Reduction of Multivariate Mixtures and Its Applications

by

Xinshuo Yang

B.S., Jilin University, 2011

M.S., University of Colorado at Boulder, 2015

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2018

This thesis entitled:
Reduction of Multivariate Mixtures and Its Applications
written by Xinshuo Yang
has been approved for the Department of Applied Mathematics

Prof. Gregory Beylkin

Dr. Zydrunas Gimbutas

Date ____________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Yang, Xinshuo (Ph.D., Applied Mathematics)

Reduction of Multivariate Mixtures and Its Applications

Thesis directed by Prof. Gregory Beylkin

Abstract. We consider a fast deterministic algorithm to identify the "best" linearly independent terms in multivariate mixtures and use them to compute an equivalent representation with fewer terms, up to user-selected accuracy. Our algorithm employs the well-known pivoted Cholesky decomposition of the Gram matrix constructed using terms of the mixture. Importantly, the multivariate mixtures do not have to be a separated representation of a function and complexity of the algorithm is independent of the number of variables (dimensions). The algorithm requires $\mathcal{O}\left(r^2N\right)$ operations, where N is the initial number of terms in a multivariate mixture and r is the number of selected terms. Due to the condition number of the Gram matrix, the resulting accuracy is limited to about 1/2 digits of the used floating point arithmetic. We also consider two additional reduction algorithms for the same purpose. The first algorithm is based on orthogonalization of the multivariate mixture and have a similar performance as the approach based on Cholesky factorization. The second algorithm yields a better accuracy, but currently in high dimensions is only applicable to multivariate mixtures in a separated representation.

We use the reduction algorithm to develop a new adaptive numerical method for solving differential and integral equations in quantum chemistry. We demonstrate the performance of this approach by solving the Hartree-Fock equations in two cases of small molecules. We also describe a number of initial applications of the reduction algorithm to solve partial differential and integral equations and to address several problems in data sciences. For data science applications in high dimensions we consider kernel density estimation (KDE) approach for constructing a probability density function (PDF) of a cloud of points, a far-field kernel summation method and the construction of equivalent sources for non-oscillatory kernels (used in both, computational physics and data science) and, finally, show how to use the reduction algorithm to produce seeds for subdividing a cloud of points into groups.

Dedication

To the memory of my grandmother.

Acknowledgements

First, I would like to sincerely express my gratitude to my thesis adviser, Gregory Beylkin, for his continuous support, endless discussion and inspirational mentorship throughout my PhD studies. Along with his technical guidance, he was very generous in sharing his life experiences. These experiences will be my primary guide during my life and academic career. I would also like to thank our research group member, Lucas Monzón, for his insightful discussion and great collaboration.

I wish to thank the members of my dissertation committee: Bengt Fornberg, Zydrunas Gimbutas, Ