grid.



(b) RMS-error versus CPU-Time: Halton points evaluation grid.



(c) On equally spaced evaluation grid



(d) On equally spaced evaluation grid

Figure 4.3: Gaussian SKI (Green) RBF interpolation (Red) with large condition numbers.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	2.3577e-2	1.5683e-2	9.0949e-13	9.3266e+4	1.0398e-3
21	39	4.2231e-3	2.2162e-3	3.1264e-12	4.8524e+6	2.5906e-3
49	109	8.1732e-4	3.0411e-4	2.0634e-11	1.0647e+9	5.0333e-3
113	271	2.1732e-4	5.5887e-5	1.9524e-10	4.5852e+11	9.3272e-3
257	641	8.9804e-5	1.5989e-5	5.2691e-9	5.4137e+13	3.1796e-2
577	1475	4.3217e-5	5.4059e-6	7.3453e-8	6.1997e+15	8.2770e-2
1281	3333	2.0993e-5	1.8532e-6	9.4408e-8	1.8098e+17	4.8576e-1
2817	7431	1.0249e-5	6.5180e-7	1.6140e-7	9.3641e+17	3.4876e+0
6145	16393	5.1016e-6	2.2785e-7	1.7983e-7	1.3552e+19	3.1338e+1
13313	35851	2.4993e-6	8.0844e-8	1.9637e-7	1.1346e+20	2.7635e+2

Table 4.6: SKI results using Gaussian RBF with shape parameter c=0.29 for each level, test function Test1-2D, evaluated at $25,600$ Halton points.

4.3.2 SKI simulations on ALICE, a computer with more RAM

All these experiments have been performed on an ordinary computer having specification “Intel Core 2 Duo CPU @ 3.16GHz 3.17GHz and 3.24GB of RAM”. It is observed that the computation is stable for large data on a sparse grid up to level 10. The corresponding full grid of level 10 has 1,050,625 nodes. We have also performed some of our experiments on a single computer node (having a pair of quad-core 2.67GHz Intel Xeon X5550 CPUs and 12GB of RAM) of the computer cluster ALICE of the University of Leicester. A computer node on ALICE is equivalent to a more power full desktop. We find that the method is stable for larger size of the data and computations can be done on a sparse grid of levels up to 12 having more than 61,000 nodes. The corresponding full grid of level 12 has 16,785,409 nodes. So this mean that our method can stably approximate a 2D data of large size such as more than 16.7 million on a full grid. These high level numerical results are given in Table 4.7. The largest size of direct RBF interpolation problem that we can solve using the same high performance computing system ALICE is given in Table 4.8. This size is nearly the same as the SKI can solve on an ordinary computer. Due to its nearly one-dimensional complexity, SKI can solve even larger interpolation problems in higher dimensions while the maximum size of classical RBF methods stays the same independent of the dimension. High dimensional performance of SKI is demonstrated by examples in Chapter 6.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	6.2215e-1	1.8363e-1	9.7838e-16	2.6912e+3	1.0561e+0
21	39	3.3061e-1	7.6066e-2	1.6834e-14	2.5325e+4	9.3820e-3
49	109	1.1072e-1	3.8767e-2	3.5380e-13	2.8184e+5	4.2680e-3
113	271	4.0627e-2	1.1304e-2	6.3793e-13	2.6522e+6	7.0460e-3
257	641	1.3900e-2	2.6530e-3	1.5116e-12	2.9516e+7	6.0849e-2
577	1475	7.6365e-3	5.2642e-4	8.4097e-13	1.7591e+8	7.0136e-2
1281	3333	5.7691e-3	2.2933e-4	1.5285e-13	1.0484e+9	4.8372e-1
2817	7431	4.4410e-3	1.6392e-4	4.1855e-14	2.3229e+9	3.3381e+0
6145	16393	4.3974e-3	1.1790e-4	4.1189e-14	5.1468e+9	2.8298e+1
13313	35851	4.4684e-3	8.5654e-5	4.6629e-14	6.5016e+9	2.5568e+2
28673	77837	4.4437e-3	6.0948e-5	4.9405e-14	8.2129e+9	2.1127e+3
61441	167951	4.2062e-3	4.3058e-5	4.8628e-14	8.7056e+9	1.8958e+4

Table 4.7: SKI results from ALICE using Gaussian RBF with shape parameter c=0.45 for each level, test function Franke2D, evaluated at 25, 600 Halton points.

N	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	6.1207e-1	1.8077e-1	5.4123e-16	1.1687e+3	1.4851e-1
25	1.5070e-1	4.7532e-2	7.2498e-14	4.9415e+4	4.5210e-3
81	2.5128e-2	4.7504e-3	5.6469e-14	2.0309e+6	9.1590e-3
289	1.4005e-2	1.0796e-3	1.4433e-15	2.3576e+7	2.6139e-2
1089	1.4103e-2	8.1099e-4	1.7764e-15	6.3673e+7	9.4756e-1
4225	1.3740e-2	5.9901e-4	1.9984e-15	8.4800e+7	1.5712e+2
4900	1.3372e-2	5.7855e-4	2.2204e-15	8.5961e+7	2.4911e+2
6400	1.2524e-2	5.4643e-4	2.4425e-15	8.7686e+7	5.8488e+2
8100	1.2414e-2	5.1773e-4	2.6645e-15	8.8882e+7	1.0851e+3
10000	1.2089e-2	4.9288e-4	2.4425e-15	8.9744e+7	1.9047e+3
11025	1.1861e-2	4.8103e-4	2.2204e-15	9.0087e+7	2.5149e+3
12100	1.1596e-2	4.6971e-4	2.4425e-15	9.0385e+7	3.6044e+3
13225	1.1299e-2	4.5915e-4	2.2204e-15	9.0645e+7	4.1111e+3
14884	1.0842e-2	4.4624e-4	2.1094e-15	9.0957e+7	5.6667e+3

Table 4.8: RBF interpolation results from the HPC ALICE using Gaussian RBF with shape parameter $c = 0.5\times 2^n$, test function Franke2D, evaluated at 25, 600 Halton points.

4.3.3 SKI with non Gaussian RBFs

So far we have presented our sparse kernel-based interpolation results with Gaussian RBF. Gaussians are positive definite and of tensor product nature as we can write

$$e^{-c^2\|\mathbf{x}_i-\mathbf{x}_j\|^2} = e^{-c^2|x_1^i-x_1^j|^2}e^{-c^2|x_2^i-x_2^j|^2}\cdots e^{-c^2|x_d^i-x_d^j|^2},$$

where $\mathbf{x}_i = (x_1^i, x_2^i, \cdots, x_d^i) \in \mathbb{R}^d$. Hence Gaussians can be still viewed in the classical sparse grid settings. Sparse grid methods based on one-dimensional tensor product of standard RBFs (such as, Wendlands, Gaussians, inverse multiquadrics, multiquadrics and thin plate splines) have been considered in [83], where the error analysis has been given based on the resulting tensor-product nature of the basis functions. The resulting methods in [83] were not supported by numerical assessment.

We present our experimental results for positive definite (PD) non Gaussians RBFs and conditionally positive definite (CPD) RBFs separately. We adjust the shape parameter so as to ensure a safe condition numbers together with few values of the shape parameter producing large condition numbers, in order to study the behaviour of SKI in the non-Gaussian context.

4.3.3(a) SKI with non Gaussian PD RBFs

In this section, we present the results from the sparse kernel-based interpolation performed with non Gaussian RBFs as basis function, which are positive definite having order $m = 0$ such as inverse quadric IQ $=\frac{1}{1+c^2r^2}$, inverse multiquadric IMQ $=\frac{1}{\sqrt{1+c^2r^2}}$, generalized inverse multiquadrics GIMQ $=\frac{1}{(1+c^2r^2)^2}$, and Wendland's function, WE32, $\varphi_{3,2}(r) = (1-cr)_+^6(35c^2r^2+18cr+3)$. The method with PD-RBFs basis converges but at a rate slightly slower as compared with that of Gaussian. This is given in Figures 4.4(a), 4.4(b). This agrees with the superior convergence of Gaussian over these basis in the context of standard RBF interpolation. We observe that the convergence of SKI is slower as compared to the standard RBF interpolation when error is compared as function of the size of the input data, but SKI is superior if error is plotted versus time. The slower convergence of SKI as compared to naïve RBF approach indicates the compromise similar to the one observed for sparse grid methods where accuracy deteriorates by a logarithmic factor for the reduction of computational cost. We also give the error comparison for the basis with large conditions numbers in Figures 4.4(c), 4.4(d), where we observed the same slower convergence behaviour of sparse grid interpolation as compared with the classical RBF method.

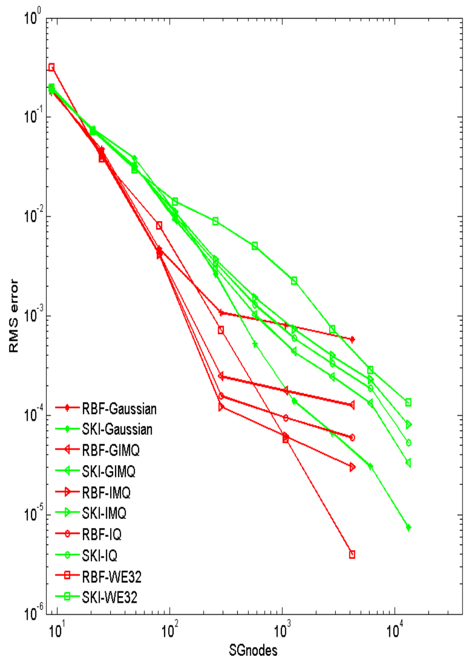
SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	7.0520e-2	4.6335e-2	1.9895e-13	1.3125e+4	5.8174e-1
21	39	2.3644e-2	1.0794e-2	1.3419e-2	9.9156e+4	9.1386e-3
49	109	1.8437e-2	8.5963e-3	1.4801e-2	5.7648e+5	4.6724e-3
113	271	1.0767e-2	4.1806e-3	9.4742e-3	2.4285e+6	8.8500e-3
257	641	5.6691e-3	1.7135e-3	4.9939e-3	8.3427e+6	2.1641e-2
577	1475	2.8008e-3	6.5234e-4	2.4594e-3	1.6121e+7	7.0627e-2
1281	3333	1.1990e-3	2.3879e-4	1.1866e-3	2.9818e+7	4.4303e-1
2817	7431	5.7483e-4	9.2579e-5	5.7126e-4	3.8280e+7	2.5541e+0
6145	16393	2.8581e-4	2.6625e-5	2.7587e-4	4.9297e+7	2.7559e+1
13313	35851	1.5608e-4	9.5637e-6	1.3374e-4	5.4030e+7	2.3579e+2

Table 4.9: SKI, using GIMQ, with $c = 0.21$ for each level, using test function Test1-2D, on 160×160 equally spaced evaluation grid.

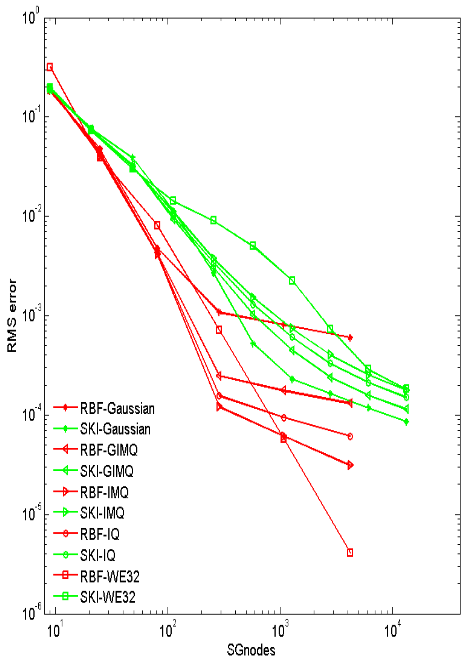
4.3.3(b) SKI with CPD RBFs

We perform some experiments with conditionally positive definite RBFs of order $m > 0$. The most common CPD RBFs are multiquadric MQ $=\sqrt{1+c^2r^2}$ with $m = 1$, thin plate spline TPS2=$r^2log(r)$ with $m = 2$, TPS3=$r^4log(r)$, $m = 3$, radial powers r, $m = 1$, r^3, with $m = 2$ etc. In the case of CPD-RBFs, one need to append a polynomial $p(\cdot) \in \pi^d_{m-1}$ to control the growth and guarantee the recovery of a unique interpolant. The convergence of MQ is nearly the same as that with the non-gaussian PD-RBFs discussed above. Apart from MQ, other CPD-RBFs have shown either very slow convergence or even no convergence at all. The condition number attached with MQ can be adjusted by varying associated shape parameter, while the condition number associated with the RBFs such as TPS2=$r^2log(r)$, TPS3=$r^4log(r)$, r, r^3 having no shape parameter is prohibitively large. The errors with safe condition numbers are given in Figures 4.5(a), 4.5(b), while those with large condition numbers are presented in Figures 4.5(c), 4.5(d).

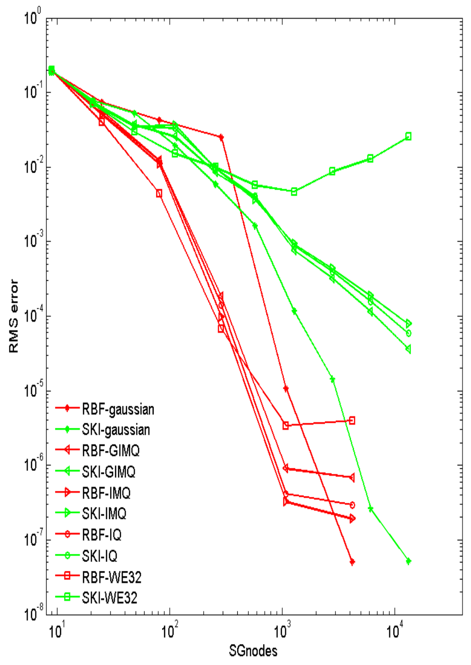
We observed that using non Gaussian RBFs (whether positive definite or conditionally positive definite), the error at the sparse grid nodes is same as the interpolation error in the remaining domain even if the condition number is good enough. This was not the case with Gaussian RBF, where error at the nodes was nearly zero for well-conditioned problems. The errors at the nodes are given in Tables 4.9, 4.10.



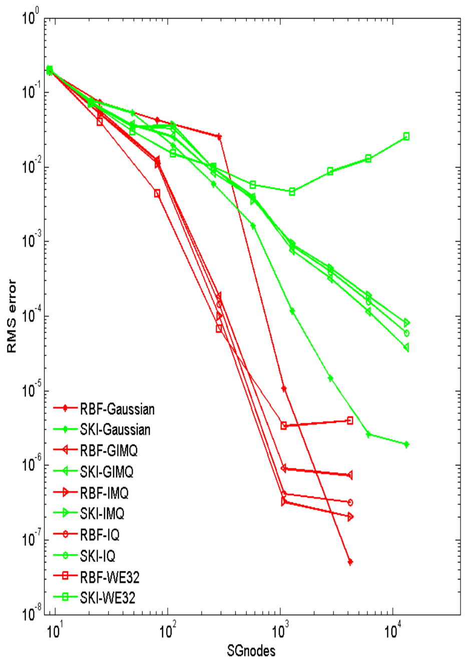
(a) On equally spaced evaluation grid



(b) On uniformly distributed evaluation grid

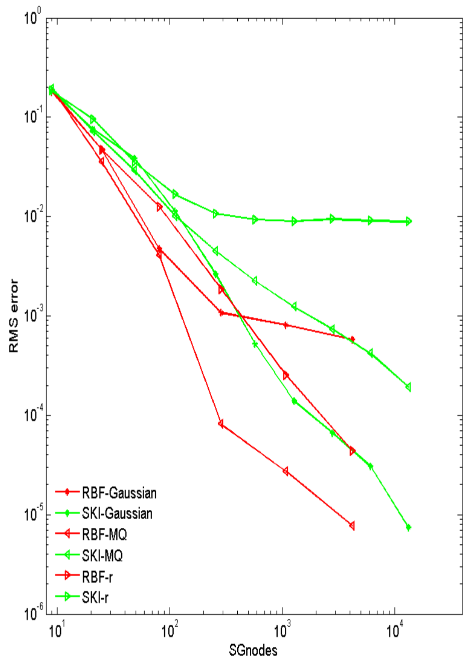


(c) On equally spaced evaluation grid

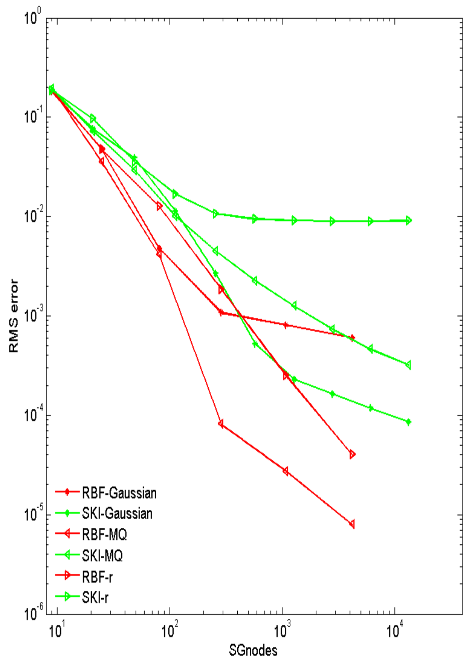


(d) On uniformly distributed evaluation grid

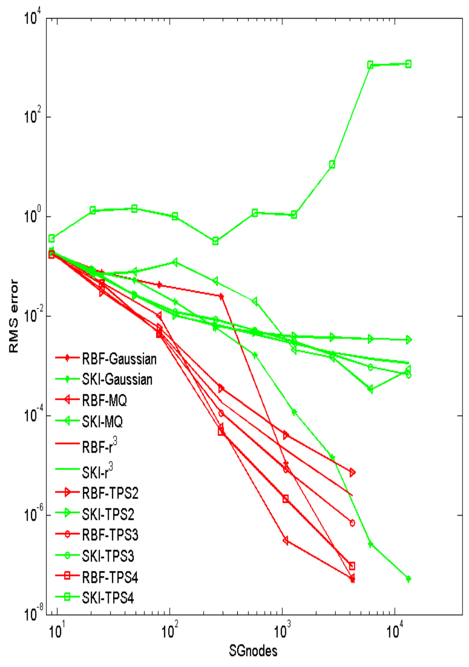
Figure 4.4: RMS-error versus SGnodes (SKI) and N (RBF): SKI(Green) and RBF-full grid (Red) with safe condition numbers (Top row) and with large condition numbers (Bottom row)



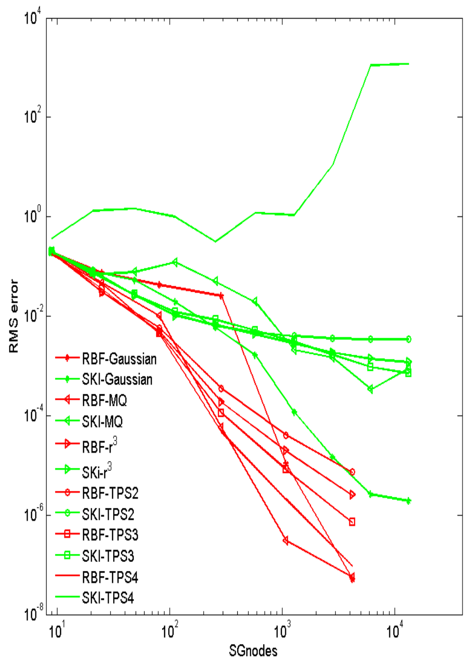
(a) On equally spaced evaluation grid



(b) On uniformly distributed evaluation grid



(c) On equally spaced evaluation grid



(d) On uniformly distributed evaluation grid

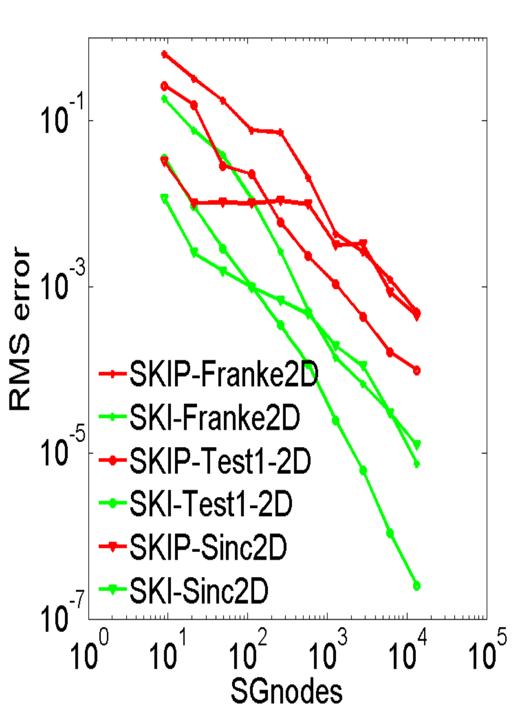
Figure 4.5: RMS-error versus SGnodes (SKI) and N (RBF): SKI(Green) and RBF-full grid (Red) for PD (Non Gaussian) and CPD RBFs, with safe condition numbers (Top row) and with large condition numbers (Bottom row)

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	9.8544e-2	6.5283e-2	2.4514e-13	2.2856e+4	6.1154e-1
21	39	4.4592e-2	1.9846e-2	2.6726e-2	1.1871e+5	1.1555e-2
49	109	3.7102e-2	1.8565e-2	3.2903e-2	5.3530e+5	4.9294e-3
113	271	2.5868e-2	1.0442e-2	2.3143e-2	2.0546e+6	9.1565e-3
257	641	1.4892e-2	5.1211e-3	1.3299e-2	7.6895e+6	2.1519e-2
577	1475	7.8016e-3	2.4188e-3	6.9619e-3	2.9498e+7	8.9453e-2
1281	3333	3.7745e-3	1.1514e-3	3.4523e-3	1.1555e+8	4.0766e-1
2817	7431	2.1540e-3	5.6427e-4	1.6739e-3	4.5767e+8	2.4855e+0
6145	16393	1.5137e-3	2.6993e-4	1.1218e-3	1.8223e+9	2.4499e+1
13313	35851	9.8175e-4	1.3131e-4	7.2981e-4	7.2734e+9	2.1747e+2

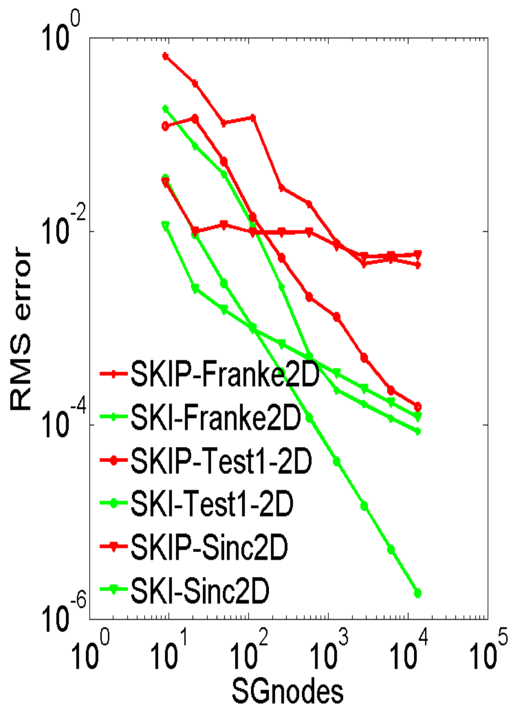
Table 4.10: SKI using MQ, with $c = 0.4$ for each level, using test function Test1-2D, on 160×160 equally spaced evaluation grid.

4.4 SKI on perturbed grids

So far we have discussed the sparse kernel-based interpolation on regular grids. Now this is the time to discuss the behavior of the method on unstructured sparse grids. It is worth mentioning here that the sparse grid methods in the context of multi-linear approximation have been used on regular sparse grids and have been extended to non smooth solutions by adaptive refinement [51], [53], [61], but these methods have not been used on perturbed grids. We perturb the parent uniform full grid by adding random numbers nearly equal to up to a quarter of the fill distance, from which we extract the perturbed sparse grid. The full grid, and the corresponding perturbed full grid, the sparse grid and the corresponding perturbed sparse grid of level 3 are given in Figure 4.7. We perform experiments on several perturbed grids, we present only our experiment on sparse grids perturbed on the same scale as shown in Figure 4.7. As long as we use the shape parameter which keeps the condition number safe, we get convergence which is slightly slower than the regular sparse grid. These results are given in Tables 4.11, 4.12. A comparison between the errors for experiments performed on regular sparse grids and perturbed sparse grids is given in Figures 4.6(a), 4.6(b). On the other hand, in the experiments with shape parameters resulting in large condition numbers, the convergence becomes very slow or even oscillates. Based on our experimental results, we conclude that SKI converges on irregular sparse grids provided a shape parameter resulting in safe condition number is used.



(a) On equally spaced evaluation grid



(b) On uniformly distributed evaluation grid

Figure 4.6: RMS-error versus SGnodes: Regular sparse grid SKI (Green) and and SKI on perturbed sparse grid (Red), using Gaussian with $c = 0.45$ producing safe condition numbers.

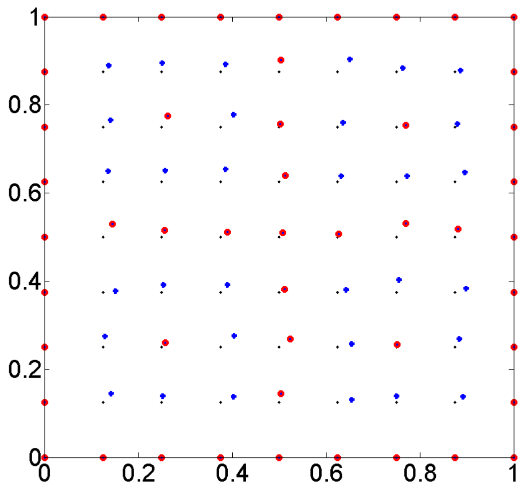


Figure 4.7: Full grid (Black dots), Perturbed full grid (Blue stars) and Perturbed sparse grid (Red small circles) of level 3.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Cond no	Time
9	9	6.3057e-1	1.8681e-1	2.6782e+3	3.0339e-1
21	39	3.2149e-1	7.4121e-2	2.5957e+4	2.1477e-2
49	109	1.7311e-1	6.6810e-2	2.1567e+5	6.7651e-3
113	271	7.6386e-2	2.5482e-2	2.2055e+6	1.1495e-2
257	641	7.1874e-2	1.5784e-2	2.1699e+7	2.9160e-2
577	1475	2.0588e-2	4.6391e-3	1.2844e+8	1.1698e-1
1281	3333	4.3316e-3	9.3708e-4	6.7882e+8	6.5614e-1
2817	7431	2.6099e-3	4.9258e-4	1.6539e+9	4.2401e+0
61450	16393	1.2238e-3	1.9542e-4	3.8533e+9	3.8792e+1
13313	35851	4.9777e-4	8.4162e-5	5.4694e+9	3.2909e+2

Table 4.11: SKI results: sparse grid is perturbed by adding random numbers from $(0, 2^{-level}/4)$, using Gaussian RBF with $c = 0.45$ for each level, test function Franke2D, on 160×160 equally spaced evaluation grid.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Cond no	Time
9	9	6.0269e-1	1.7513e-1	6.1538e+2	1.6220e-3
21	39	2.8433e-1	6.6306e-2	6.5810e+3	3.6460e-3
49	109	1.0168e-1	3.3382e-2	3.7959e+3	6.5749e-3
113	271	7.0602e-2	2.1208e-2	2.8177e+5	1.1905e-2
257	641	3.1135e-2	3.7970e-3	3.4217e+4	2.9880e-2
577	1475	1.2240e-2	1.7385e-3	5.7015e+6	1.3945e-1
1281	3333	1.3272e-2	1.0215e-3	1.1318e+5	7.2855e-1
2817	7431	3.2151e-3	3.4584e-4	3.2640e+7	4.3356e+0
6145	16393	2.9941e-3	2.4442e-4	1.7656e+5	4.1444e+1
13313	35851	6.9303e-4	8.9808e-5	6.7086e+7	3.4659e+2

Table 4.12: SKI results: sparse grid is perturbed by adding random numbers from $(0, 2^{-level}/4)$, using Gaussian RBF with $c = 2^{h_r}/3$, using test function Franke2D, on 160×160 equally spaced evaluation grid.

4.5 Discussion

The proposed algorithm, Sparse Kernel-based Interpolation, is not only stable for large data but also good in efficiency and thus can addresses the complexity issue. The parallel nature of SKI by solving small full grid problems together with the dimensional insensitivity of RBFs makes the two seemingly different approaches attractive for large and/or high dimensional problems. Since the combination is very suitable to be implemented in parallel, so that SKI could prove to be more effective for high dimensional interpolation problems, specially when implemented in parallel on high precession modern computing systems. The method is numerically stable for most RBFs with variable accuracy. It is also observed that the run time decreases in terms of centers used, and SKI can solve large interpolation problems with $N = 61,000$ sparse grid nodes for $d = 2$. The corresponding full grid bi-variate classical RBF interpolation of level 12, would require more than 16 million degrees of freedom. We observe that the convergence is faster or the same as the full grid interpolation and it can be easily implemented for larger data in less time. The size of the largest classical RBF interpolation problem that we can solve on ALICE is nearly $15,000$ regardless of the dimension, while on the same computer SKI can solve a problem with $N = 61,000$ sparse grid nodes for d=2. Due to the d-dimensional complexity, the size of the largest naïve RBF interpolation problem that can be solved on a given machine stays the same and does not increase with dimension, while SKI has nearly one-dimensional complexity. Hence the capability of SKI of addressing the complexity may be more attractive in higher dimensions. We shall look at the high dimensional implementation of SKI in Chapter 6.

The convergence of SKI has been verified by numerical examples with positive definite RBFs such as Gaussian, GIMQ, IMQ and IQ as basis functions, and among them the Gaussian shows the best convergence. Among the conditionally positive definite RBFs, SKI with MQ has a convergence nearly same as that with the above mentioned inverse multiquadrics. While the other CPD RBFs such as TPS2, TPS3 have either very slow convergence or no convergence at all.

Also, SKI converges at irregular sparse grid but at a somewhat slower rate than that for the regular sparse grids.

The numerical implementation has been done in *Matlab* serially. The Sparse Kernel-based Interpolation has advantages as compared to the direct full grid standard RBF interpolation. Firstly, obtaining the partial solution $u_{\mathbf{l}}$ on the full grids $\mathfrak{X}_{\mathbf{l}}$ of small resolution allows the possibility of using the existing codes which makes the sparse kernel-based interpolation as a straight forward technique for complicated problems and secondly, the discretisation on the small (full) grids can be done in parallel and this makes the combination technique perfectly suited for implementation on the modern high performance computing systems. In SKI, we make use of the combination technique accessing some nodes several times and is therefore expected to require more storage than the direct sparse grid methods. However, it is still making use of the same less input information required by the direct sparse grid algorithms. In any case, compared to the full grids

SKI requires very small amount of memory storage and is still good for large data and high dimensions. In addition, due to its parallel nature of solving independent but small problems, the SKI method can be implemented on modern HPC clusters in parallel environment. Moreover, even when implemented in series on a ordinary computer, SKI is faster and can solve larger interpolation problem than the direct interpolation on a sparse grid. SKI gives better efficiency in terms of stability, complexity and the run time and even in convergence over the full grid standard RBF interpolation approaches in most cases. In some cases this efficiency might be at the cost of negligible loss of accuracy.

Chapter 5

Multilevel Sparse Kernel-Based Interpolation

Multilevel approximation methods are concerned with the construction of a hierarchical representation of a model data, that stems from an unknown function, at several different resolutions. In practice a data set could be very large and its density in the set of data sites is subject to strong variations. Due to their heterogenous nature, such data sets naturally incorporate multiple resolutions. Hence, multilevel scattered interpolation methods turns out to be the appropriate tools for an efficient an effective recovery of such data sets [35], [93] [76], [30], [63], [77]. In the mid 90's, Floater and Iske presented a multilevel interpolation algorithm [35], which is the first idea on combining a thinning algorithm and compactly supported RBF interpolation. The multilevel interpolation combines the advantages of stationary and non stationary classical RBF interpolation, thereby accelerating the convergence. The method is based on a decomposition of the centers into a nested sequence of subsets of centres. These subsets are designed so that they have a uniform density and the density increases smoothly from lower to higher level. Density of a set $X \subset \Omega$ of centres can me measured by its fill distance $h_{X,\Omega}$ and by its separation distance q_X. This scheme has been further improved by using scattered data filtering instead of non-adaptive decomposition [63] and, more recently, developed further by combining the adaptive domain decomposition with stable local polyharmonic spline interpolations [65]. The convergence analysis of the method can be found in [55], [33], while application of these ideas in interpolation and in the solution of PDEs can be found in [76], [77] and [30], [93], respectively.

In this chapter, we review briefly the multilevel RBF interpolation and, subsequently, we present a multilevel version of the SKI algorithm.

5.1 Multilevel classical RBF interpolation

Given data $(\mathbf{x}_i, y_i)$; where $y_i = f(\mathbf{x}_i)$ for $i = 1, \ldots, N$, are the values of an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ at pairwise distinct data sites $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$, $y_i \in \mathbb{R}$, we want to find a continuous unknown function $s : \mathbb{R}^d \to \mathbb{R}$ such that $s(\mathbf{x}_i) = y_i$, $i = 1, \ldots, N$, where N is large enough.

To recover the unknown function s, we decompose the point set X into the following nested sequence of subsets

$$X_1 \subset \ldots \subset X_K = X. \tag{5.1}$$

Let s_k denote the multilevel interpolant to the given data set at the k-th level, and let $P_k f_{k-1}$ denote the interpolant to the residual $f - s_{k-1}$ at X_k. Set $s_o \equiv 0$ and let $s_k = s_{k-1} + P_k f_{k-1}$. Then $s_k|_{X_k} = f|_{X_k}$ [35], [65].

In numerical linear algebra a similar process is known as *iterative refinement.* The basic idea of the algorithm is to scale the support of the basis proportional to the fill distance at each level, interpolate the data at the coarsest level, and after the first level interpolate to residuals on progressively finer sets. In other words, it starts with smoothly varying basis functions on a very thin data set to recover the global behaviour of the unknown data function, followed by the use of increasingly oscillating basis functions on finer levels to recover the local behaviour of the data.

The coarsest level interpolant of the scheme is of crucial importance, as this where the global behaviour of the data is recovered. In other words, the convergence of the scheme is closely dependent on the approximation strength of the initial interpolant [64]. In order to achieve better approximation, the use of polyharmonic splines have been recommended [65], [64] as the basis for the coarsest level interpolant followed by the use of a suitable basis function of local support. In addition to better global recovery, the use of a CPD polyharmonic spline of order m guarantees the reproduction of any d-variate polynomial from the space π^d_{m-1} [64].

At the levels after the first, it is the support radius (in case of CSRBFs) or the scaling (in the case of GSRBFs) of the basis function used in the multilevel scheme that can be used to improve the approximation of the residual interpolants used for the updates. If the support radius/scaling of the basis function at the i-th step is kept proportional to the density of X_i, then the number of the data centers lying in the the region of influence of the basis function is bounded by a constant being independent of the level i. This results in the condition numbers of the interpolation matrix to be uniformly bounded and independent of the level i. This means a constant number of iterations will be required if an iterative linear solver is used. On the other hand in the context of compactly supported RBF, the bandwidth of the corresponding sparse system remains constant, and hence matrix-vector multiplication in the scheme will need linear time. In

[35], the support radius at each step is taken to be nearly equal to $5 \times h_{X_i,\Omega}$ in a model problem. In [63], the author remarks that the performance of the multilevel interpolation closely depend on the hierarchy (5.1) of the given data, and thereby suggests algorithms for construction of suitable data hierarchies. A multilevel approximation scheme based on adaptively constructed hierarchies is given in [65].

5.2 Multilevel sparse kernel-based interpolation

We apply the idea of the multilevel interpolation [35], [65] discussed above, to the sparse kernel-based interpolation presented in Chapter 4.

The main motivation for using multilevel decomposition in the sparse kernel-based interpolation comes from two facts:

- Each sparse grid interpolant S_n^c is a linear combination of partial interpolants $S_{A_\mathfrak{l}}$, where $S_{A_\mathfrak{l}}$ uses basis function $\varphi_{A_\mathfrak{l}}$, (where $\varphi_{A_\mathfrak{l}}(\cdot) = \varphi(A_\mathfrak{l}\cdot)$ and $\varphi(\cdot)$ is an RBF,) with their support kept proportional to density of the corresponding constituent grid $\mathfrak{X}_\mathfrak{l}$ by a suitably chosen scaling matrix $A_\mathfrak{l}$.

- The sparse grids from lower to higher level are nested, so that the sparse grid from level one to the current level provides us with a hierarchical decomposition of the sparse grid data. This nested-ness of the sparse grids can be seen in Figure 5.1, where $\mathfrak{Y}_{1,d}^s \subset \mathfrak{Y}_{2,d}^s \cdots \subset \mathfrak{Y}_{6,d}^s$.

Let Ω be an open set in $\mathbb{R}^d$, let X be the data sites in Ω, and let $f|_X$ be the given data. We extract the sparse grid data on $\mathfrak{Y}_{n,d}^s$ from the the given full grid data on X. Then decompose the sparse grid data into a hierarchy,

$$\mathfrak{Y}_{1,d}^s \subset \mathfrak{Y}_{2,d}^s \cdots \subset \mathfrak{Y}_{n,d}^s,$$

of nested sparse grids. This is simply the sequence of sparse grids from level 1 to n, whose density increases (the fill distance and separation distance decreases) from lower to higher levels.

The next step is to evaluate the sparse grid interpolant S_1^c to the data on the coarsest sparse grid Ω_1^s, and set $\triangle S_1^c = S_1^c$. In the followings steps, we continue by computing $\triangle S_k^c$ the interpolant to the residual $(f - S_{k-1}^c)|_{\mathfrak{Y}_{k,d}^s}$ from the previous interpolant on the current sparse grid, for $2 \leq k \leq n$. The current interpolant is then defined as $S_k^c = S_{k-1}^c + \triangle S_1^c$, where $S_0^c = 0$.

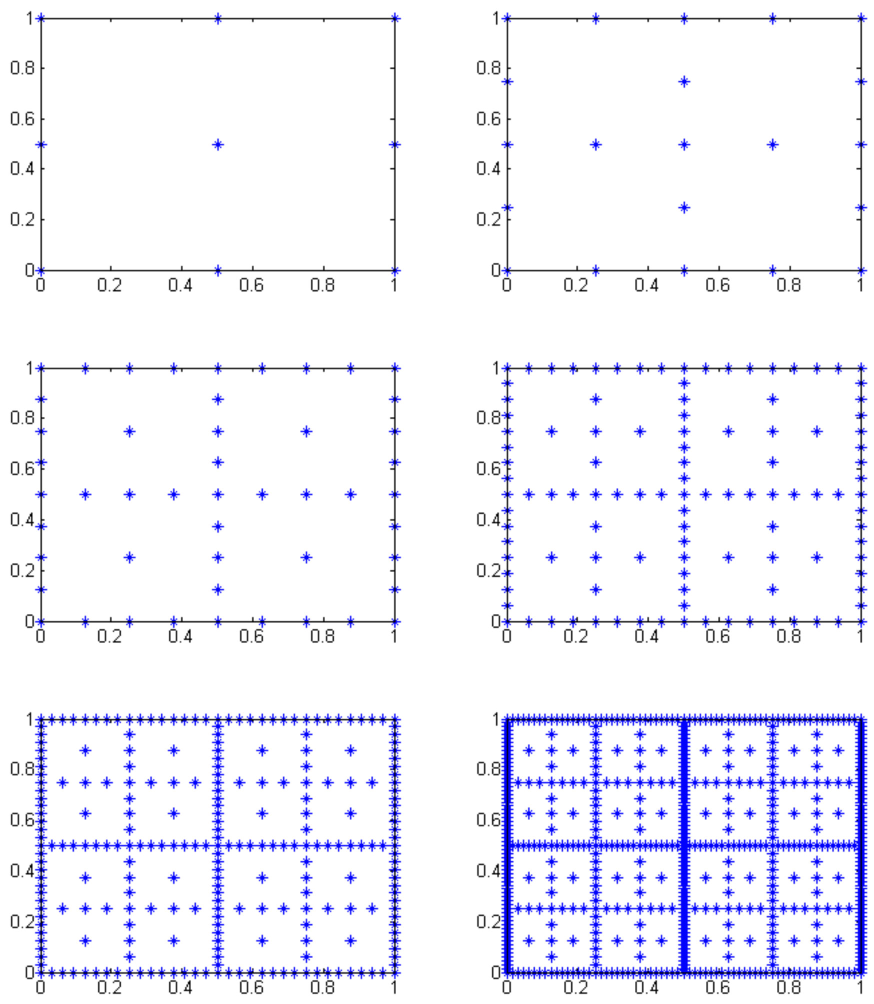


Figure 5.1: First six sparse grids in 2D

Algorithm 5.1 (Multilevel sparse kernel-based interpolation (MLSKI))

```
Input: Sparse grid data decomposition, $\mathfrak{Y}^s_{1,d} \subset \mathfrak{Y}^s_{2,d} \cdots \subset \mathfrak{Y}^s_{n,d}$, and data $f|_{\mathfrak{Y}^s_{n,d}}$
    1: Initialize the SKI interpolant, $S^c_0(\cdot) \equiv 0$;
    2: For, $k = 1, \ldots, n$, do
        2a: Evaluate the SKI residual approximant $\triangle S^c_k(\cdot) = f(\cdot) - S^c_{k-1}(\cdot)$ on $\mathfrak{Y}^s_{k,d}$
        2b: Update $S^c_k(\cdot) \leftarrow S^c_{k-1}(\cdot) + \triangle S^c_k(\cdot)$.
Output: Sequence of progressive approximations, $S^c_1$, $S^c_2$, $\cdots$, $S^c_n$, to $f$.
```

This algorithm is motivated by the the same idea of the classical multilevel interpolation. The method captures the global behaviour of the data by finding its sparse kernel-based interpolant to the data at the coarsest sparse grid. At the subsequent levels, more details are recovered by adding the SKI interpolant to the finer details left over from the previous interpolant. Finally, we get a sequence of n approximations to the given data at n sparse grids $\mathfrak{Y}^s_{1,d}, \dots, \mathfrak{Y}^s_{n,d}$. Due to the global recovery of the data at the initial level, the speed of convergence of MLSKI depends on the interpolation properties of the coarser level interpolant. Due to the application of SKI algorithm at each step, MLSKI scheme has the linear complexity and run time. In addition, MLSKI is capable of capitalising on the parallel nature of SKI and, therefore, could prove to be more effective in high dimensions specially if implemented in parallel on modern high computing systems.

5.3 Stability of MLSKI

The SKI algorithm naturally incorporates all the features required for the application of its multilevel version MLSKI. Moreover, the stability of MLSKI method directly depends on the stability of SKI. The scaling of the basis function and choosing its shape parameter has been discussed in Section 4.2. In applications, we follow the same procedure by adjusting the support radius of the basis functions in MLSKI, to keep the condition numbers uniformly bounded. To this end, we choose the scaling matrix $A_{\mathfrak{l}} = \text{diag}(2^{\mathfrak{l}})$ and the shape parameter as given in Section 4.2. The main scaling is done by $A_{\mathfrak{l}}$ and then using the strategy given in Section 4.2, we choose suitable shape parameter. In our experience the constant $K_d = 1$ gives nearly gives ideal conditioning of the small problem at the cost of very slow convergence. The choice $K_d = 3$, produces safe condition numbers together with faster convergence. Even if we choose $A_{\mathfrak{l}} = \text{diag}(2^{\mathfrak{l}})$, and, use suitable constant shape parameter for all the levels, the algorithm can be stable. For example, $c = 0.45$ for Gaussian, $c = 0.4$ for MQ, $c = 0.21$ for GIMQ, $c = 0.25$ for IMQ and $c = 0.25$ for IQ results in stable computations with safe condition numbers.

5.4 Numerical Experiments

We perform numerical experiments by interpolating data with a wide range of RBFs such as Gaussians, multiquadrics, inverse multiquadrics, polyharmoinc splines. We present numerical results for the interpolation of the data generated from test functions, Test1-2D, Test2-2D, Test3-2D, Test4-2D, Franke2D and Sinc2D listed in Chapter 4. In our numerical experiments from now on , "RBF" denotes classical radial basis interpolation, "MLRBF" denotes the classical multilevel interpolation of Floater and Iske [35], "SKI" denotes our sparse kernel-based interpolation and “MLSKI” the multilevel sparse kernel-based interpolation described in Section 5.2. Also, “N” denotes the number of full grid centres for the the classical and multilevel RBF interpolations, “SGnodes” denotes the are sparse grid nodes of the current level, while “DOFs(SG)” denotes the total number of times SGnodes are visited during the combination technique. We evaluate the errors at equally spaced 160×160 points and also at 25,600 Halton points in $[0, 1]^2$. In all the experiments, on ALICE as well as on an ordinary computer, the methods are implemented in *Matlab* serially.

5.4.1 Examples with Gaussian RBF

We implement MLSKI with the anisotropic Gaussian, $\varphi_{A_l}(\cdot) = \exp(-c^2\|A_l \cdot\|^2)$ as basis function. As expected, the multilevel sparse kernel-based interpolation algorithm is taking longer time than the direct sparse kernel-based interpolation but being linear is still affordable. The root mean squared error of the interpolants obtained with SKI, MLSKI, RBF and MLRBF are given in Figures 5.2, 5.3.

The multilevel sparse kernel-based algorithm is prominently cheaper than the multilevel RBF interpolation. In particular, these savings become more visible when size of the problem is large. When the time taken by MLSKI is measured in terms of problem size, MLSKI is superior to MLRBF.

The method is not only faster when considering the time as a function of the problem size, but it also wins by a margin when the convergence is studied as a function of the machine time. This can be seen in Figure 5.2, where the condition number stays safe and a fast convergence is observed. Also, for the non smooth function Test2-2D the performance of MLSKI is good.

In Figure 5.3, we give the RMS-error versus N (RBF and MLRBF) and SGnodes (for SKI and MLSKI). The stable computations given in Figure 5.3, show that multilevel sparse kernel-based interpolation is superior when compared with the direct SKI results. Apart from this, superior performance of MLSKI over the multilevel standard RBF interpolation has also been observed.

Moreover, experiments where large condition numbers are observed, the results suffer from instabilities. The corresponding numerical results are omitted. MLSKI gives better performance even in some of these unstable computations. We recall that this loss of accuracy in the unstable computations, has also been observed for direct SKI in Chapter 4. This can be seen in terms of the L_∞-error at the nodes in Table 5.1, where the L_∞-error at the nodes is nearly the machine precession. On the other hand, we observe that this error at nodes is evaluated to be nearly equal to the general error when the condition is large.

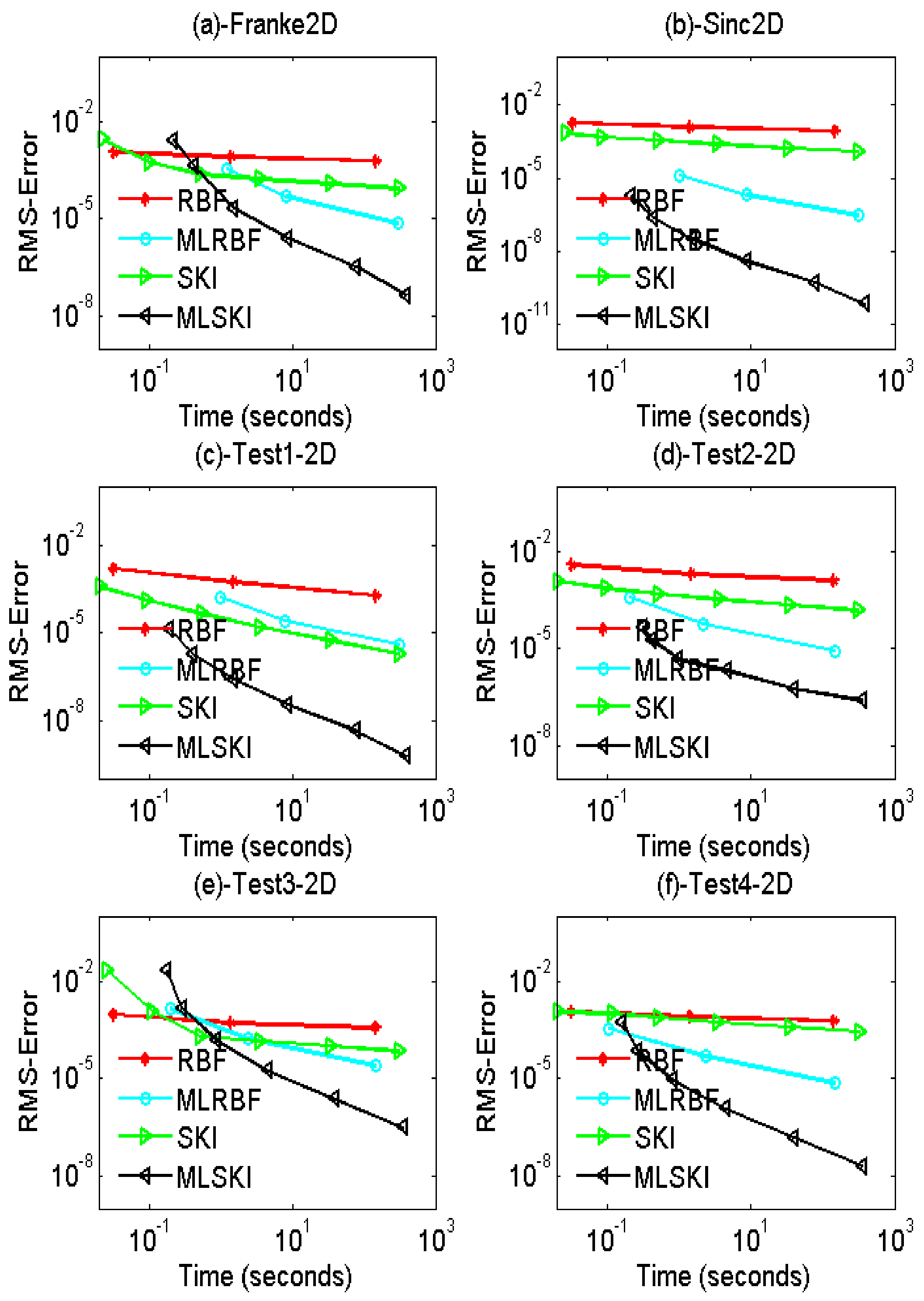


Figure 5.2: RMS error versus elapsed time using Gaussian RBF: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 25, 600 Halton points.

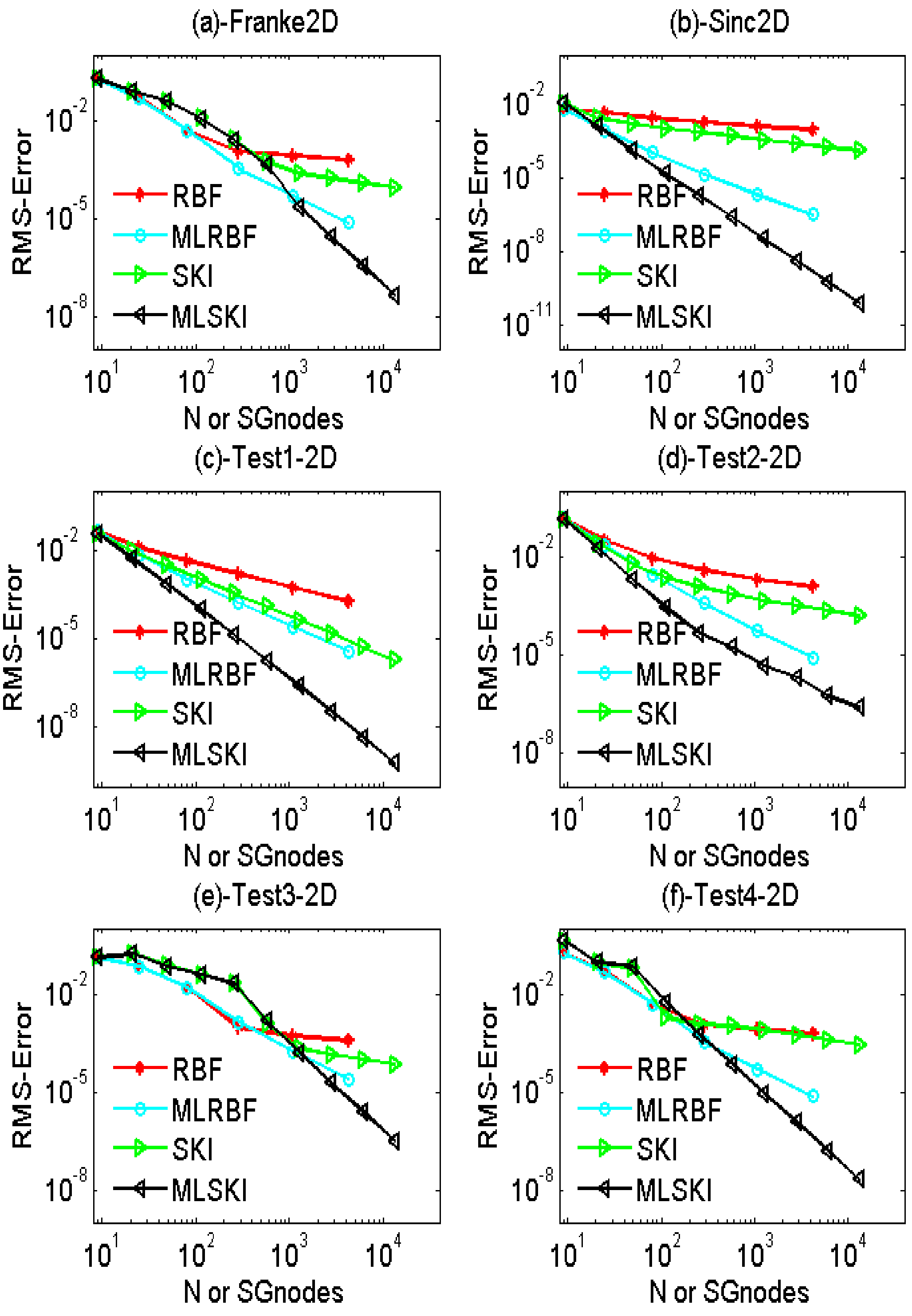


Figure 5.3: RMS-error versus N (RBF) and SGnodes (SKI) using Gaussian RBF: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers , evaluated at 25, 600 Halton points.

5.4.1(a) MLSKI simulations on ALICE

The experiments presented above have been performed on an ordinary computer. We extend our numerical simulations of MLSKI to larger sparse grids in $\mathbb{R}^2$, on ALICE. We can implement the scheme on sparse grid of level 12, where the number of nodes is $61,441$ while this number is more than 16 million for the corresponding full grid. The numerical results for multilevel sparse interpolation obtained on ALICE are given in Table 5.1, where the superior fast convergence of MLSKI is given in terms of its errors can be seen. Hence the suggested multilevel sparse kernel-based interpolation accelerates the convergence of SKI and has the linear complexity of SKI. In addition, MLSKI produces a superior performance in terms of complexity and convergence over the classical multilevel RBF schemes. Hence SKI has itself superior complexity and MLSKI has the potential to accelerate its convergence.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
9	9	6.2215e-1	1.8363e-1	9.7838e-16	2.6912e+3	2.9129e-1
21	39	3.3237e-1	7.6547e-2	3.1142e-14	2.5325e+4	3.3962e-1
49	109	1.1130e-1	3.8660e-2	4.9336e-13	2.8184e+5	3.9263e-1
113	271	4.0379e-2	1.0835e-2	7.8865e-13	2.6522e+6	4.5205e-1
257	641	1.2649e-2	2.5117e-3	7.6195e-13	2.9516e+7	5.2501e-1
577	1475	2.4678e-3	4.0273e-4	2.3428e-13	1.7591e+8	6.6061e-1
1281	3333	2.2043e-4	2.1030e-5	1.2689e-14	1.0484e+9	1.2329e+0
2817	7431	3.5287e-5	2.5391e-6	2.4980e-15	2.3229e+9	4.7431e+0
6145	16393	6.2139e-6	3.2696e-7	3.9552e-16	5.1468e+9	3.3353e+1
13313	35851	1.1784e-6	4.2920e-8	8.3267e-17	6.5016e+9	2.8112e+2
28673	77837	2.1204e-7	5.6557e-9	1.3878e-17	8.2129e+9	2.3971e+3
61441	167951	4.1321e-8	7.6854e-10	3.4694e-18	8.7056e+9	2.1586e+4

Table 5.1: Multilevel SKI results from ALICE, using Gaussian RBF with $c = 0.45$ for each level, test function Franke2D, evaluated at $25,600$ Halton points.

5.4.2 Examples with non Gaussian positive definite RBFs

After the encouraging performance of SKI and the high order convergence of MLSKI with Gaussian in the experiments of this chapter, we turn to the non Gaussian basis. Among the non Gaussian RBF, the inverse multiquadrics are positive definite. The most common in practice among this family of RBFs, are inverse quadric IQ $=\frac{1}{1+c^2r^2}$, inverse multiquadric IMQ $=\frac{1}{\sqrt{1+c^2r^2}}$, and generalized inverse multiquadrics GIMQ $=\frac{1}{(1+c^2r^2)^2}$.

We present our results with safe condition numbers, and those with large condition numbers are omitted. Results obtained by implementing SKI, MLSKI, RBF, MLRBF with GIMQ are given in Figure 5.4. For IMQ these numerics are given in Figure 5.5. While the corresponding results for IQ are given in Figure 5.6. We have also used WE32, $\varphi_{3,2}(r) = (1 - cr)_+^6(35c^2r^2 + 18cr + 3)$ as basis function and the results obtained are presented in Figures 5.7, where the parameter c used is small thus making WE32 nearly globally supported. By using IQ, IMQ and GIMQ as basis functions, we observe the same good performance of MLSKI over MLRBF as has been observed with Gaussian. The convergence is slower than that obtained with Gaussian and agrees with a similar performance of these basis in the context of classical RBF interpolation. This superior behaviour of MLSKI is lost if the condition number of the resulting systems are large. These results are similar to those previously observed in the examples of this kind in Section 5.4.1, therefore, we omit these results for brevity here.

5.4.3 Examples with conditionally positive definite RBFs

The conditionally positive definite RBFs such multiquadric MQ $=\sqrt{1 + c^2r^2}$, polyharmonic splines such as TPS2=$r^2log(r)$, TPS3=$r^4log(r)$, TPS4=$r^6log(r)$, TPS5=$r^8log(r)$, and radial powers, for example, r^3 are used as basis function in SKI and MLSKI experiments in this section. We present the results obtained with MQ in Figure 5.8. Error comparison for r^3 is given in Figures 5.9, while the corresponding results for TPS2 are given in Figures 5.10 and those for TPS3 can be seen in Figures 5.11.

In the stable regime of condition number, MLSKI with MQ, produces results (see Figure 5.8) which are more or less similar to those obtained for IQ, IMQ and GIMQ. On the other hand, MLSKI with splines such as TPS2, TPS3 and r^3 become nearly unstable and very slow convergence rates are observed. It is not clear why is this the case and further research is needed in this direction.

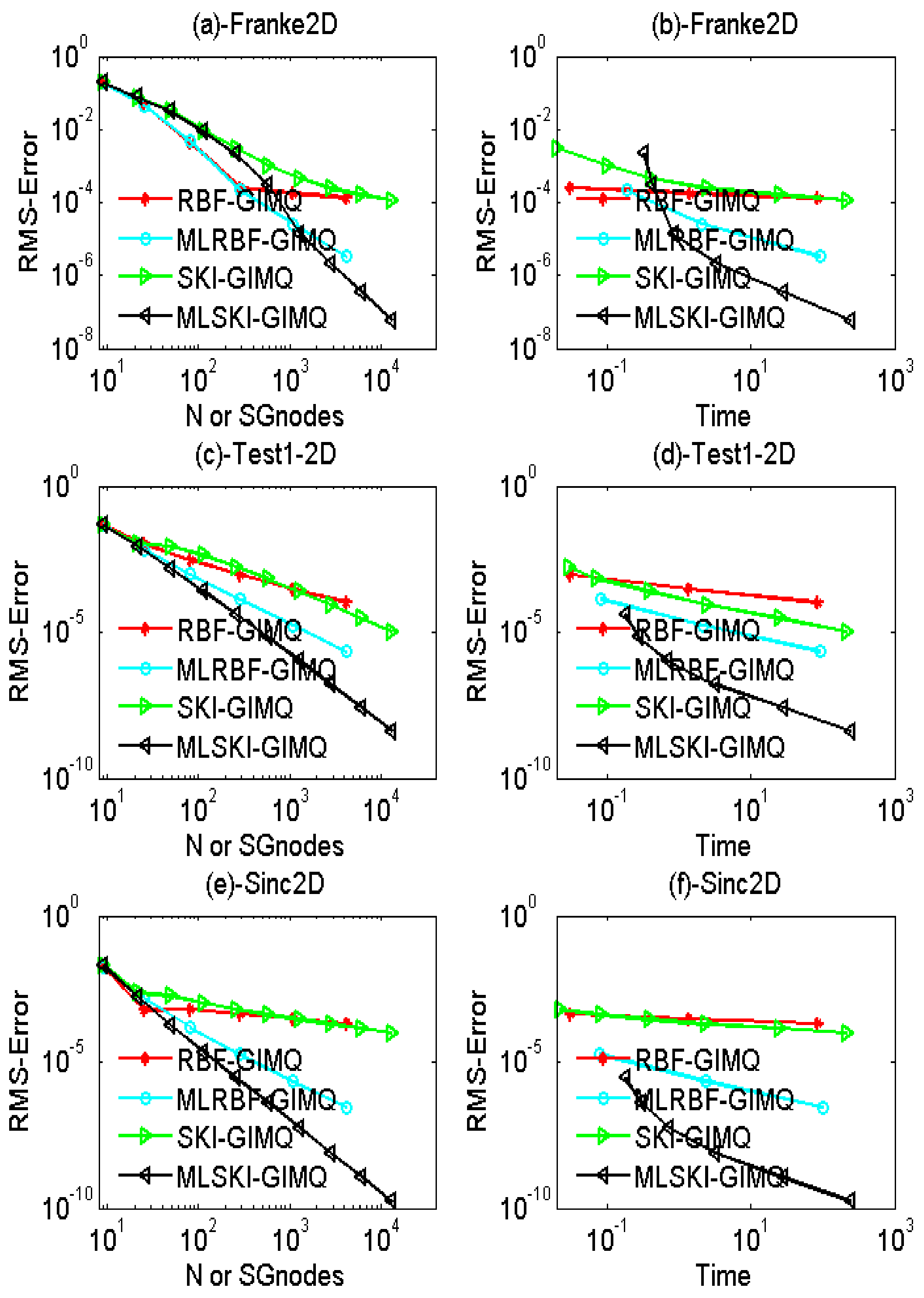


Figure 5.4: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using GIMQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 25,600 Halton points.

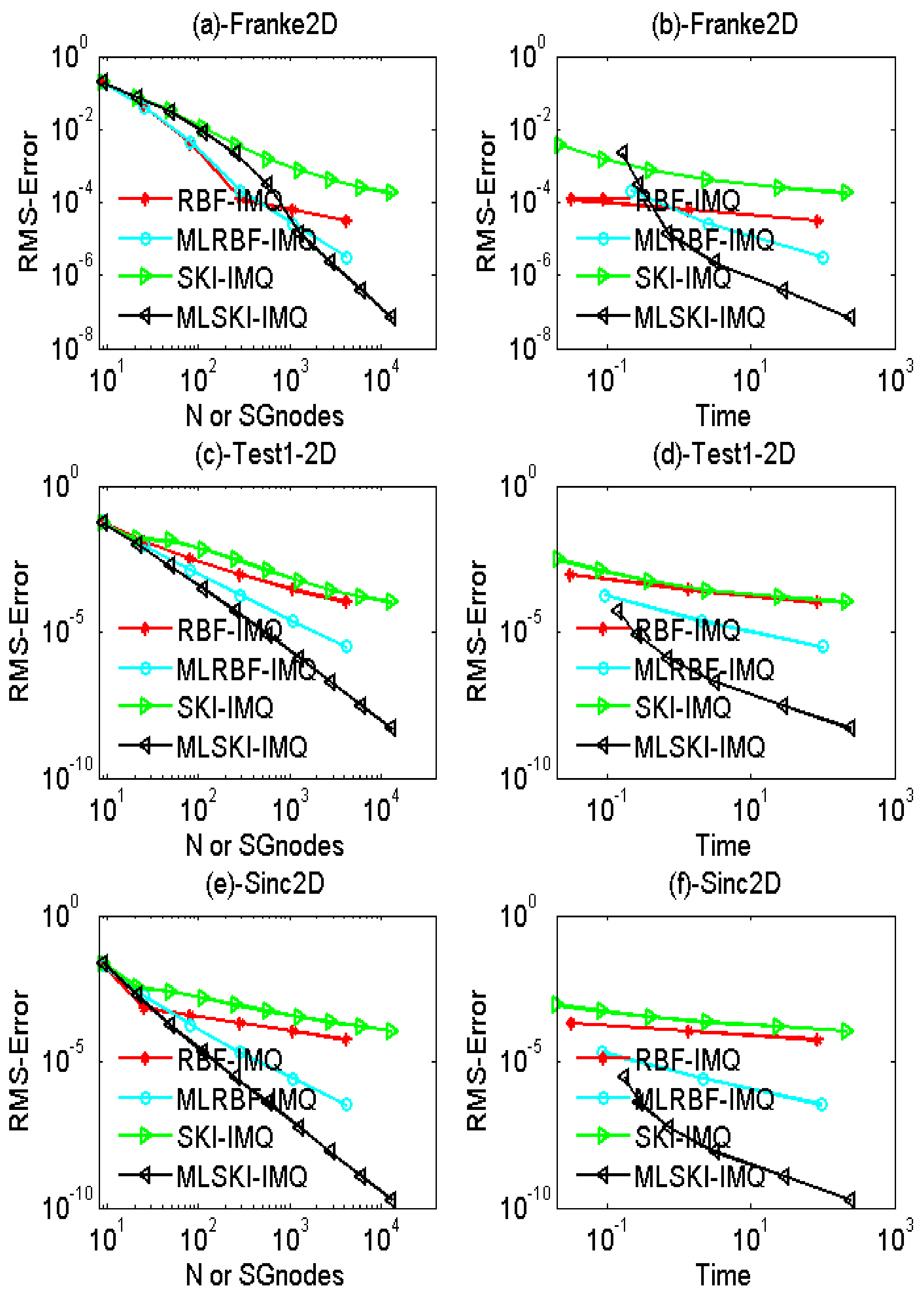


Figure 5.5: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using IMQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 25, 600 Halton points.

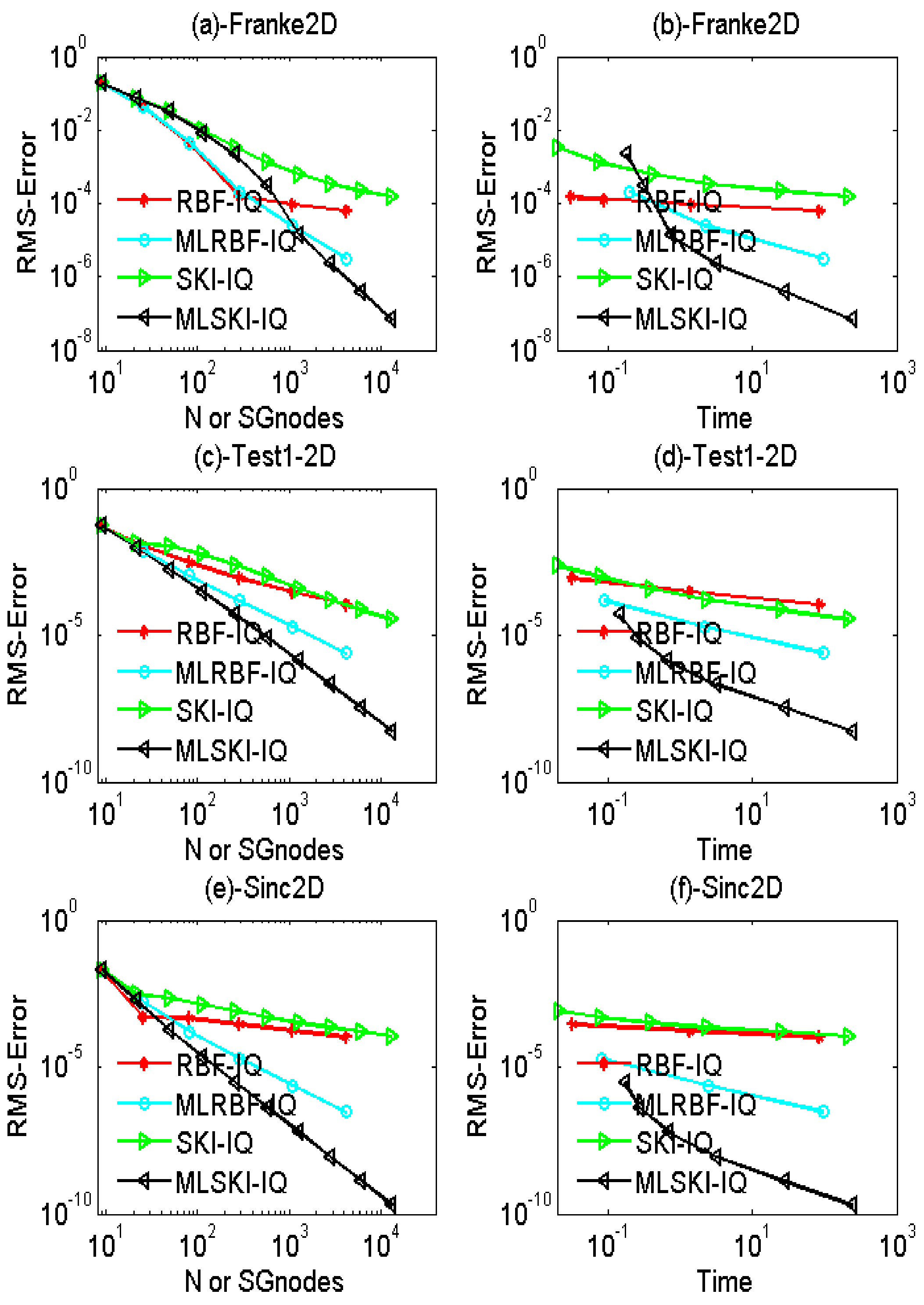


Figure 5.6: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using IQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 25, 600 Halton points.

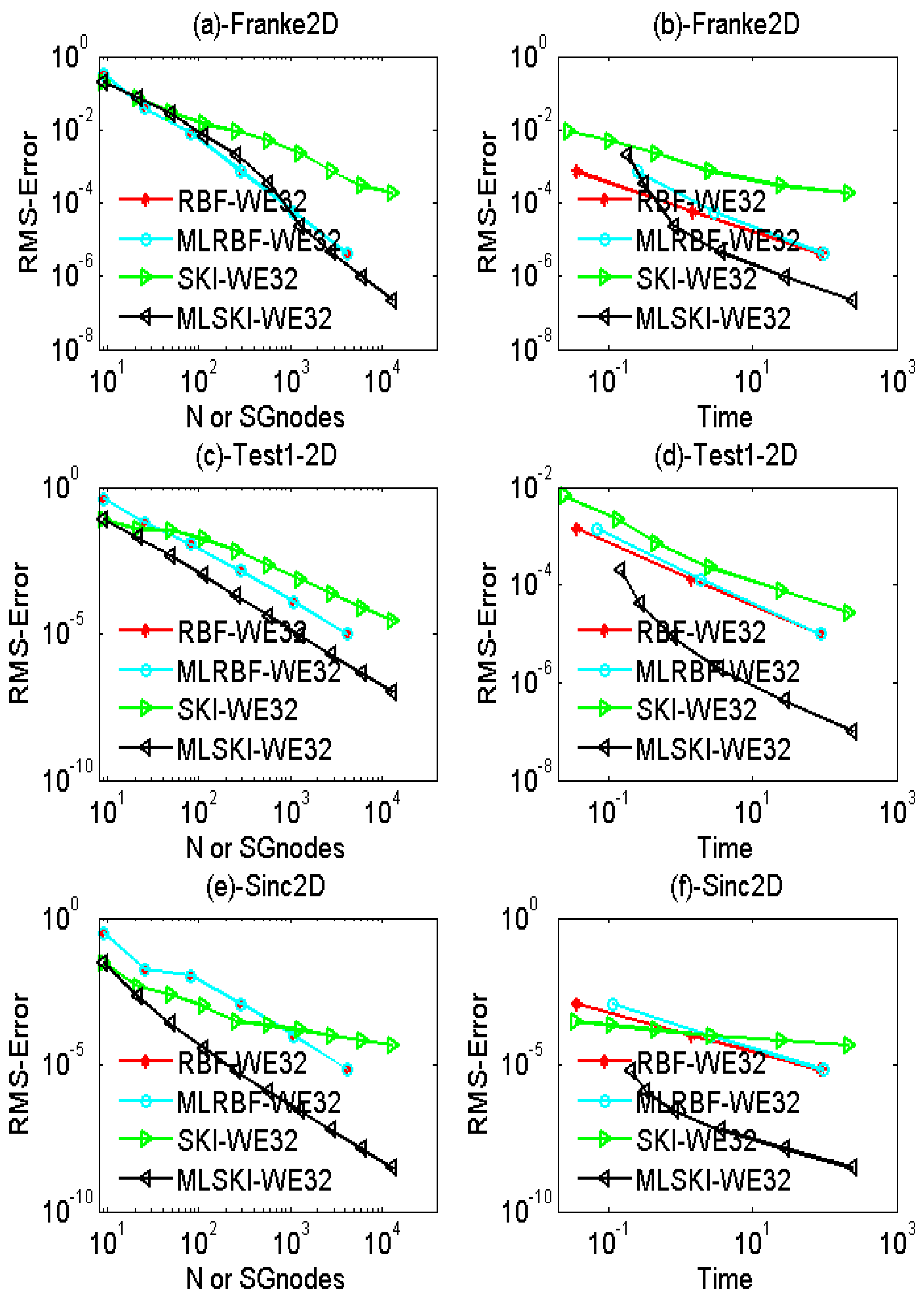


Figure 5.7: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using WE32 as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 25, 600 Halton points.

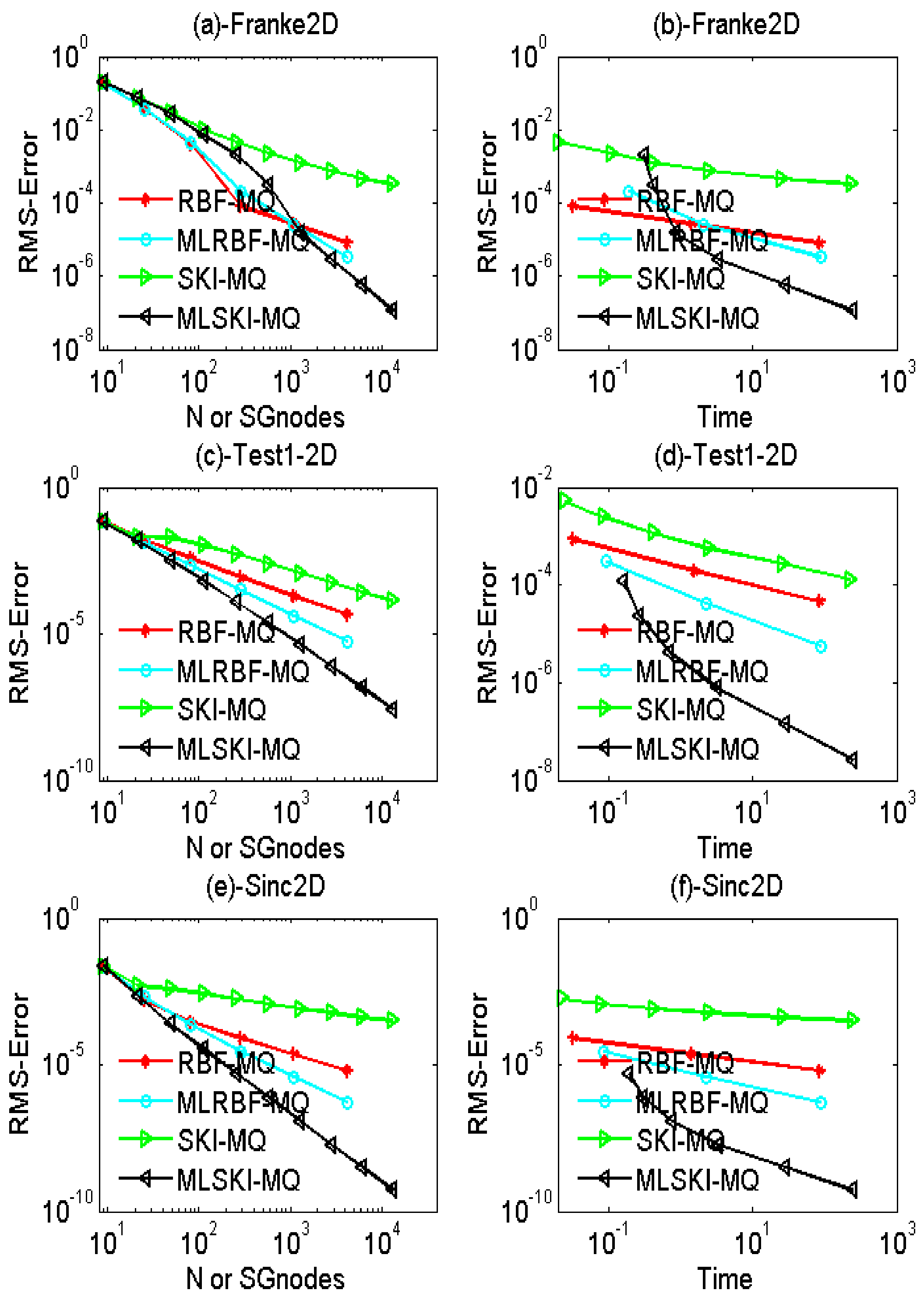


Figure 5.8: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using MQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 25, 600 Halton points.

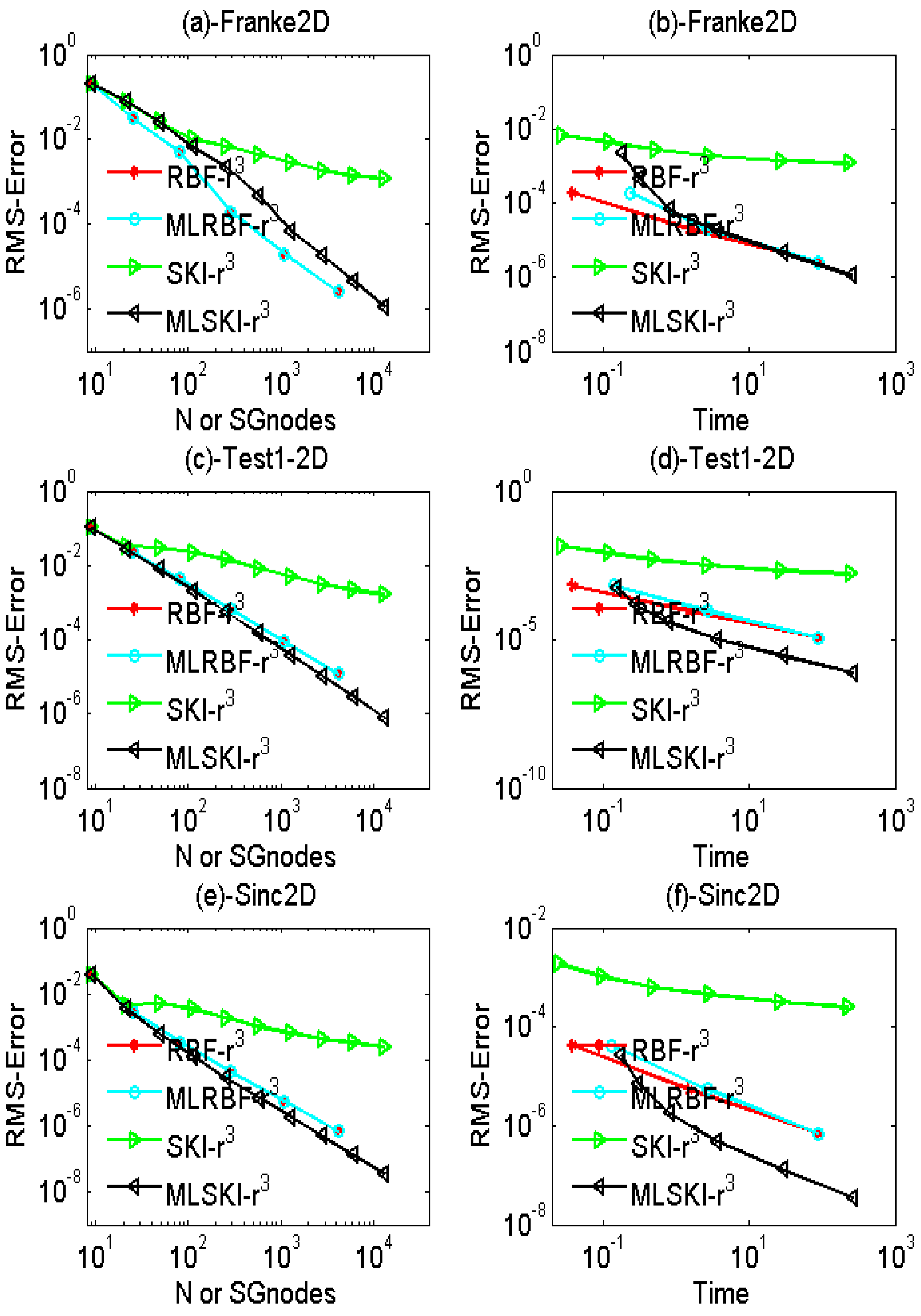


Figure 5.9: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using "r^3" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at $25,600$ Halton points.

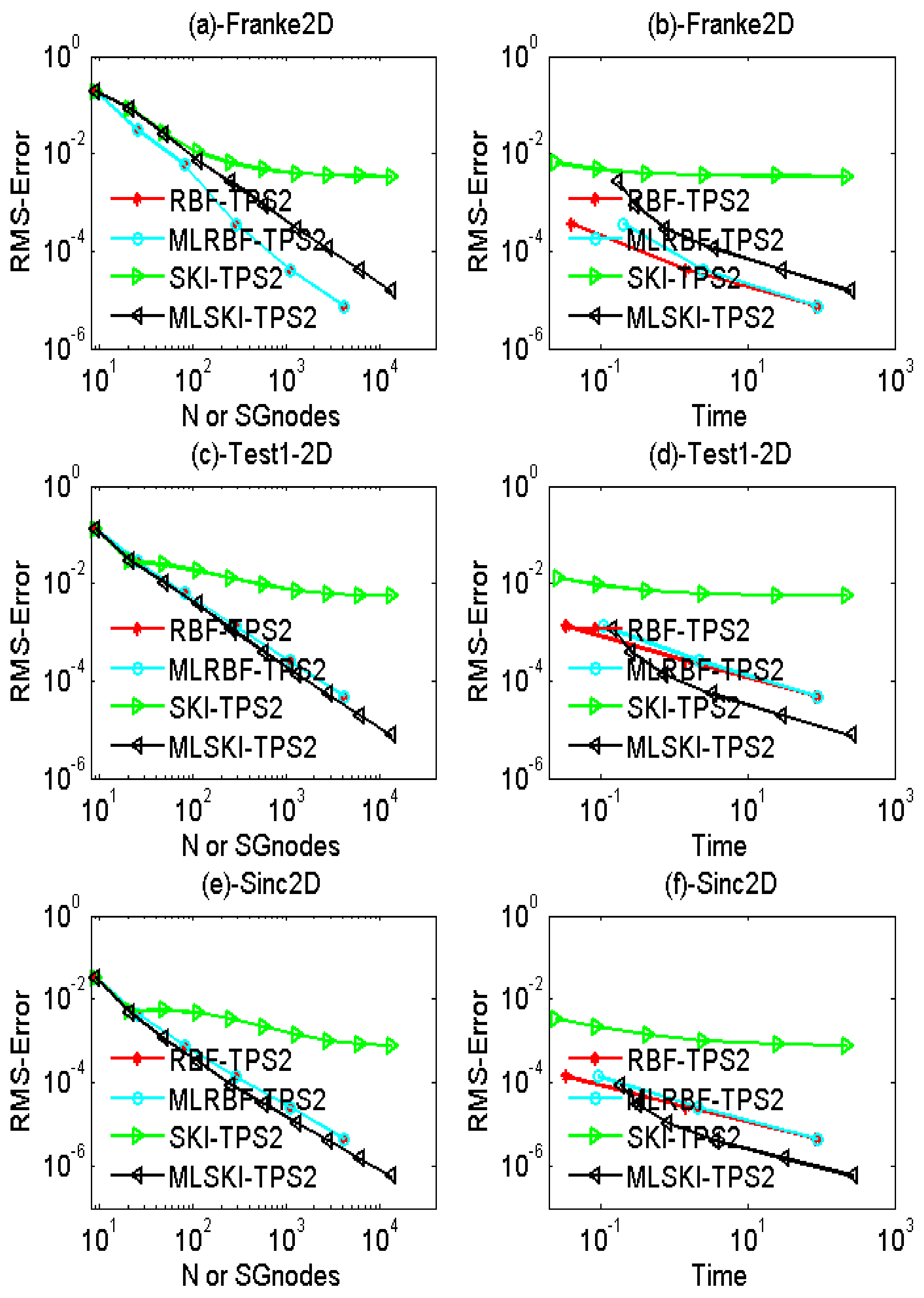


Figure 5.10: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using "$TPS2 = r^2 \log r$" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 25,600 Halton points.

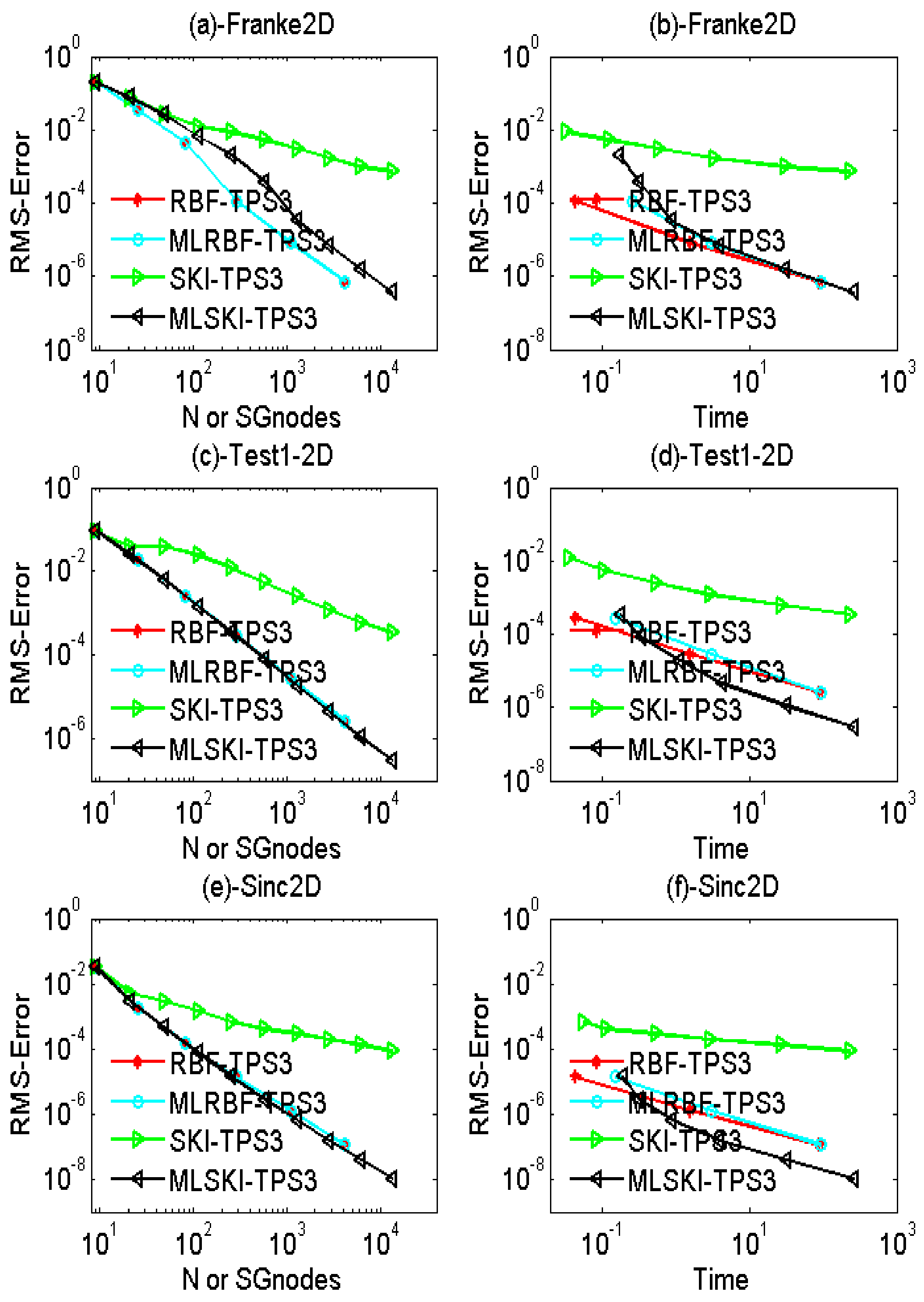


Figure 5.11: Left: RMS-error versus N (RBF) and SGnodes (SKI), Right: RMS-error versus CPU time. Using "$TPS3 = r^4 \log r$" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 25,600 Halton points.

5.5 Discussion

In this chapeter, the multilevel sparse kernel-based interpolation method has been presented. Numerical results of the MLSKI implementation, with wide a range of RBFs as basis, have been presented. As far as the condition numbers remain safe, MLSKI is capable of accelerating convergence of SKI by several orders with the same complexity and linear time of SKI. Moreover in experiments with nearly same condition numbers, MLSKI outperforms the standard multilevel RBF interpolation in terms of convergence and run time by a notable margin. We observe good convergence performance with positive definite RBFs such as Gaussian, IMQ, GIMQ and IQ. In the examples with conditionally positive definite, MQ has a convergence nearly equal to that of the inverse multiquadrics. In experiments with TPS2, TPS3, r^3, MLSKI becomes nearly unstable and the convergence is either very slow or even no convergence at all. We are not sure about the possible explanation of this behaviour and further investigations are required in this direction.

Due to the application of SKI algorithm at each step, MLSKI scheme has the linear complexity and run time. In addition, MLSKI is capable of capitalizing on the parallel nature of SKI and, therefore, could prove to be more effective in high dimensions specially if implemented in parallel on modern high computing systems. All the results presented have been obtained through a serial implementation of the algorithm. Due to its parallel nature, the generally superior performance of MLSKI in terms of run time and complexity could be even more prominent if implemented in a parallel environment.

The fast convergence obtained with basis functions such as Gaussian, MQ, GIMQ, IMQ, IQ, the capability to solve large 2-dimensional interpolations problems, the potential effectiveness and simple implementation in high dimensions are good features of MLSKI algorithm. MLSKI could be an alternative method for large interpolation problems, in particular when high order convergence is a requirement. These observations are good enough to motivate the application of MLSKI to d-variate interpolation, for $d = 2$. Higher dimensional experiments are presented in Chapter 6.

Chapter 6

High dimensional sparse kernel-based interpolation

The easy implementation and insensitivity of the RBFs to the dimension parameter d makes them potentially attractive for high dimensional problems. The only difference in different dimensions is calculating the radial distance r. In practice, the classical RBF interpolation is very expensive for large problems and is limited to only a modest size and low dimension of the interpolation problem. For example, an equally spaced full grid of level 6 in $[a, b]^3$ has size $(2^6 + 1)^3 = 2{,}74{,}625$ and in $[a, b]^4$ it becomes $(2^6 + 1)^4 = 17{,}850{,}625$.

In all our experiments so far, we have discussed the sparse kernel based interpolation of a bivariate function. That is, given the data $\{(x_1^i, x_2^i, y^i);$ for $i = 1, \cdots, N\} \subset \mathbb{R}^3$, we had to find a function $\mathcal{F}(x_1, x_2)$, such that $\{y^i = \mathcal{F}(x_1^i, x_2^i);\ for\ i = 1, \cdots, N\}$. This interpolation is usually referred to as $3D$ interpolation [37]. After achieving the encouraging results for $d = 2$, we perform sparse kernel based interpolation and its multilevel approach in higher dimensions, i.e., given the data $\{(x_1^i, \cdots, x_d^i, y^i);$ for $i = 1, \cdots, N\} \subset \mathbb{R}^{d+1}$, we are going to find a d-variate function $\mathcal{G}(x_1, \cdots, x_d)$ such that $\{y^i = \mathcal{G}(x_1^i, \cdots, x_d^i);\ for\ i = 1, \cdots, N\}$ for $d = 3$, i.e., we shall be interpolating $4D$ and $5D$ data. We use a desktop computer "Core 2 Duo CPU @ 3.16GHz 3.17GHz and 3.24GB of RAM" for d= 3. For d=4, we use ALICE to be able to run many experiments at the same time by accessing as many nodes of ALICE. Each experiment was run in serial on a single node (having a pair of quad-core 2.67GHz Intel Xeon X5550 CPUs and 12GB of RAM) which is equivalent to a more powerful desktop computer. Thus all the CPU timing and problem size shown here relate to computations that can be run on the ordinary (less powerful) computer mentioned above. However, it is worth mentioning that the largest size of the problem in each of these computation has been kept the same for which all these schemes can be implemented on the smaller computer of 3GM RAM mentioned above. Moreover, we are going to report separately the largest size of the problem that SKI can solve on a single node of ALICE for d =3, 4 in Tables 6.1, 6.2, 6.3.

6.1 MLSKI with Gaussians

6.1.1 3-variate interpolation

We present some numerical experiments by implementing the SKI scheme (4.1) and its multilevel version described in Chapter 5 to approximate a 3-variate data.

We use the following 3-variate test functions to generate the data in our experiments.

Franke3D:

$$\begin{aligned} F_1(x_1,x_2,x_3) &= \frac{3}{4}e^{(-(9x_1-2)^2-(9x_2-2)^2-(9x_3-2)^2)/4} \\ &+\frac{3}{4}e^{-((9x_1+1)^2)/49-((9x_2+1)^2)/10-((9x_3+1)^2)/29} \\ &+\frac{1}{2}e^{-((9x_1-7)^2)/4-(9x_2-3)^2-((9x_3-5)^2)/2} \\ &-\frac{1}{5}e^{-((9x_1-4)^2)/4-(9x_2-7)^2-((9x_3-5)^2)} \end{aligned}$$

Test1-3D:

$$G(x_1,x_2,x_3) = 4^3x_1(1-x_1)x_2(1-x_2)x_3(1-x_3).$$

Test2-3D:

$$G_1(x_1,x_2,x_3) = (r^2+r^4)\log(r), \text{ where, } r=\sqrt{x_1^2+x_2^2+x_3^2}.$$

Test3-3D:

$$G_2(x_1,x_2,x_3) = \frac{1}{2}x_2\cos^4(x_1^2+x_2+x_3-1).$$

Test4-3D:

$$G_3(x_1,x_2,x_3) = \sqrt{\frac{18}{\pi}}e^{(-x_1^2-81x_2^2-x_3^2)}.$$

Sinc3D:

$$H(x_1,x_2,x_3) = \prod_{i=1}^{3}\frac{\sin(\pi x_i)}{\pi x_i}.$$

The combination formula of Algorithm (4.1) for d=3 is:

$$S_n^c = \sum_{|\mathbf{l}|_1=n+2} S_{A_\mathbf{l}} - 2\sum_{|\mathbf{l}|_1=n+1} S_{A_\mathbf{l}} + \sum_{|\mathbf{l}|_1=n} S_{A_\mathbf{l}},$$

and $\mathfrak{Y}^s_{5,3}$ is shown in Figure 6.1. We present our numerics from SKI, MLSKI, RBF and MLRBF algorithms used to interpolate the data generated from the above test functions. In the numerical experiments, we observe that the run time of SKI is less than the time spent by the direct RBF interpolation in particular when the problem is large. A similar relation between time and the problem size has been observed for MLSKI and MLRBF interpolation. Hence, SKI and MLSKI have superiority in terms of run time over RBF and MLRBF respectively. The comparison from the stable experiments are given in Figures 6.2, 6.3, while we omit the results with larger condition numbers. In Figures 6.2, 6.3, the superior performance of MLSKI over MLRBF is clearly observed. We obtain smaller error in smaller time for SKI and MLSKI as compared to RBF and MLRBF approaches, respectively. In fact MLSKI out performs MLRBF by big margin in both the cases irrespective of the fact that the error is considered as a function of the elapsed Figure 6.2, or the size of the input data Figure 6.3. The numerical results for large condition number show that convergence could be faster in some cases. But generally the superiority decreases and the error at the sparse grid nodes approaches the general error instead of zero when the condition is large, the results corresponding to larger condition numbers are omitted for brevity. Most of the results of this section reconfirms the general superiority of SKI and MLSKI already observed in $\mathbb{R}^2$ and extends this to its success for interpolation d-variate function for $d = 3$. We observe that this performance could be worse in some cases, for example, in the numerical examples where data comes from function Test3-3D.

We have performed the experiments presented in Figures 6.2, 6.3, on an ordinary computer of 3GB RAM. The SKI and MLSKI algorithms, can be implemented up to a sparse grid of level 8, having 21,249 nodes to approximate a 3-variate function. The corresponding full grid of this level would have 16,974,593 nodes, while the maximum number of nodes for the RBF and MLRBF interpolation on the same computer is nearly 5,000.

We continue with our 3D experiments on ALICE. SKI and MLSKI implementation on ALICE can stably compute on a sparse grid $\mathfrak{Y}^s_{10,3}$ containing 114,689 data sites. These results are given in Tables 6.1, 6.2. A full grid of level ten, in $[a, b]^3 \in \mathbb{R}^3$, would be as large as 10^9. Moreover, in our 3-variate experiments the maximum size of the RBF and MLRBF interpolation problem on ALICE is only 15,000. Hence the power of SKI and MLSKI of addressing the complexity is becoming clearer as compared to that previously observed in the lower dimension.

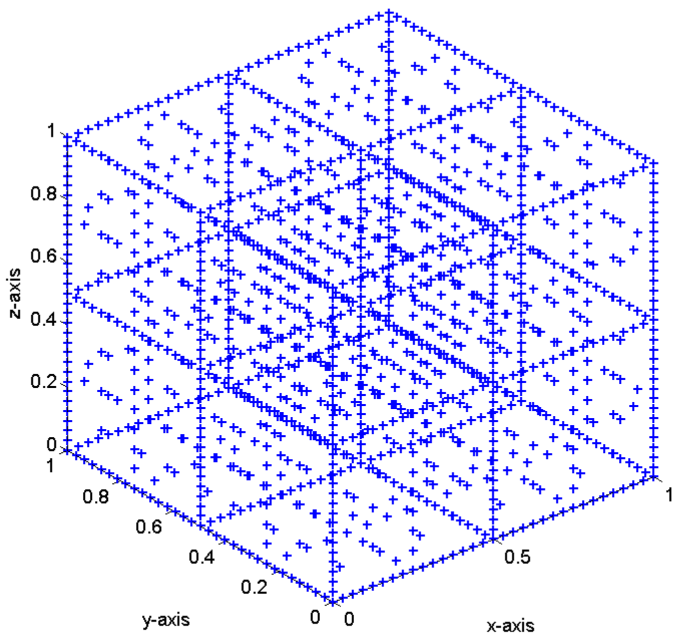


Figure 6.1: Sparse grid $\mathfrak{Y}^{s}_{5,3}$of level 5 in $[0\ 1]^3$.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
27	27	6.8808e-1	1.0179e-1	1.0546e-15	1.4863e+4	4.1636e-1
81	162	5.5583e-1	7.6769e-2	2.3190e-13	2.5376e+5	4.2074e-2
225	630	2.4538e-1	3.8721e-2	4.2744e-14	6.2956e+4	1.2179e-1
593	1997	1.5956e-1	2.1968e-2	2.2597e-12	1.0617e+7	1.2703e-1
1505	5687	6.4305e-2	7.1189e-3	1.3312e-13	6.9125e+5	4.3387e-1
3713	15188	1.4063e-2	1.8689e-3	9.8557e-12	4.2986e+8	2.7177e+0
8961	38868	2.4668e-2	5.8622e-4	8.5910e-14	6.2618e+6	2.4513e+1
21249	96471	6.6167e-3	1.7822e-4	1.8969e-12	9.2544e+9	2.3052e+2
49665	233949	1.4608e-2	2.9099e-4	5.0737e-14	2.1038e+7	2.0410e+3
114689	557030	6.0204e-3	8.9496e-5	9.4498e-13	1.0594e+11	1.6639e+4

Table 6.1: SKI results from ALICE $c = \frac{2^{h_r}}{3}$, using test function Franke3D, evaluated on 125,000 Halton points.

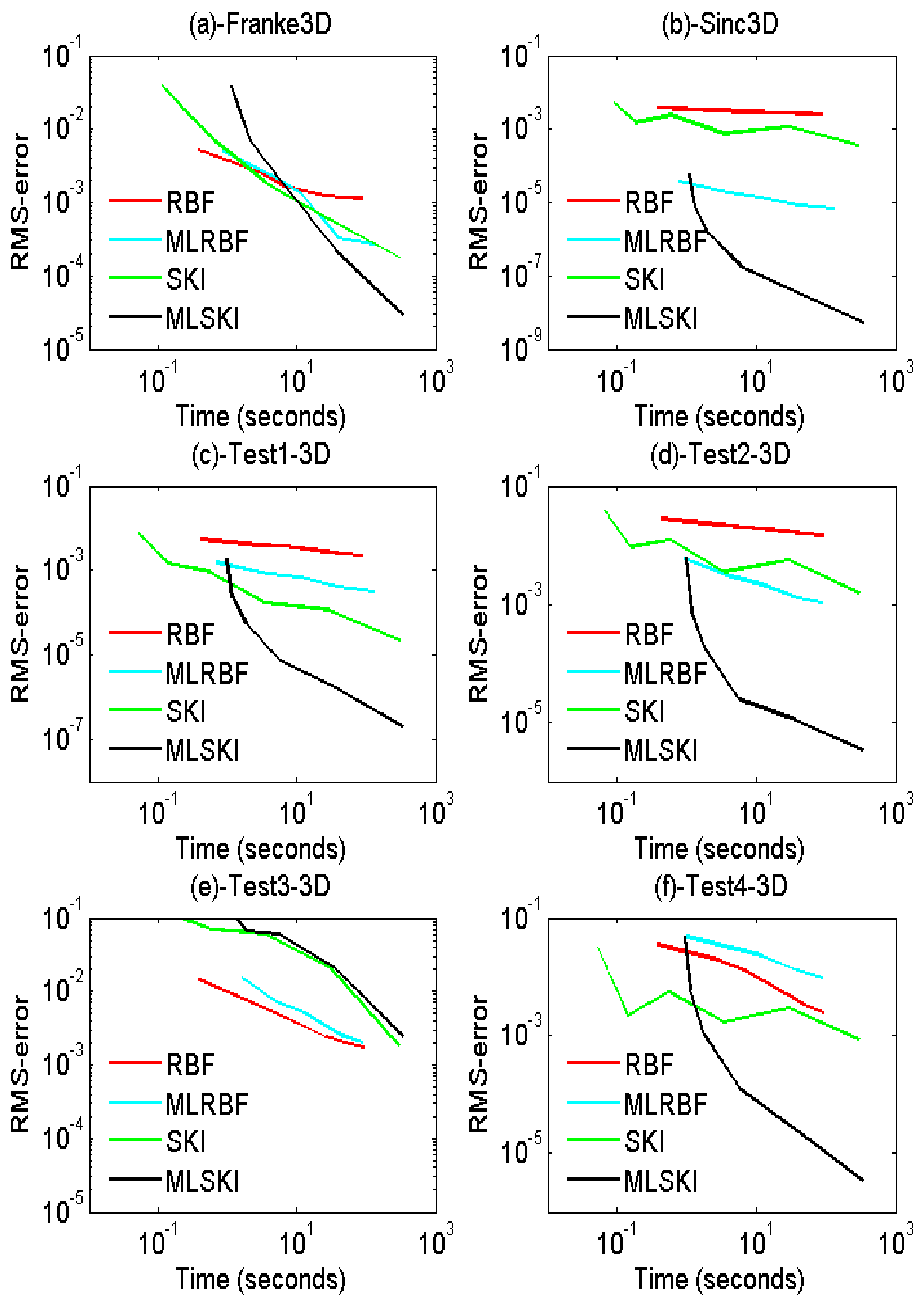


Figure 6.2: RMS-error versus CPU-time using Gaussian RBF: SKI (Green), RBF (Red), MLSKI (Black), MLRBF (Cyan), with safe condition numbers. Error evaluated at 125,000 Halton points.

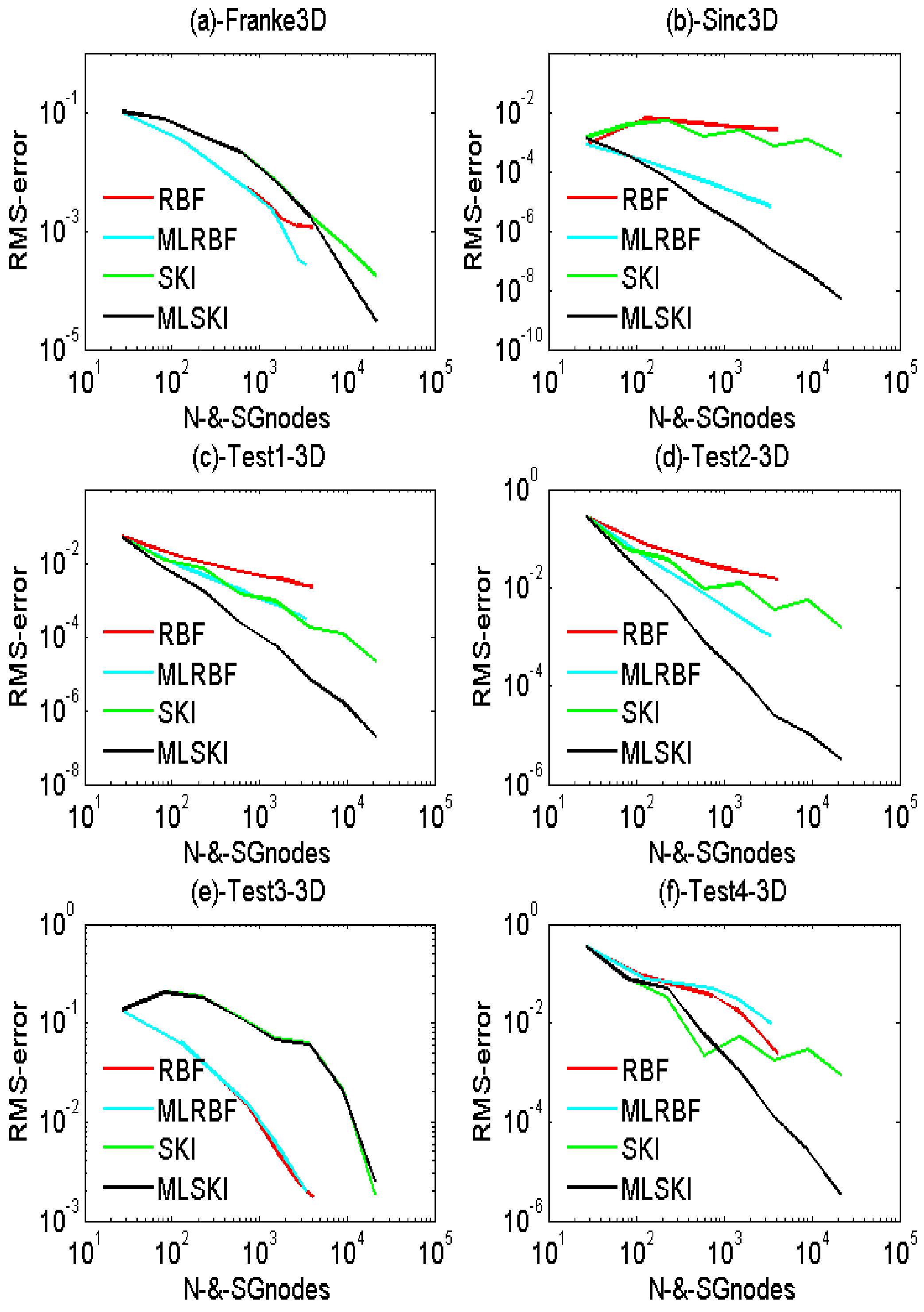


Figure 6.3: RMS-error versus N (RBF) and SGnodes (SKI) using Gaussian RBF: SKI (Green), RBF (Red), MLSKI (Black), MLRBF (Cyan), with safe condition numbers. Error evaluated at 125,000 Halton points.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
27	27	6.8808e-1	1.0179e-1	1.0546e-15	1.4863e+4	7.1541e+0
81	162	5.5853e-1	7.7339e-2	2.0734e-13	2.5376e+5	7.3306e+0
225	630	2.4324e-1	3.8389e-2	3.4241e-14	6.2956e+4	7.5388e+0
593	1997	1.5884e-1	2.1676e-2	2.2856e-12	1.0617e+7	7.8438e+0
1505	5687	6.2918e-2	6.7591e-3	4.2147e-14	6.9125e+5	8.5777e+0
3713	15188	1.3403e-2	1.7755e-3	2.4004e-12	4.2986e+8	1.2183e+1
8961	38868	2.2041e-3	2.2448e-4	3.2266e-15	6.2618e+6	4.2074e+1
21249	96471	3.3081e-4	2.9755e-5	9.8095e-14	9.2544e+9	3.2839e+2
49665	233949	8.9456e-5	4.5151e-6	5.9550e-17	2.1038e+7	2.9008e+3
114689	557030	1.5829e-5	5.7471e-7	3.6464e-15	1.0594e+11	2.5160e+4

Table 6.2: MLSKI results from ALICE , using Gaussian RBF with $c = \frac{2^{h_r}}{3}$, test function Franke3D, evaluated at 125,000 Halton points.

6.1.2 4-variate interpolation

Approximating an unknown data that stems from a function of 4 or more independent variables has always been a challenge in practice. For instance, RBF interpolation of a full-grided data is limited to nearly 10 points in each co-ordinate direction for d=4. This may not be enough to get a desired accuracy. We perform the following experiments by implementing SKI and its multilevel counterpart to approximate a 4-variate function. The combination formula of SKI algorithm for 4-variate interpolation becomes:

$$S_n^c = \sum_{|\mathbf{l}|_1=n+3} S_{A_\mathbf{l}} - 3\sum_{|\mathbf{l}|_1=n+2} S_{A_\mathbf{l}} + 3\sum_{|\mathbf{l}|_1=n+1} S_{A_\mathbf{l}} - \sum_{|\mathbf{l}|_1=n} S_{A_\mathbf{l}}$$

We extend the test functions from the lower dimensional examples to get the following 4-variate functions.

Franke4D:

$$\begin{aligned} F_1(x_1,x_2,x_3,x_4) &= \frac{3}{4}e^{(-(9x_1-2)^2-(9x_2-2)^2-(9x_3-2)^2)/4-(9x_4-2)^2)/8} \\ &+\frac{3}{4}e^{-((9x_1+1)^2)/49-((9x_2+1)^2)/10-((9x_3+1)^2)/29-((9x_4+1)^2)/39} \\ &+\frac{1}{2}e^{-((9x_1-7)^2)/4-(9x_2-3)^2-((9x_3-5)^2)/2-((9x_4-5)^2)/4} \\ &-\frac{1}{5}e^{-((9x_1-4)^2)/4-(9x_2-7)^2-((9x_3-5)^2)-((9x_4-5)^2)} \end{aligned}$$

Test1-4D:

$$G(x_1, x_2, x_3, x_4) = 4^4 x_1(1 - x_1)x_2(1 - x_2)x_3(1 - x_3)x_4(1 - x_4).$$

Test2-4D:

$$G_1(x_1, x_2, x_3, x_4) = (r^2 + r^4)\log(r), \text{ where, } r = \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}.$$

Test3-4D:

$$G_2(x_1, x_2, x_3, x_4) = \frac{1}{2} x_2 \cos^4(x_1^2 + x_2 + x_3 + x_4^2 - 1).$$

Test4-4D:

$$G_3(x_1, x_2, x_3, x_4) = \sqrt{\frac{18}{\pi}} e^{(-x_1^2 - 81(x_2^2 + x_4^2) - x_3^2)}.$$

Sinc4D:

$$H(x_1, x_2, x_3, x_4) = \prod_{i=1}^{4} \frac{\sin(\pi x_i)}{\pi x_i}.$$

The data to be interpolated in each example to follow has been generated from one of the above test functions. The numerical results obtained are plotted in Figures 6.4, 6.5. We observe the same superiority of SKI and MLSKI algorithms over the direct RBF and MLRBF interpolation in terms of convergence, complexity and run time as already been observed in the case of lower dimensions. MLSKI accelerates the convergence of SKI. We also perform experiments resulting in larger condition numbers, and a loss of accuracy similar to the corresponding lower dimensional examples is observed. For brevity, results corresponding to large condition numbers are omitted. Using an ordinary computer, SKI and MLSKI produces stable computations on a sparse grid of level seven, which has 52,993 nodes, while the corresponding problems size for RBF and MLRBF that can be solved on the same machine is nearly 5,000. Moreover, on the corresponding full grid of level 7, we would have to deal with 276,922,881 nodes. The implementation of SKI and MLSKI has been successful for data on sparse grid of level nine , which contains 331,780 nodes, when the bigger computer ALICE is used. Note that a full grid of this level is as large as 10^{11}. Moreover, maximum size of RBF and MLRBF interpolation problem in our experience on ALICE is limited to only 15,000 data sites. The numerical results up to level nine are presented in Table 6.3. Thus proposed SKI and MLSKI algorithms have been successfully applied to the interpolation problems in $\mathbb{R}^4$ with encouraging performance in particular in terms of complexity. The numerical investigations of this section suggest that SKI and MLSKI can be successfully implemented in $\mathbb{R}^d$ for $d \geq 2$ and confirms its superior performance over the direct and multilevel RBF approaches.

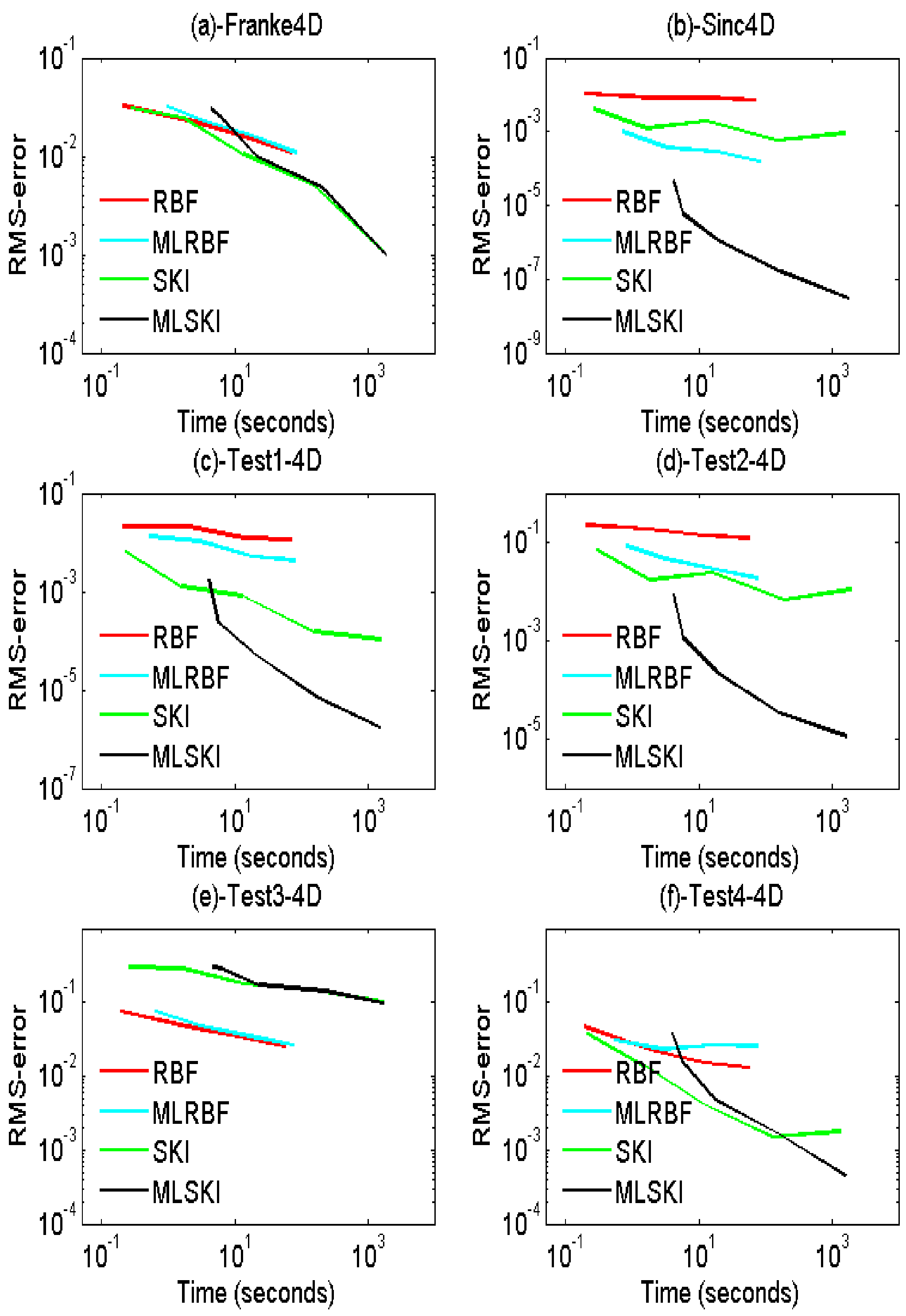


Figure 6.4: RMS-error versus CPU-time using Gaussian RBF: SKI (Green), RBF (Red), MLSKI (Black), MLRBF (Cyan), with safe condition numbers. Error evaluated at 194,481 uniformly distributed points.

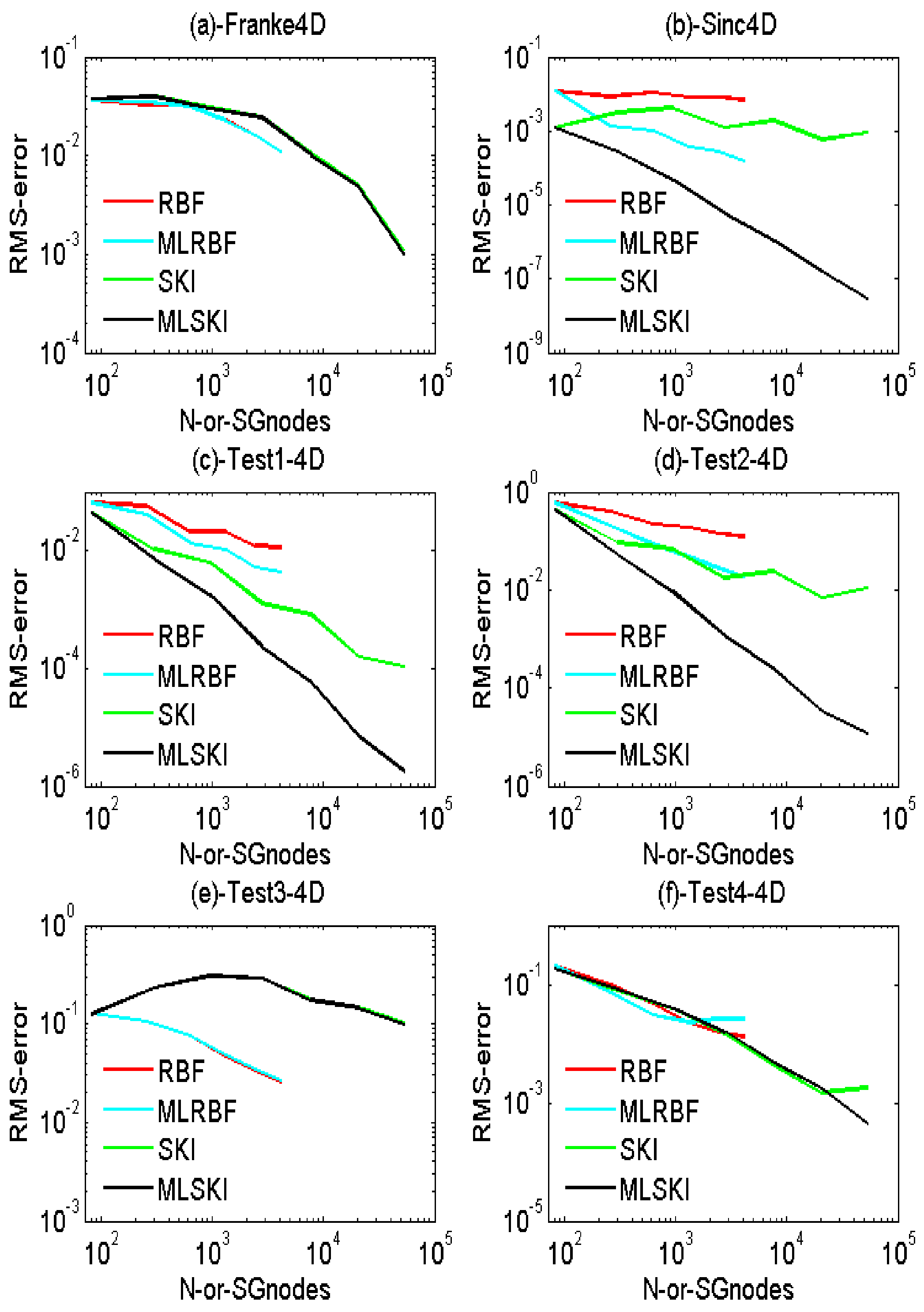


Figure 6.5: RMS-error versus N (RBF) and SGnodes (SKI) using Gaussian RBF: SKI (Green), RBF (Red), MLSKI (Black), MLRBF (Cyan), with safe condition numbers. Error evaluated at 194,481 Halton points..

6.2 Numerical examples with non Gaussian positive definite RBFs

All the numerics presented so far in this chapter are obtained with Gaussian basis function. Examples with inverse multiquadrics GIMQ, IMQ and IQ as basis are presented now.

6.2.1 3-variate interpolation

We are using test functions of Section 6.1.1 to generate data. Numerical results of RBF, MLRBF, SKI and MLSKI obtained with GIMQ as basis function are combined in Figure 6.6. For IMQ, these results are presented in Figure 6.7. While examples with IQ can be seen in Figure 6.8. These results reconfirms the nice features of SKI and MLSKI observed in the 2-dimensional experiments with PD non Gaussian RBFs discussed in Chapter 4, 5. We have only presented results from stable experiments and the results corresponding to larger condition numbers are omitted.

6.2.2 4-variate interpolation

We present our compare the interpolation errors of RBF, MLRBF, SKI and MLSKI implemented with GIMQ in Figure 6.10. Such results for IMQ in Figure 6.11, and those for IQ are given in Figure 6.12. The results presented correspond to the experiments with safe condition numbers. For brevity, we omit the results for larger condition number. The results show the encouraging performance of the proposed algorithms with these positive definite RBFs. Our observations made in this section are analogous to those observed for the algorithm with the same RBFs in the lower dimensional experiments presented in this chapter and in Chapters 4, 5.

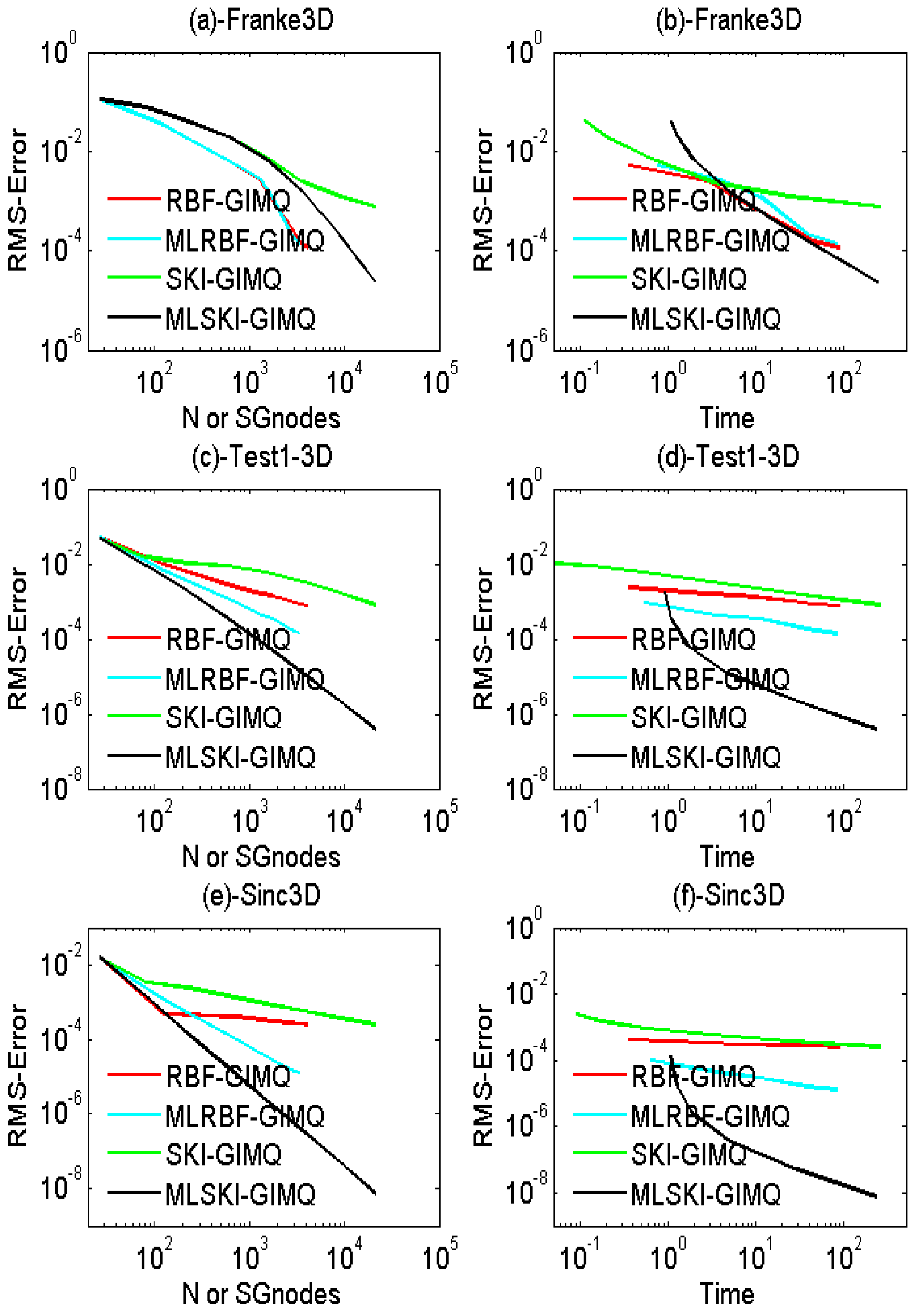


Figure 6.6: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using GIMQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 125,000 Halton points.

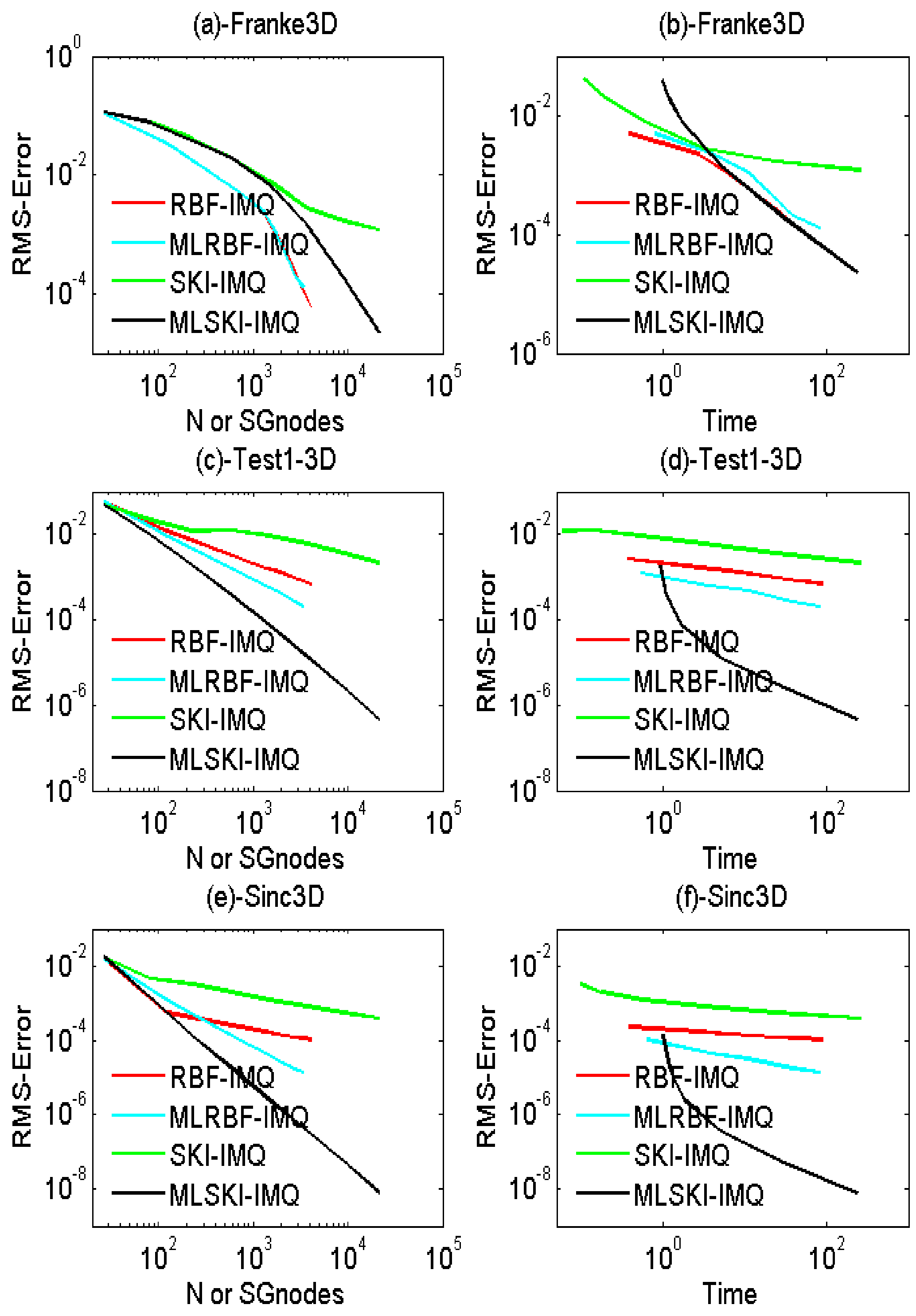


Figure 6.7: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using IMQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 125,000 Halton points.

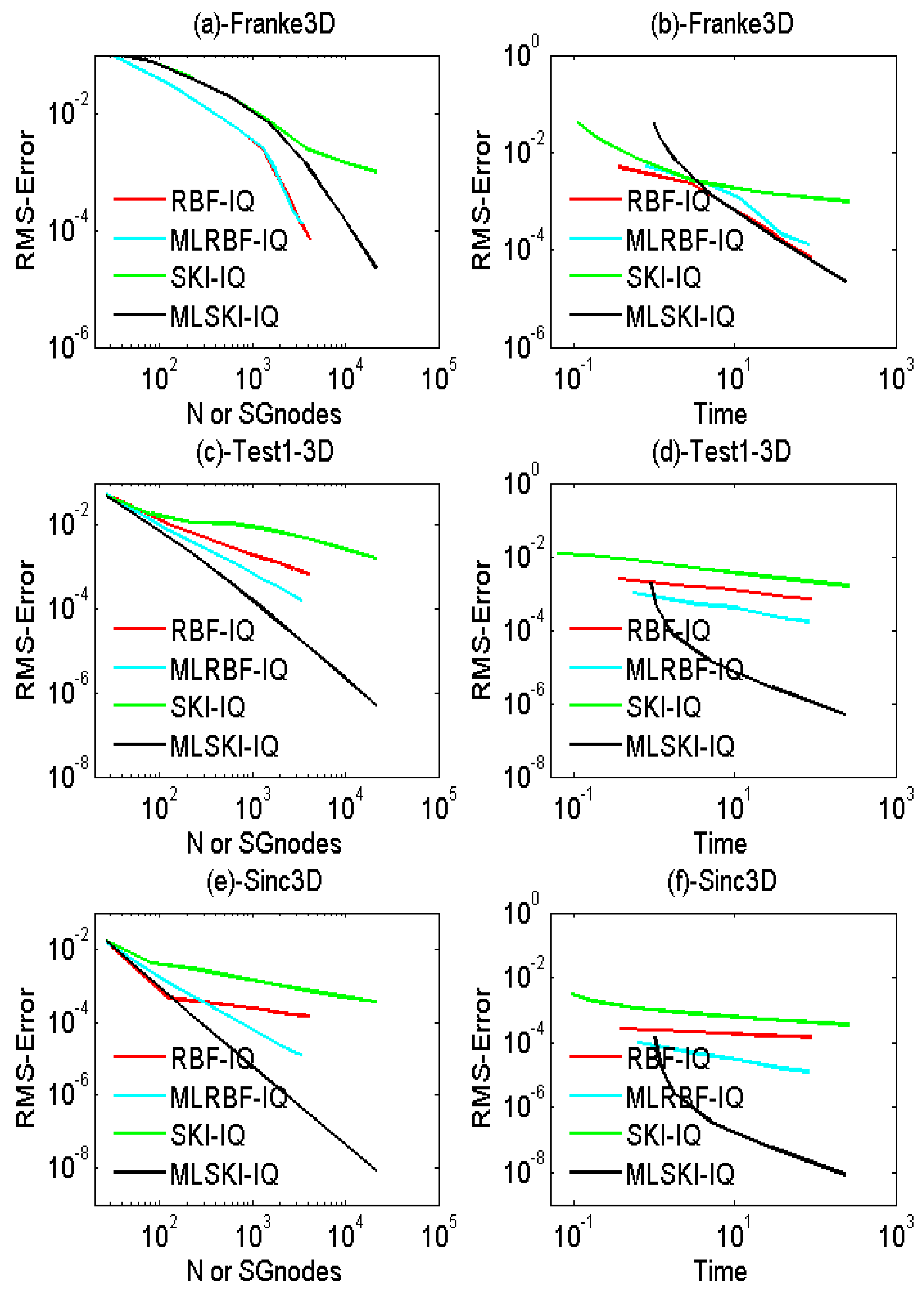


Figure 6.8: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using IQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers , evaluated at 125,000 Halton points.

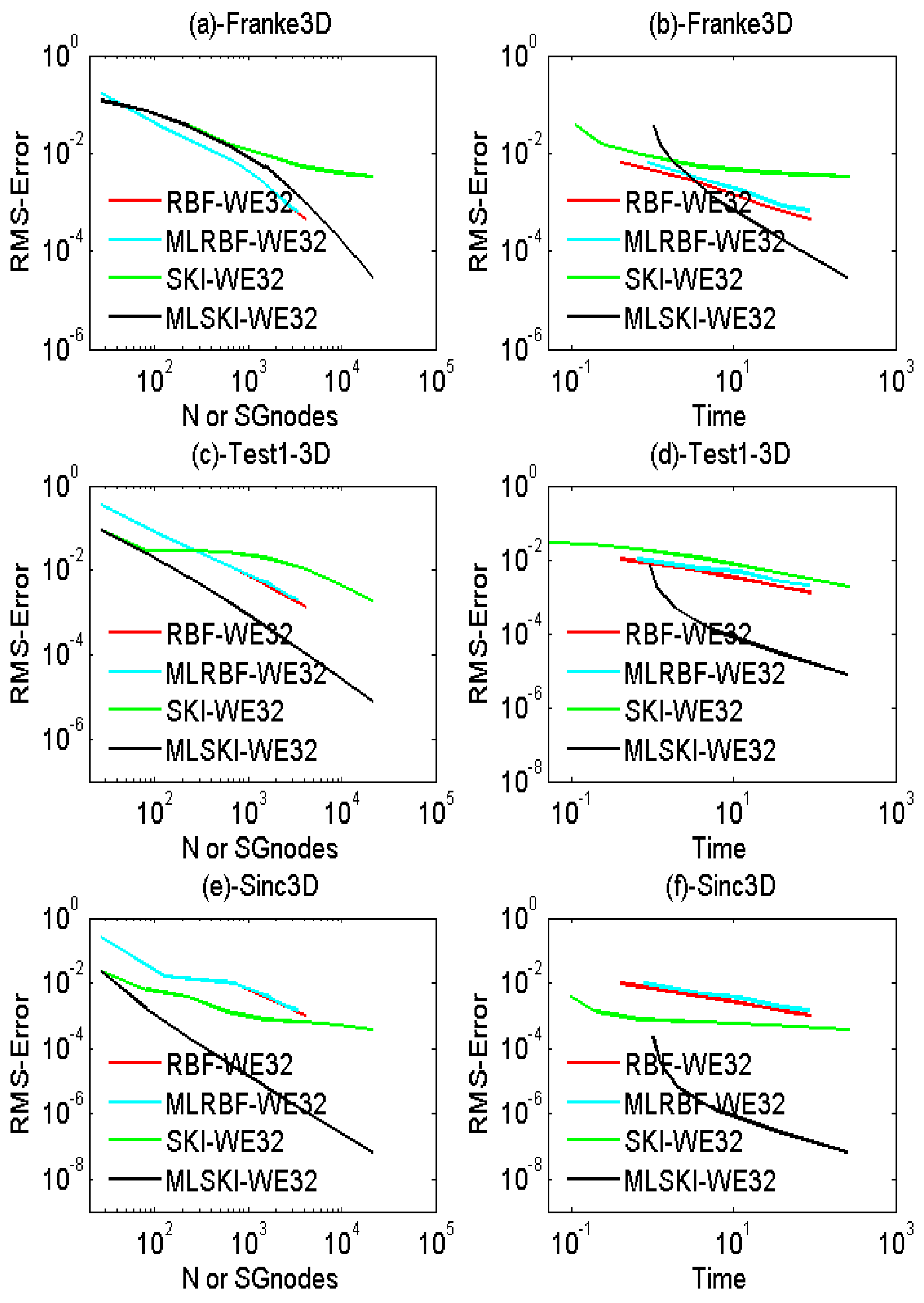


Figure 6.9: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using WE32 as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 125,000 Halton points.

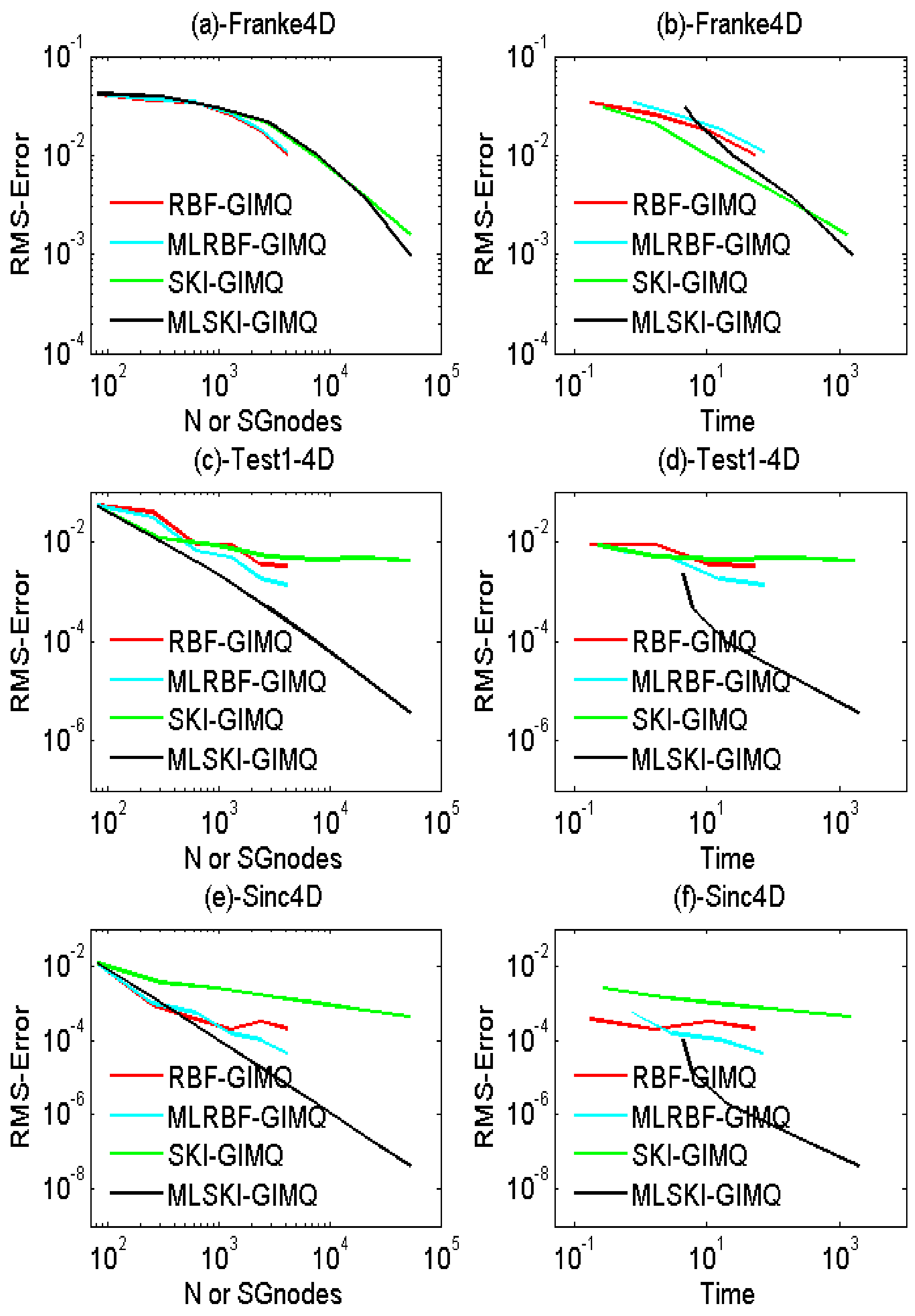


Figure 6.10: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using GIMQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 194,481 Halton points.

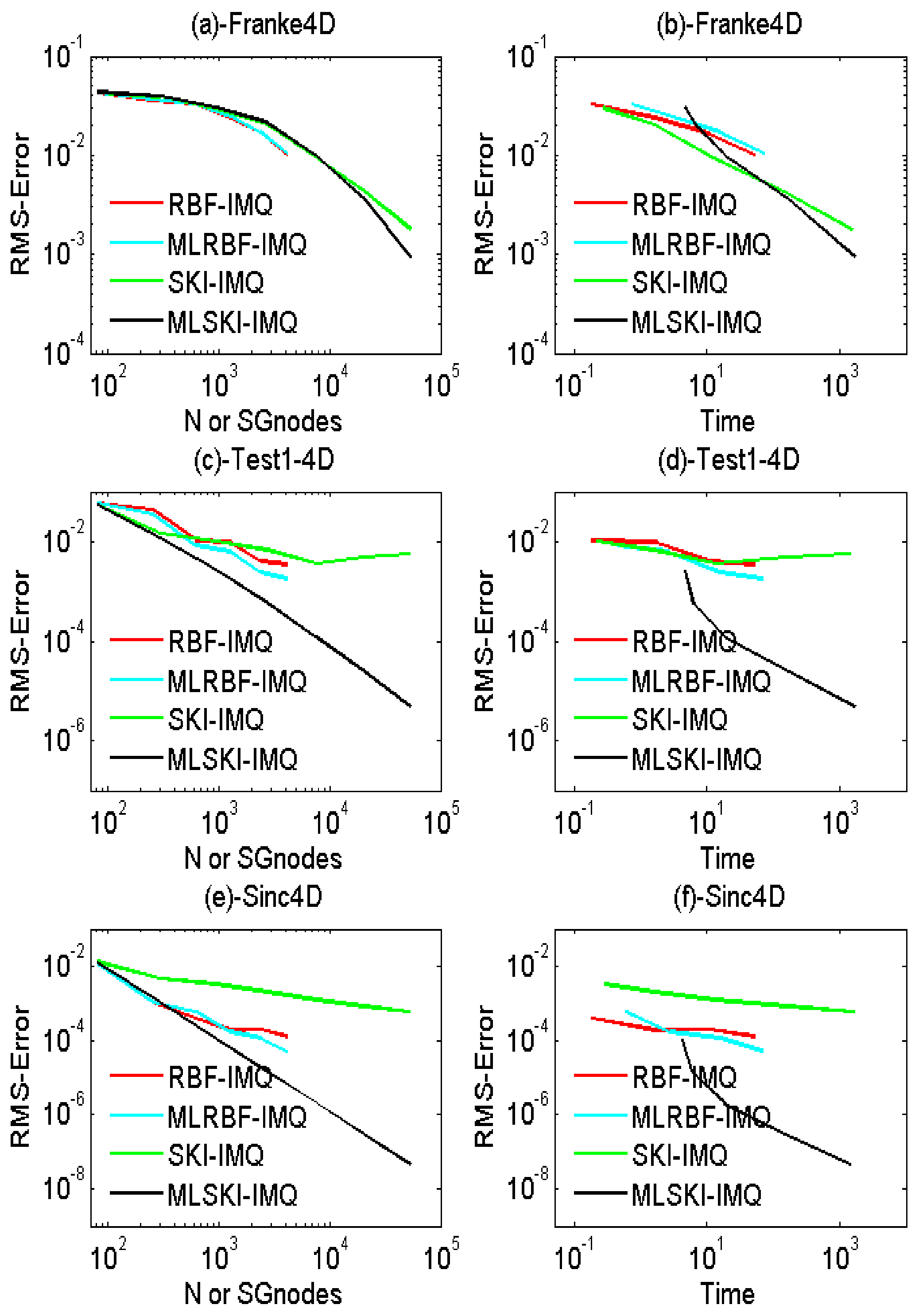


Figure 6.11: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using IMQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 194,481 Halton points.

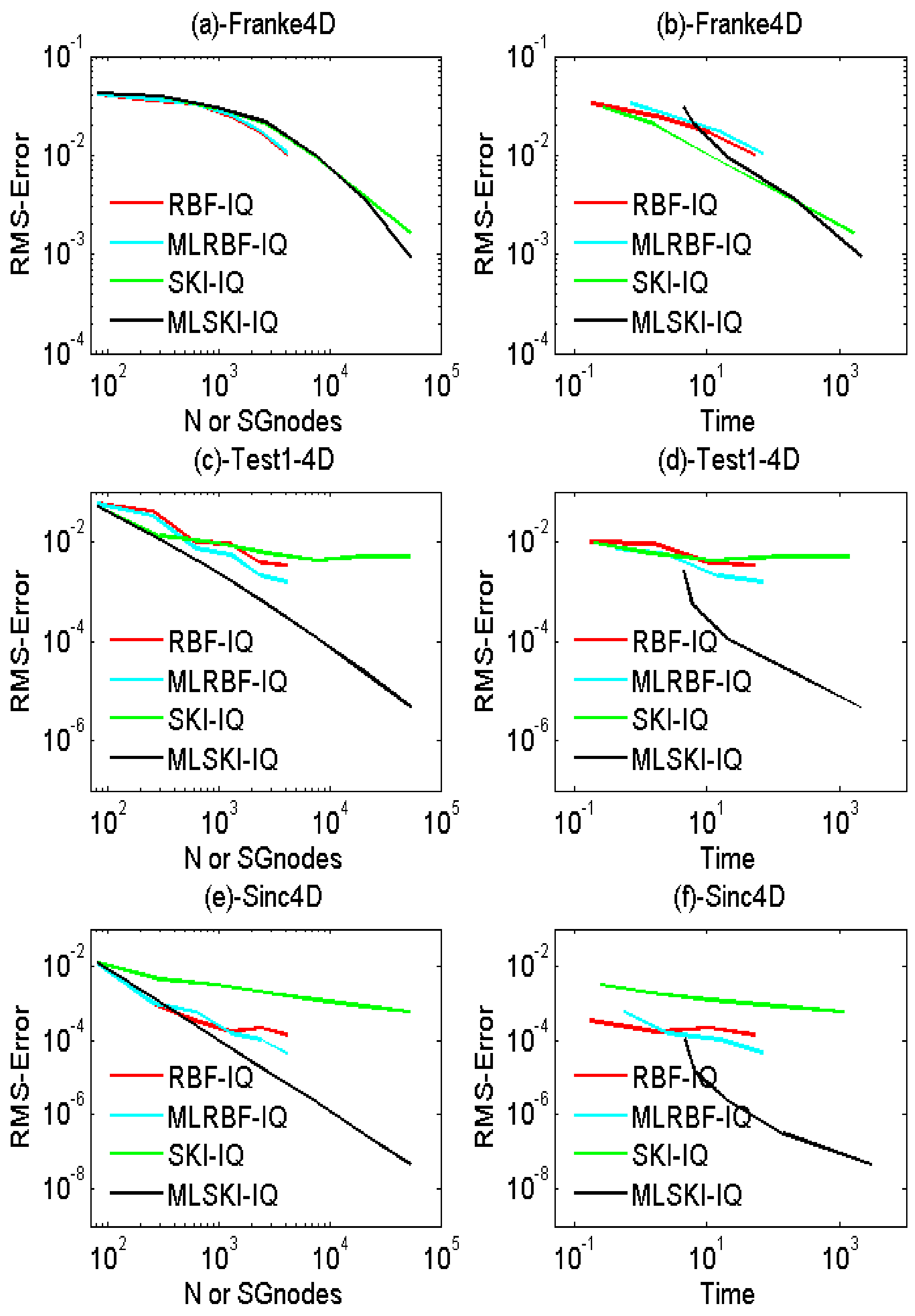


Figure 6.12: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using IQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 194,481 Halton points.

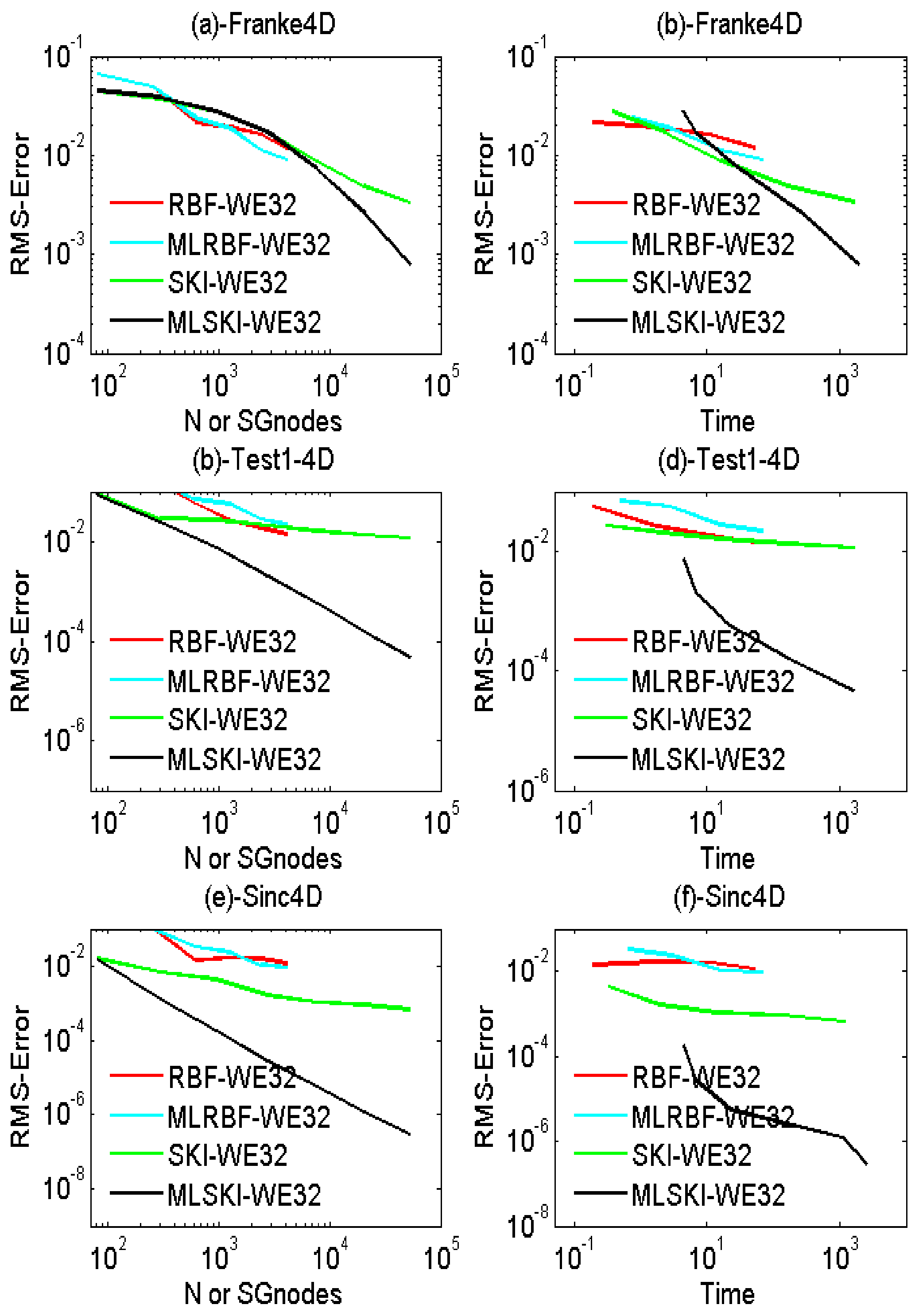


Figure 6.13: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using WE32 as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 194,481 Halton points.

6.3 Examples with conditionally positive definite RBFs

We repeat some of our numerical experiments SKI and MLSKI implementation in 3-dimensions with some of the CPD RBFs such as MQ, r^3, TPS2=$r^2 \log r$, TPS3=$r^4 \log r$, and TPS4=$r^6 \log r$.

6.3.1 3-variate interpolation

Using MQ as basis, the error comparison is given in Figures 6.14. These error for r^3 are compared in Figure 6.15, for TPS2 in Figure 6.16 and for TPS3 in Figure 6.17. The performance of the algorithm with MQ is nearly same that has been observed with the inverse multiquadrics in the previous section. The experiments with r^3, TPS2, TPS3 produce a comparatively slower convergence. This is because, these examples are nearly unstable. These experimental results are in agreement with the corresponding examples in 2-dimensions presented in Chapter 4 and Chapter 5.

6.3.2 4-variate interpolation

In this section we compare the 4-variate interpolants obtained by applying RBF and SKI along with their multilevel version by using CPD RBFs as basis function. The errors comparison for MQ is given in Figures 6.18, , for TPS2 in Figures 6.19 and for TPS3 in Figures 6.20. In these examples except the slow convergence for Sinc4D, MQ produce the same convergence observed in the lower dimensions. While larger condition numbers together with slower convergence for TPS2 and TPS3 are our observations in this section. Generally this performance of MLSKI with conditionally positive definite RBFs is a confirmation of the numerical results observed in the corresponding experiments of 3-variate interpolation of this chapter and bi-variate interpolation discussed in Chapter 4 and Chapter 5.

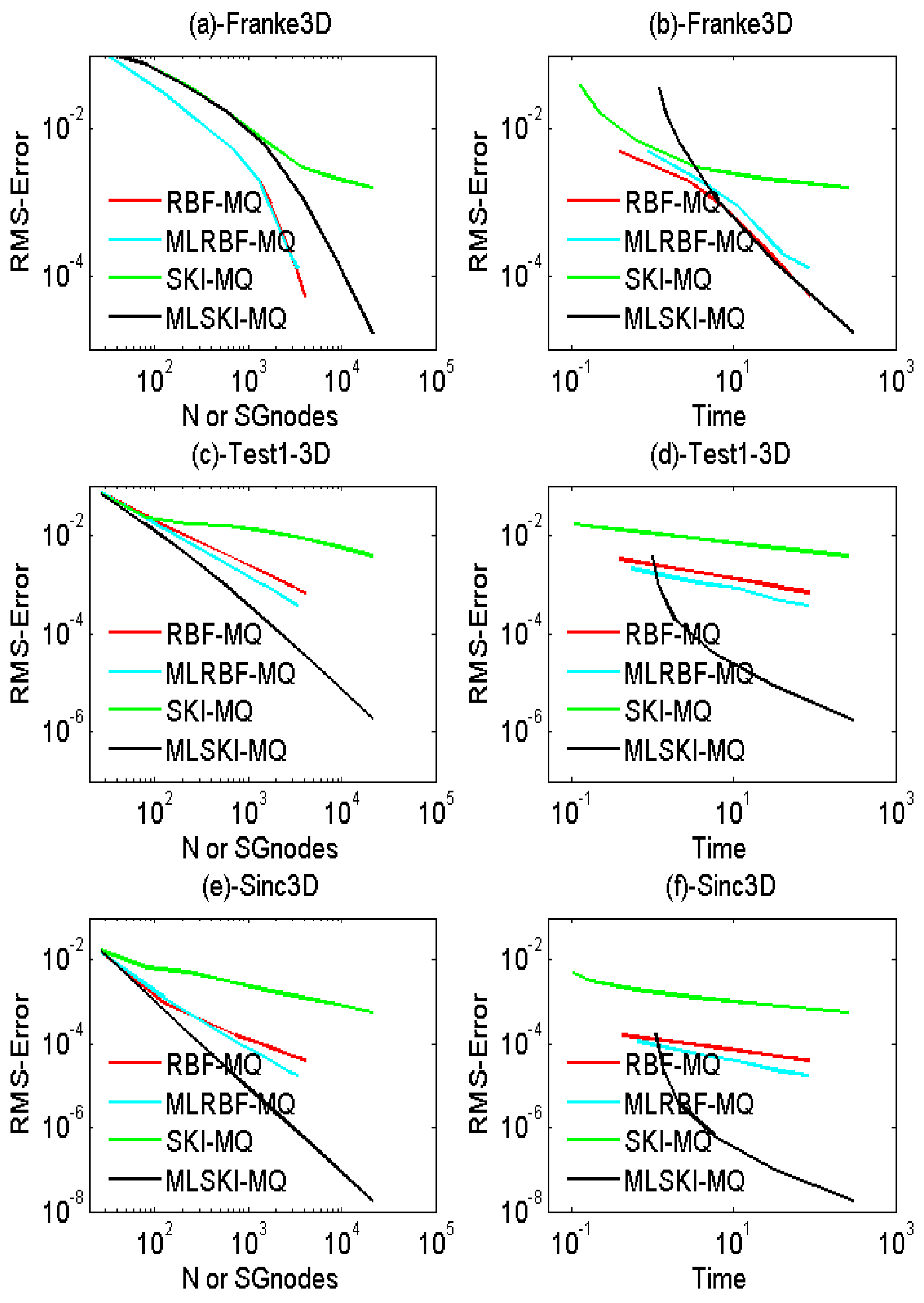


Figure 6.14: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using MQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 125,000 Halton points.

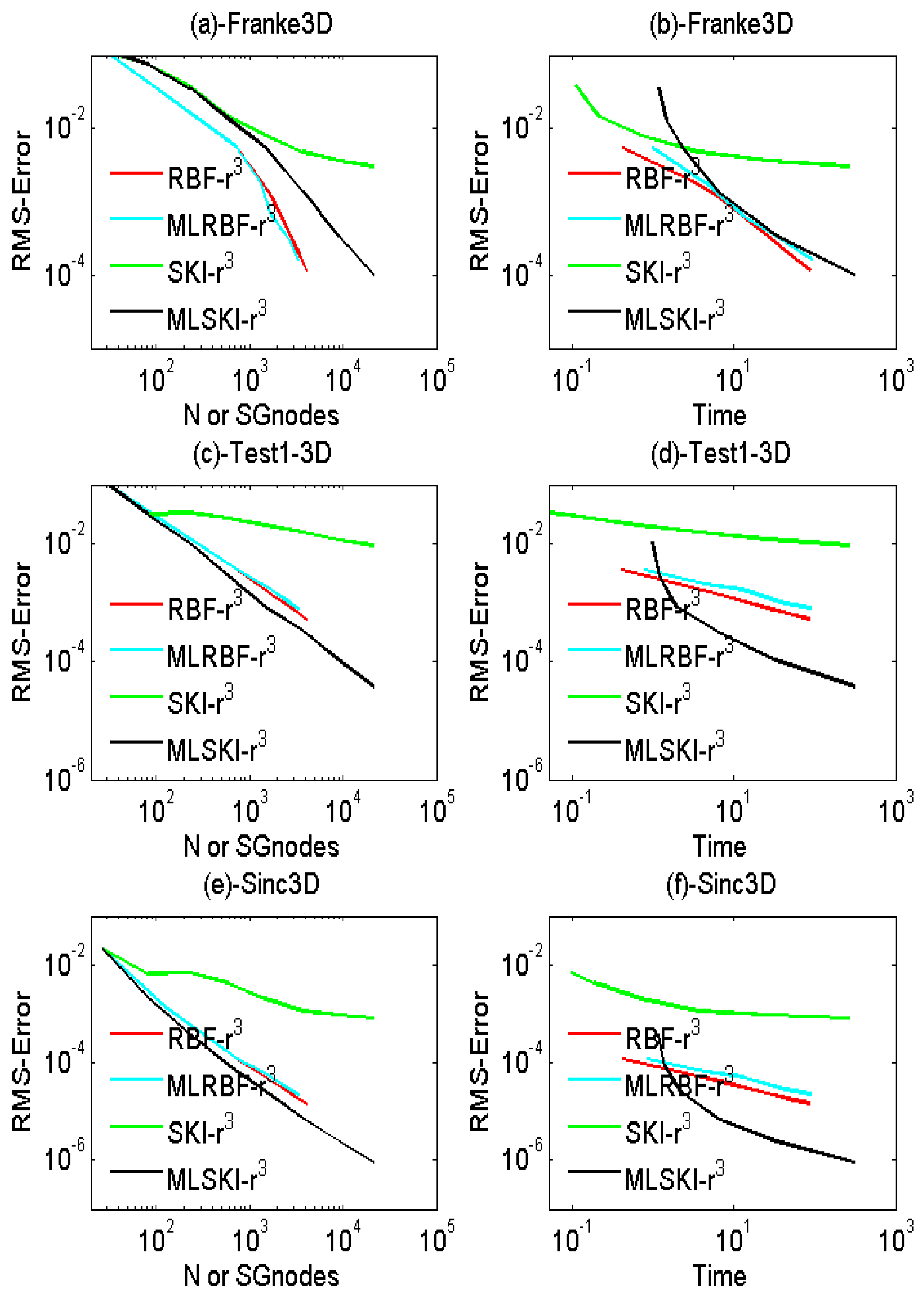


Figure 6.15: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using "r^3" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 125,000 Halton points.

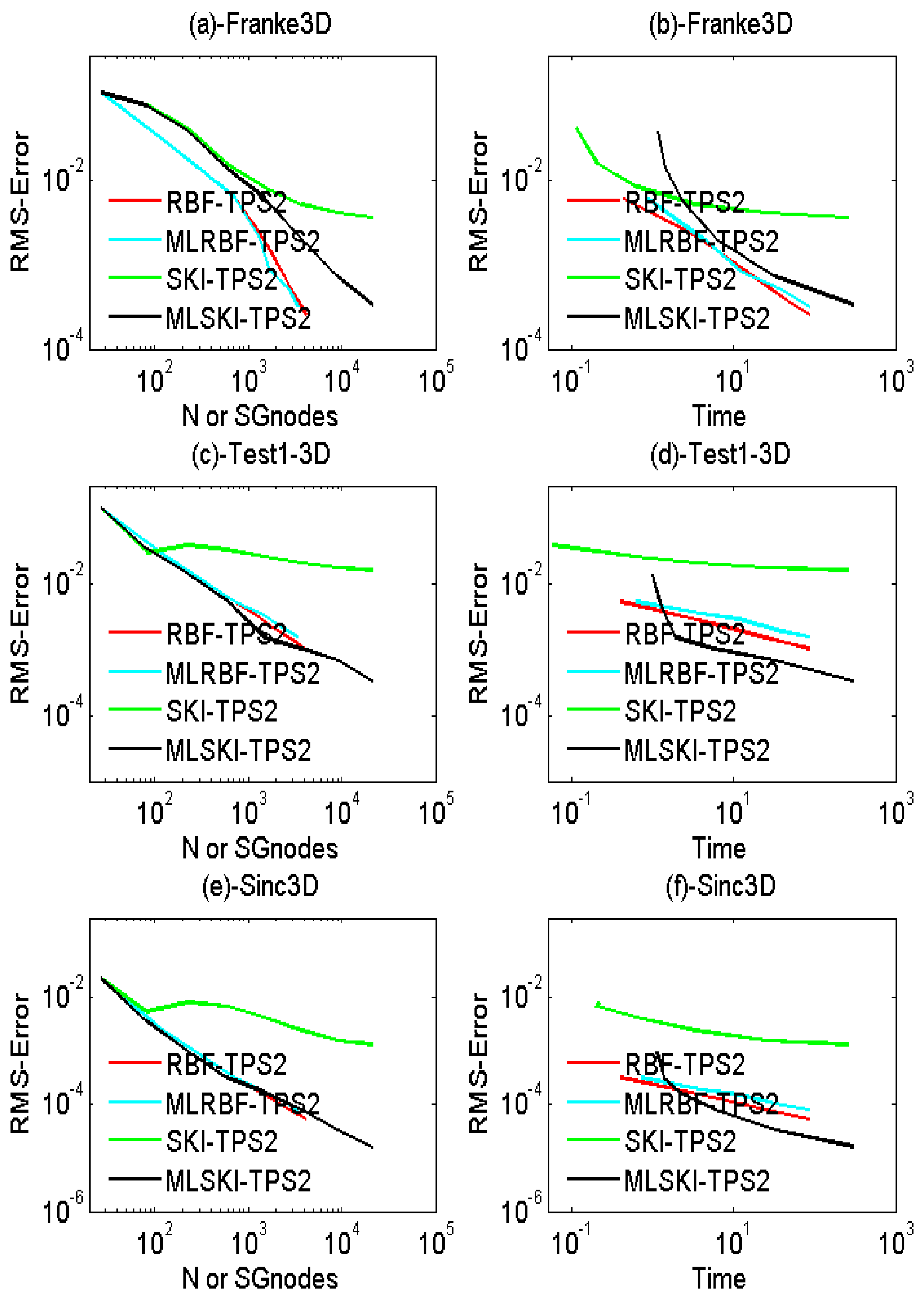


Figure 6.16: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using "$TPS2 = r^2 \log r$" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 125,000 Halton points.

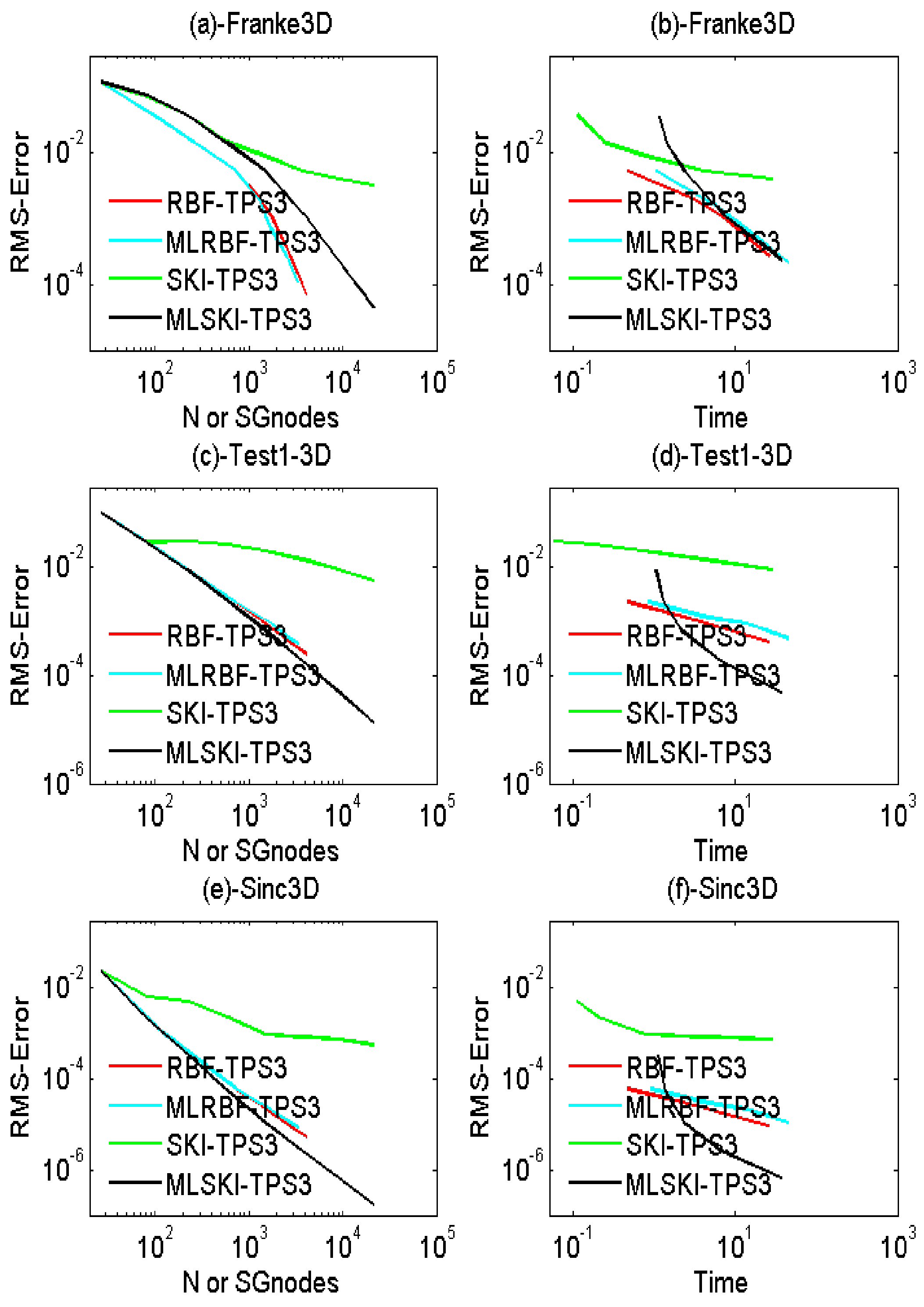


Figure 6.17: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using "$TPS3 = r^4 \log r$" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 125,000 Halton points.

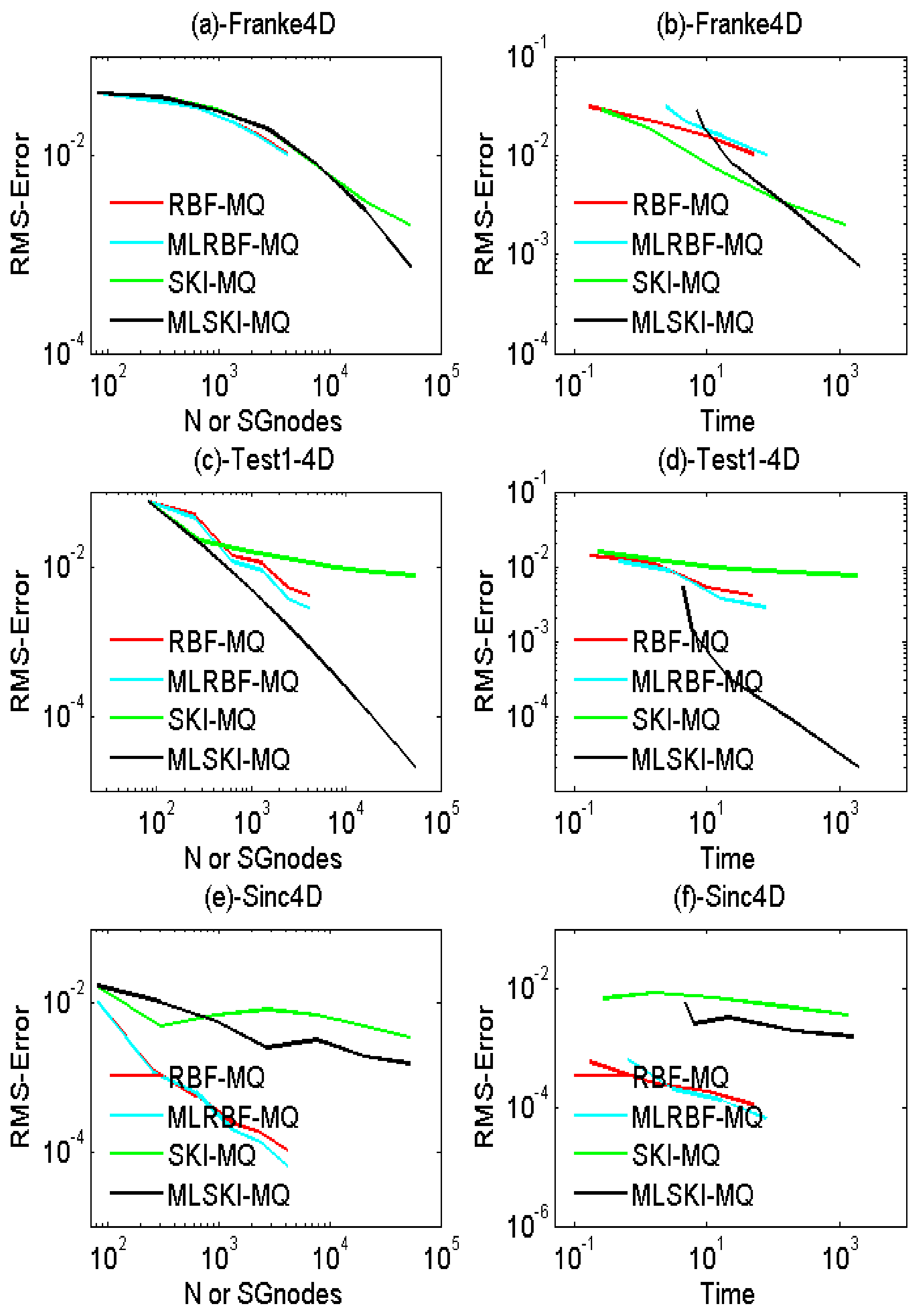


Figure 6.18: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using MQ as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), with safe condition numbers, evaluated at 194,481 Halton points.

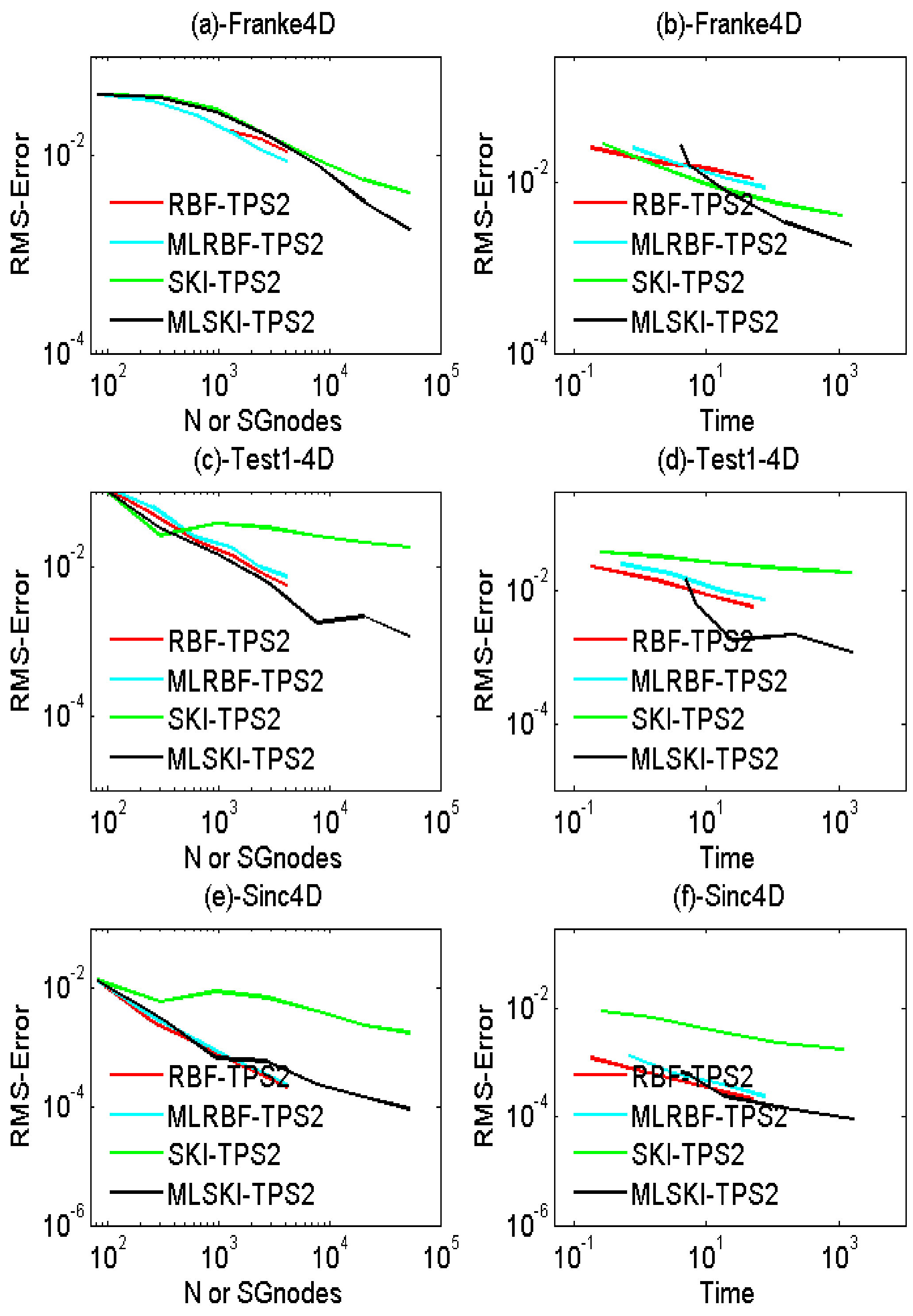


Figure 6.19: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using "$TPS2 = r^2 \log r$" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 194,481 Halton points.

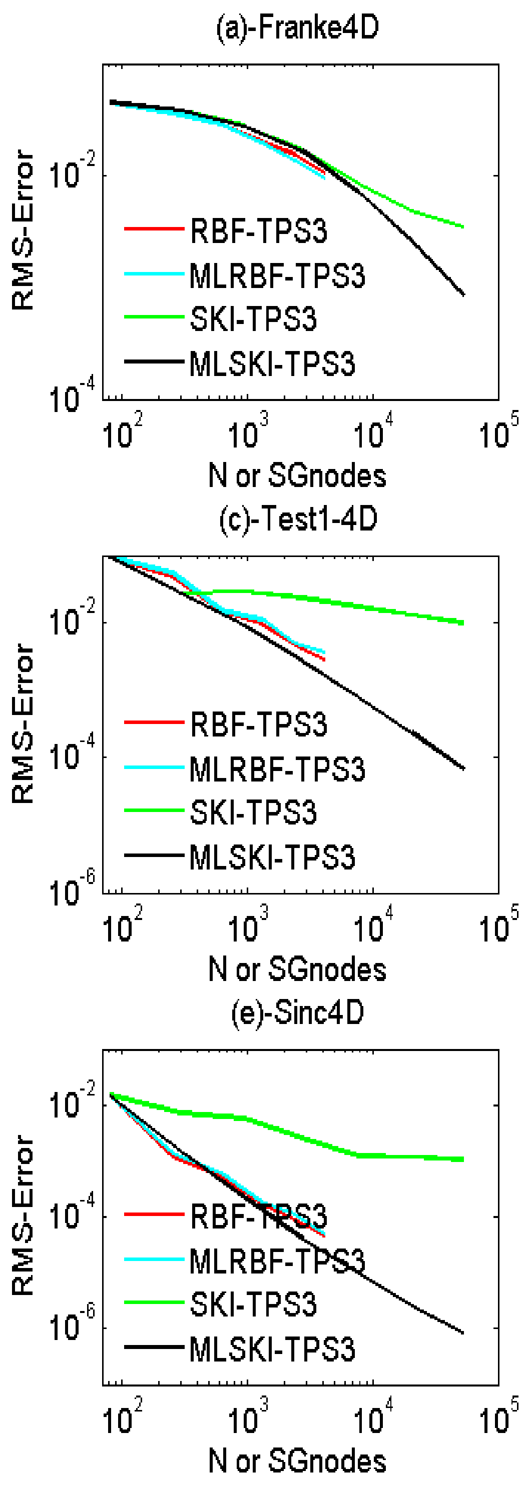


Figure 6.20: Left: RMS-error versus N (RBF) and SGnodes (SKI). Right: RMS-error versus CPU time. Using "$TPS3 = r^4 \log r$" as basis: SKI (Green), RBF (Red), MLSKI (Black) and multilevel RBF (Cyan), evaluated at 194,481 Halton points.

SGnode	DOFs(SG)	L_∞-Err	RMS Err	Err at Nodes	Cond no	Time
81	81	7.9105e-2	4.4589e-2	2.4940e-12	3.6544e+5	4.2305e-1
297	621	2.4067e-2	1.0677e-2	1.2195e-11	8.6224e+6	4.2266e-2
945	2943	1.9844e-2	6.3598e-3	1.8718e-12	1.0568e+6	2.4616e-1
2769	11139	5.6653e-3	1.2672e-3	5.2150e-11	3.6076e+8	1.6921e+0
7681	36901	4.7096e-3	8.2613e-4	1.5854e-12	1.4065e+7	1.4203e+1
20481	11211	1.3155e-3	1.5425e-4	4.0899e-11	1.4848e+10	1.6426e+2
52993	320675	1.1548e-3	1.0690e-4	9.0949e-13	1.2741e+8	1.7663e+3
133889	877655	3.2099e-4	1.9243e-5	1.9412e-11	6.0115e+11	1.6442e+4
331777	2322185	2.8385e-4	1.3934e-5	1.3358e-12	1.1542e+9	1.6964e+5

Table 6.3: SKI interpolation results from HPC ALICE, using Gaussian RBF with $c = \frac{2^{h_r}}{3}$, using test function Test1-4D, evaluated at 194,481 Halton points.

6.4 Discussion

The results presented in this chapter, show that SKI and MLSKI can be easily and successfully applied to interpolating a d-variate function for d=3, 4 with a wide range of RBFs as basis function. SKI and MLSKI with positive definite RBFs such as Gaussian, GIMQ, IMQ and IQ produce good convergence, whereby Gaussian has the best convergence among them. Among the CPD RBFs, the performance of MQ matches with the above mentioned inverse multiquadrics. While TPS2, TPS3 and r^3 have very low convergence as the algorithms with these RBFs become nearly unstable. Due to the growth of the problem size with dimensions, direct implementation of standard full grid RBF interpolation is limited e.g., to nearly 10 points in each dimension for $d = 4$. But in practice much more points may be required in each co-ordinate direction to achieve a desired accuracy. This makes it a difficult task to solve and/or achieve better convergence in high dimensions. The proposed SKI and MLSKI have nearly one-dimensional complexity and have the potential to solve large and high dimensional problems. SKI is superior in run time and complexity to the direct RBF interpolation. Hence, the algorithm can solve a d-variate interpolation problem on sparse grid containing 114,690 for $d = 3$ and 331,780 for $d = 4$, respectively. While this number in our classical RBF interpolation on the same machine is only nearly 15,000 regardless d. Though, this superiority may be at the cost of losing some accuracy in some particular cases, but in most of our stable experiments better convergence has been observed. In general, SKI achieves better convergence in less time as compared to the classical RBF approach. An accelerated convergence may be a requirement of the given problem, we have shown that MLSKI is capable of providing a faster convergence. MLSKI has shown a faster convergence than the multilevel RBF interpolation. We observe that both SKI and MLSKI generally outperforms their direct interpolation counterparts in terms of complexity, run time and convergence by an encouraging margin in particular when it comes to high dimensions.

Chapter 7

Conclusions and future work

7.1 Conclusions

We have presented an algorithm for interpolation on structured or mildly unstructured data sets in $\mathbb{R}^d$, for $d \geq 2$. Implementation of radial basis functions is insensitive to the dimension parameter d and sparse grid methods address the complexity issue in high dimensions. The sparse kernel-based interpolation algorithm presented in Chapter 4, couples the power of radial basis function and the complexity of sparse grid methods. SKI is inspired by the combination technique of sparse grid/hyperbolic cross methods. In this scheme, the sparse grid is represented as the union of full and anisotropic grids. We use the spaces of anisotropic radial basis functions $\varphi_{A_\mathbf{l}}(\cdot)$ to solve the constituent full grid interpolation problems and, correspondingly, the SKI space is equal to the direct sum of anisotropic RBF spaces $\{\varphi_{A_\mathbf{l}}(\cdot) : |\mathbf{l}|_1 = n, \cdots, n+d-1\}$. In the multilevel sparse kernel based interpolation algorithm, we follow the multilevel interpolation scheme of Floater and Iske [35]. The multilevel sparse kernel based interpolation makes use of the nested property of the sparse grids from lower to higher level for the decomposition of the data to apply the Floater and Iske scheme to SKI. MLSKI is using SKI at each step and has, therefore, linear complexity and run time. The multilevel sparse kernel-based interpolation accelerates the convergence. It is faster than the RBF interpolation and the multilevel RBF interpolation and is superior in convergence than the direct SKI, RBF and multilevel RBF schemes.

The combination technique approach comes at the cost of some overhead, as typically the component spaces include some redundancy as some of the data sites are visited several times. However, the parallel nature of solving small but full grid interpolation problems makes SKI and MLSKI algorithm attractive for high dimensional interpolation and makes it perfectly fit for parallel computing. Moreover, the parallel nature of solving small but independent interpolation problem results in the reduction of computation and storage requirements even when applied on the same computer in serial. This advantage becomes more prominent as the dimension of the problem increases. In addition, SKI uses an existing computer code for RBF interpolation to solve the constituent anisotropic interpolation problems, hence making the coding relatively straight forward.

The bi-variate numerical results presented in Chapters 4, 5 confirm that MLSKI is generally superior to the classical RBF interpolation on gridded data. The SKI visits some nodes more than once, but still faster when machine time is considered as a function of the sparse grid nodes irrespective of the fact that this redundancy is included or excluded. SKI mostly outperforms RBF in terms of accuracy as function of machine time. As far as the computations remain stable, the convergence is also faster if considered as a function of the input data size. In the unstable regime (i.e., when the maximum condition number $\geq 10^{10}$), we still observe good accuracy of MLSKI, but the error at the centers is evaluated to be of the order of the interpolation error. This indicates a possible loss of accuracy due to instabilities, which could potentially make the computation unreliable. When implemented on an ordinary computer "Core 2 Duo CPU @ 3.16GHz 3.17GHz and 3.24GB of RAM", SKI effectively and successfully computes on a sparse grid of level up to 10, having 13,313 nodes. Some simulations of the proposed scheme have been performed on one of the computer nodes (having a pair of quad-core 2.67GHz Intel Xeon X5550 CPUs and 12GB of RAM) of the computers cluster called ALICE of the *University of Leicester*. There, SKI and MLSKI turns out to be stable for larger data sizes and computations can be done on sparse grids of levels up to 12, having more than 60,000 nodes. On the other hand, for classical RBF interpolation ALICE can deal with nearly 15,000, regardless of the dimension. So the MLSKI algorithm gives generally better efficiency in terms of stability, accuracy, complexity and the run time over the classical RBF method.

Due to its nearly one-dimensional complexity, SKI and hence MLSKI is capable of solving large interpolation problems in higher dimensions. The algorithm has also been successfully implemented for interpolation of d-variate function in $\mathbb{R}^d$ for d=3, 4. These high dimensional numerical results reconfirm the superior performance of the scheme, as was observed in the 2-dimensional experiments. On the ordinary machine mentioned above, the scheme can solve interpolation problem on sparse grids $\mathfrak{Y}^s_{8,3}$ and $\mathfrak{Y}^s_{7,4}$ of size 21,249 and 52,993 in $\mathbb{R}^3$ and $\mathbb{R}^4$ respectively. On a single node of ALICE these numbers go up to $\mathfrak{Y}^s_{10,3}$ and $\mathfrak{Y}^s_{9,4}$, that is size 114,690 and 331,780 nodes in $\mathbb{R}^3$ and $\mathbb{R}^4$. The corresponding full grids will have sizes of order 10^9 in $\mathbb{R}^3$ and 10^{11} in $\mathbb{R}^4$. The numerical results presented in Chapter 4, Chapter 5, and Chapter 6, show that our algorithm is not only faster but also generally superior in terms of convergence (it might be slower in some cases if error is considered as a function of the input data size), stability and complexity than the classical RBF interpolation in $\mathbb{R}^d$ for $d \geq 2$.

We have used a desktop computer "Core 2 Duo CPU @ 3.16GHz 3.17GHz and 3.24GB of RAM" for d=2, 3. For d=4, we use ALICE to be able to run many experiments at the same time by accessing as many nodes of ALICE. Each experiment was run in serial on a single node (having a pair of quad-core 2.67GHz Intel Xeon X5550 CPUs and 12GB of RAM) which is equivalent to a more powerful desktop computer. Thus all the CPU timing and problem size shown here relate to computations that can be run on an ordinary computer computer mentioned before. However, it is worth mentioning that the largest size of the problem in each of these computation has been kept the same for which all these schemes can be implemented on the smaller computer of 3GM RAM mentioned above. Moreover, we have reported separately the largest size of the problem that SKI can solve on a single node of ALICE for d =2, 3, 4 in Sections 4.3.2, 5.4.1.1(a), and in Tables 6.1, 6.2, 6.3.

The MLSKI algorithm was first implemented with positive definite RBFs such as Gaussian and inverse multi quadrics. The best convergence is observed for Gaussian. MLSKI performance with inverse multi quadrics is also good but it is slower than the convergence with Gaussian. The superior convergence of Gaussians over inverse multi-quadrics observed in this thesis may be due to the high algebraic/spectral convergence orders [94] in the context of direct RBF interpolation. The scheme has also been implemented with the conditionally positive definite RBFs such as multi-quadrics, poly-harmonics, radial powers etc. Among the CDP RBFs, MQ produces nearly the same convergence as the inverse multi-quadrics, while for others the convergence is either very slow or is not present.

Experimental results show that SKI is not only convergent for regular sparse grids, but its convergence on irregular sparse grids has also been observed. These numerical experiments are performed on perturbed sparse grids of level n, where the perturbation is kept proportional to a quarter of 2^{-n}. The convergence for perturbed sparse grid was found to be slightly slower than the corresponding convergence on regular sparse grids. For perturbation greater than $2^{-n}/4$, convergence is slower while for a perturbation smaller than $2^{-n}/4$ convergence approaches the one on regular sparse grids.

In our experiments, we have implemented the naïve RBF approach to solve the partial interpolation problems in the SKI scheme. Thus, it is fair to compare our method with the naïve RBF interpolation method, as we shall do through out all the experiments reported in this thesis. We remark that the fast RBF methods mentioned in the introduction can be applied to accelerate SKI. The implementations of the proposed methods in all our experiments, on an ordinary computer as well as on ALICE, have been performed in *Matlab* and in a non parallel environment. Since the proposed method is perfect for implementation in parallel fashion on modern high performance computing (HPC) systems, its generally superior performance in terms of run time and complexity could become more visible if implemented in parallel.

7.2 Outlook and future work

The main focus of this thesis has been on the development of SKI method and its implementation to address the complexity issue faced by interpolation problem in high dimensions. Its success has been verified through numerical experiments. In addition, the multilevel version of SKI has been successful in accelerating its convergence.

Our next task in the near future is to look into the theory of the SKI algorithm. As the approximation space at each level in the SKI scheme is a direct sum of anisotropic RBF spaces, one possible direction to look into the error analysis could be to combine idea for the analysis of anisotropic interpolation presented in [6] with the methods of proofs for the convergence analysis of the sparse grid combination technique given in [12] and [13].

The error analysis of SKI could be further extended to analyse MLSKI. One possible way to do this, could be following the method used for the error estimates of classical multilevel interpolation presented in [55].

Further questions that need to be answered are the following: why the convergence of SKI performance is the best with Gaussian? SKI still converges with positive definite RBFs (e.g., inverse multiquadrics) but convergence is slightly slower than Gaussians. When applied with conditionally positive definite RBFs of order $m > 1$, convergence is either very slow or there is no convergence at all. MLSKI still shows convergence using conditionally positive definite RBFs of order $m = 1$ (e.g., multiquadric RBF)? One possible reason that could explain these issues, might be the polynomial part of the anisotropic RBF interpolant $S_{A_{\mathbf{l}}}(\cdot)$ on the constituent small grids $\mathfrak{X}_{\mathbf{l}}$ in SKI algorithm and hence in its MLSKI partner. The tensor product nature of Gaussian could provide further explanation to its best performance over the non-Gaussian RBFs. Another fact that explains the slower convergence of inverse multiquadrics is their similar behaviour in the context of standard RBF interpolation in comparison with Gaussian RBF.

Another question we need to answer is the errors of the schemes at the interpolation centres. SKI with Gaussian, gives the error at sparse grid nodes nearly equal to the machine zero as long as the condition number is small. This behaviour of SKI by using Gaussians, is identical to that of the standard RBF interpolation schemes. On the other hand, when SKI is implemented with non Gaussians RBF, unlike the standard RBF methods, this error is not equal to zero and remains nearly the same as its general error at any point. We expect that most of these question could be answered, if the error analysis of the scheme is made possible.

SKI is solving $\mathcal{O}(n^{d-1})$ independent problems of one dimensional complexity $\mathcal{O}(2^n)$ and we therefore look forward to apply SKI and MLSKI high dimensions say d=10, 20.

We have mentioned that the algorithms work only for structured and mildly unstructured data. We intend to extend the algorithms for their application to a truly scattered data by using it together with the idea of local interpolation.

Another area of our interest is applying the SKI to RBF methods for the solution of partial differential equations (PDE) such as Kansa's non-symmetric RBF collocation scheme (also called multiquadric collocation method)or Fasshauer's symmetric collocation approach, in particular, for high dimensions. We have started the foundation work to apply it to the collocation methods, but it would be too early at this stage, to say something about any success or failure in this direction.

As mentioned in the literature overview of this work, sparse grids were introduced to circumvent the curse of dimensionality in the finite element methods (FEM) and finite difference method (FDM) for the solution of PDEs and were based on tensor product multi-linear approximation spaces. SKI is based on anisotropic RBFs spaces and could therefore have the potentials to accelerate the convergence of FEM/FDM. So another area of interest is to applying the SKI methods in the context of Galerkin methods for the solution of partial differential equations.

SKI is solving $\mathcal{O}(n^{d-1})$ independent problems of one dimensional complexity $\mathcal{O}(2^n)$ and we therefore look forward to apply it for dimensions say d=10, 20.

Bibliography

[1] N. Arad, N. Dyn, and D. Reisfeld. Image warping by radial basis functions: applications to facial expressions. *CVGIP: Graph. Models Image Process.*, 56(2):161–172, 1994.

[2] K. I. Babenko. Approximation by trigonometric polynomials in a certain class of periodic functions of several variables. *Soviet Math. Dokl.*, 1:672–675, 1960.

[3] G. Baszenski. nth order polynomial spline blending. In *Multivariate approximation theory, III (Oberwolfach, 1985)*, volume 75 of *Internat. Schriftenreihe Numer. Math.*, pages 35–46. Birkhäuser, Basel, 1985.

[4] M. Bauer, O. Buchtala, H. Timo, K. Ralf, S. Bernhard, and W. Robert. Technical data mining with evolutionary radial basis function classifiers. *Applied Soft Computing.*, 9:765–774, 2009.

[5] B. J. C. Baxter. Conditionally positive functions and p-norm distance matrices. *Constr. Approx.*, 7(4):427–440, 1991.

[6] R. Beatson, O. Davydov, and J. Levesley. Error bounds for anisotropic RBF interpolation. *J. Approx. Theory*, 162(3):512–527, 2010.

[7] R. K. Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration. *Adv. Comput. Math.*, 11(2-3):253–270, 1999.

[8] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740 (electronic), 2000.

[9] R. Belmann. *Adaptive Control process: a guide tour.* Princeton Universoty Press, Princeton, 1961.

[10] D. Brown, L. Ling, E. Kansa, and J. Levesley. On approximate cardinal preconditioning methods for solving pdes with radial basis functions. *Engineering Analysis with Boundary Elements*, 29(4):343–353, 2005. Mesh Reduction Methods - Part III.

[11] H.-J. Bungartz and M. Griebel. A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives. *J. Complexity*, 15(2):167–199, 1999.

[12] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Two proofs of convergence for the combination technique for the efficient solution of sparse grid problems. In *Domain decomposition methods in scientific and engineering computing (University Park, PA, 1993)*, volume 180 of *Contemp. Math.*, pages 15–20. Amer. Math. Soc., Providence, RI, 1994.

[13] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. A proof of convergence for the combination technique for the Laplace equation using tools of symbolic computation. *Math. Comput. Simulation*, 42(4-6):595–605, 1996. Symbolic computation, new trends and developments (Lille, 1993).

[14] H.-J. Bungartz, M. Griebel, and U. Rüde. Extrapolation, combination, and sparse grid techniques for elliptic boundary value problems. *Comput. Methods Appl. Mech. Engrg.*, 116(1-4):243–252, 1994. ICOSAHOM'92 (Montpellier, 1992).

[15] R. E. Carlson and T. A. Foley. The parameter $\mathbf{R}^2$ in multiquadric interpolation. *Comput. Math. Appl.*, 21(9):29–42, 1991.

[16] R. E. Carlson and T. A. Foley. Interpolation of track data with radial basis methods. *Comput. Math. Appl.*, 24(12):27–34, 1992. Advances in the theory and applications of radial basis functions.

[17] R. E. Carlson and B. K. Natarajan. Sparse approximate multiquadric interpolation. *Comput. Math. Appl.*, 27(6):99–108, 1994.

[18] J. C. Carr, W. R. Fright, and R. K. Beatson. Surface interpolation with radial basis functions for medical imaging. *IEEE Transactions on Medical Imaging*, 16:96–107, 1997.

[19] G. Casciola, D. Lazzaro, L. B. Montefusco, and S. Morigi. Shape preserving surface reconstruction using locally anisotropic radial basis function interpolants. *Comput. Math. Appl.*, 51(8):1185–1198, 2006.

[20] G. Casciola, L. B. Montefusco, and S. Morigi. The regularizing properties of anisotropic radial basis functions. *Appl. Math. Comput.*, 190(2):1050–1062, 2007.

[21] T. F. Chan and D. E. Foulser. Effectively well-conditioned linear systems. *SIAM J. Sci. Statist. Comput.*, 9(6):963–969, 1988.

[22] C. S. Chen, C. A. Brebbia, and H. Power. Dual reciprocity method using compactly supported radial basis functions. *Comm. Numer. Methods Engrg.*, 15(2):137–150, 1999.

[23] C. S. Chen, M. Ganesh, M. A. Golberg, and A. H.-D. Cheng. Multilevel compact radial functions based computational schemes for some elliptic problems. *Comput. Math. Appl.*, 43(3-5):359–378, 2002. Radial basis functions and partial differential equations.

[24] C. S. Chen, M. D. Marcozzi, and S. Choi. The method of fundamental solutions and compactly supported radial basis functions: a meshless approach to 3D problems. In *Boundary elements, XXI (Oxford, 1999)*, volume 6 of *Int. Ser. Adv. Bound. Elem.*, pages 561–570. WIT Press, Southampton, 1999.

[25] F.-J. Delvos. *d*-variate Boolean interpolation. *J. Approx. Theory*, 34:99–114, 1982.

[26] M. R. Dubal. Domain decomposition and local refinement for multiquadric approximations. I. Second-order equations in one-dimension. *J. Appl. Sci. Comput.*, 1(1):146–171, 1994.

[27] N. Dyn, D. Levin, and S. Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM J. Sci. Statist. Comput.*, 7(2):639–659, 1986.

[28] G. E. Fasshauer. Solving partial differential equations by collocation with radial basis functions. In A. Le Méhauté, C. Rabut, and L. L. Shumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 131–138, Nashville, TN, 1997. Vanderbil University Press.

[29] G. E. Fasshauer. Hermite interpolation with radial basis functions on spheres. *Adv. Comput. Math.*, 10(1):81–96, 1999.

[30] G. E. Fasshauer. Solving differential equations with radial basis functions: multilevel methods and smoothing. *Adv. Comput. Math.*, 11(2-3):139–159, 1999. Radial basis functions and their applications.

[31] G. E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007. With 1 CD-ROM (Windows, Macintosh and UNIX).

[32] G. E. Fasshauer and Mccourt M. J. Stable evaluation of gaussian rbf interpolants. *Submitted*, 2011.

[33] G. E. Fasshauer and J. W. Jerome. Multistep approximation algorithms: improved convergence rates through postconditioning with smoothing kernels. *Adv. Comput. Math.*, 10(1):1–27, 1999.

[34] G. E. Fasshauer and J. G. Zhang. Preconditioning of radial basis function interpolation systems via accelerated iterated approximate moving least squares approximation. In *Progress on meshless methods*, volume 11 of *Comput. Methods Appl. Sci.*, pages 57–75. Springer, New York, 2009.

[35] M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *J. Comput. Appl. Math.*, 73(1-2):65–78, 1996.

[36] J. Flusser. An adaptive method for image registration. *Pattern Recognition.*, 25(1):45–54, 1992.

[37] T. A. Foley. Interpolation and approximation of 3-D and 4-D scattered data. *Comput. Math. Appl.*, 13(8):711–740, 1987.

[38] B. Fornberg, E. Larsson, and N. Flayer. Stable compuattion with gaussain radial basis functions. *SIAM J. Sci. Comput.*, 33(2):869–892, 2011.

[39] B. Fornberg and C. Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM J. Sci. Comput.*, 30(1):60–80, 2007/08.

[40] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Comput. Math. Appl.*, 48(5-6):853–867, 2004.

[41] R. Franke. Scattered data interpolation: tests of some methods. *Math. Comp.*, 38(157):181–200, 1982.

[42] A. Gaikwad and I. M. Toke. Gpu based sparse grid technique for solving multidimensional options pricing pdes. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, WHPCF '09, pages 6:1–6:9, New York, NY, USA, 2009. ACM.

[43] J. Garcke. A dimension adaptive sparse grid combination technique for regression. 2011. accepted.

[44] J. Garcke and M. Griebel. On the parallelization of the sparse grid approach for data mining. In S. Margenov, J. Wasniewski, and P. Yalamov, editors, *Large-Scale Scientific Computations, Third International Conference, LSSC 2001, Sozopol, Bulgaria*, volume 2179 of *Lecture Notes in Computer Science*, pages 22–32. Springer, 2001. also as SFB 256 Preprint 721, Universität Bonn, 2001.

[45] J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1-2):1–25, April 2009.

[46] J. Garcke, M. Hegland, and O. Nielsen. Parallelisation of sparse grids for large scale data analysis. *ANZIAM Journal*, 48(1):11–22, 2006.

[47] F. Girosi. Some extensions of radial basis functions and their applications in artificial intelligence. *Comput. Math. Appl.*, 24(12):61–80, 1992. Advances in the theory and applications of radial basis functions.

[48] C.A. Glasbey and K.V. Mardia. A review of image warping methods. *Journal of Applied Statistics.*, 25:155–171, 1998.

[49] G. H. Golub and Charles F. Van L. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[50] M. Griebel. The combination technique for the sparse grid solution of PDEs on multiprocessor machines. *Parallel Processing Letters*, 2(1):61–70, 1992. also as SFB Bericht 342/14/91 A, Institut für Informatik, TU München, 1991.

[51] M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.

[52] M. Griebel and S. Knapek. Optimized tensor-product approximation spaces. *Constr. Approx.*, 16(4):525–540, 2000.

[53] M. Griebel and F. Koster. Multiscale methods for simulation of turbulent flow. In *Herschel, E, editers, Numerical Flow Simulation*, volume 82 of *Notes on Numerical Fluid Mechanics and Multidisciplinary design*, pages 203–214. Springer-Verlag, 2003.

[54] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In *Iterative methods in linear algebra (Brussels, 1991)*, pages 263–281. North-Holland, Amsterdam, 1992.

[55] S. J. Hales and J. Levesley. Error estimates for multilevel approximation using polyharmonic splines. *Numer. Algorithms*, 30(1):1–10, 2002.

[56] R. L. Hardy. Multiquadrics of topography and other irregular surface. *J. Geophys Res*, 76:1905–1915, 1971.

[57] R. L. Hardy. Geodetic application of multiquadric analysis, AVN Allg. *Varmess. Nachr*, 79:389–406, 1972.

[58] R. L. Hardy. Research results in application equations to serveying and mapping problem. *Survg. mapp*, 35:321–332, 1975.

[59] R. L. Hardy. Theory and applications of the multiquadric-biharmonic method. 20 years of discovery 1968–1988. *Comput. Math. Appl.*, 19(8-9):163–208, 1990.

[60] M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(2–3):249–275, 2007.

[61] P. W. Hemker. Sparse-grid finite-volume multigrid for 3D-problems. *Adv. Comput. Math.*, 4(1-2):83–110, 1995.

[62] N. J. Higham. *Accuracy and stability of numerical algorithms.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

[63] A. Iske. Hierarchical scattered data filtering for multilevel interpolation schemes. In *Mathematical methods for curves and surfaces (Oslo, 2000)*, Innov. Appl. Math., pages 211–221. Vanderbilt Univ. Press, Nashville, TN, 2001.

[64] A. Iske. *Multiresolution methods in scattered data modeling.* Spriger- Verlag, Berlin Heidelberg New York, 2004.

[65] A. Iske and J. Levesley. Multilevel scattered data approximation by adaptive domain decomposition. *Numer. Algorithms*, 39(1-3):187–198, 2005.

[66] E. J. Kansa. Multiquadrics — a scattered data approximation scheme with applications to computational fluid-dynamics. I. Surface approximations and partial derivative estimates. *Comput. Math. Appl.*, 19(8-9):127–145, 1990.

[67] E. J. Kansa. Multiquadrics — a scattered data approximation scheme with applications to computational fluid-dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Comput. Math. Appl.*, 19(8-9):147–161, 1990.

[68] E. J. Kansa and R. E. Carlson. Improved accuracy of multiquadric interpolation using variable shape parameters. *Comput. Math. Appl.*, 24(12):99–120, 1992. Advances in the theory and applications of radial basis functions.

[69] E. J. Kansa and Y. C. Hon. Circumventing the ill-conditioning problem with multiquadric radial basis functions: applications to elliptic partial differential equations. *Comput. Math. Appl.*, 39(7-8):123–137, 2000.

[70] A. Klimke and B. Wohlmuth. Computing expensive multivariate functions of fuzzy numbers using sparse grids. *Fuzzy Sets and Systems*, 154(3):432–453, 2005.

[71] W. A. Light, E. W. Cheney, and N. Dyn. Interpolation by piecewise-linear radial basis functions. I. *J. Approx. Theory*, 59(2):202–223, 1989.

[72] L. Ling and E. J. Kansa. Preconditioning for radial basis functions with domain decomposition methods. *Math. Comput. Modelling*, 40(13):1413–1427 (2005), 2004.

[73] L. Ling and E. J. Kansa. A least-squares preconditioner for radial basis functions collocation methods. *Adv. Comput. Math.*, 23(1-2):31–54, 2005.

[74] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2(1):11–22, 1986.

[75] C. T. Mouat. *Fast algorithms and preconditioning techniques for fitting radial basis functions.* PhD thesis, Department of Mathematics, University of Canterbury, 2001.

[76] F. J. Narcowich, R. Schaback, and J. D. Ward. Multilevel interpolation and approximation. *Appl. Comput. Harmon. Anal.*, 7(3):243–261, 1999.

[77] Y. Ohtake, A. Belyaev, and H.-P. Seidel. 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graph. Models*, 67(3):150–165, 2005.

[78] U. Pettersson, E. Larsson, G. Marcusson, and J. Persson. Improved radial basis function methods for multi-dimensional option pricing. *J. Comput. Appl. Math.*, 222(1):82–93, 2008.

[79] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Adv. Comput. Math.*, 11(2-3):193–210, 1999. Radial basis functions and their applications.

[80] R. Schaback. Creating surfaces from scattered data using radial basis functions. In *Mathematical methods for curves and surfaces (Ulvik, 1994)*, pages 477–496. Vanderbilt Univ. Press, Nashville, TN, 1995.

[81] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Adv. Comput. Math.*, 3(3):251–264, 1995.

[82] R. Schaback. On the efficiency of interpolation by radial basis functions. In *Surface fitting and multiresolution methods (Chamonix–Mont-Blanc, 1996)*, pages 309–318. Vanderbilt Univ. Press, Nashville, TN, 1997.

[83] A. Schreiber. *The method of Smolyak in multivariate interpolation.* PhD thesis, der Mathematisch-Naturwissenschaftlichen Fakultäten, der Georg-August-Universität zu Göttingen, 2000.

[84] R. Sibson and G. Stone. Computation of thin-plate splines. *SIAM J. Sci. Statist. Comput.*, 12(6):1304–1313, 1991.

[85] W. Sickel and F. Sprengel. Interpolation on sparse grids and tensor products of Nikol′skij-Besov spaces. *J. Comput. Anal. Appl.*, 1(3):263–288, 1999. Dedicated to Professor Paul L. Butzer on the occasion of his 70th birthday.

[86] S. A. Smolyak. Quadrature and interpolation of formulas for tensor product of certian classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963.

[87] G. W. Stewart. *Introduction to matrix computations.* Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, 1973. Computer Science and Applied Mathematics.

[88] J. Stewart. Positive definite functions and generalizations, an historical survey. *Rocky Mountain J. Math.*, 6(3):409–434, 1976.

[89] E. Süli and D. F. Mayers. *An introduction to numerical analysis.* Cambridge University Press, Cambridge, 2003.

[90] V. N. Temlyakov. Approximation of functions with bounded mixed derivative. In *Proc. Steklov Institute Math, 1989.* AMS, 1989.

[91] L. N. Trefethen and David Bau, III. *Numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[92] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(4):389–396, 1995.

[93] H. Wendland. Numerical solution of variational problems by radial basis functions. In *Approximation theory IX, Vol. 2 (Nashville, TN, 1998)*, Innov. Appl. Math., pages 361–368. Vanderbilt Univ. Press, Nashville, TN, 1998.

[94] H. Wendland. *Scattered data approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, Cambridge, 2005.

[95] Z. Wu. Multivariate compactly supported positive definite and compactly supported radial functions. *Adv. Comput. Math.*, 4(4):283–292, 1995.

[96] C. A. Zala and I. Barrodale. Warping aerial photographs to orthomaps using thin plate splines. *Adv. Comput. Math.*, 11(2-3):211–227, 1999. Radial basis functions and their applications.

[97] C. Zenger. Sparse grids. In *Parallel algorithms for partial differential equations (Kiel, 1990)*, volume 31 of *Notes Numer. Fluid Mech.*, pages 241–251. Vieweg, Braunschweig, 1991.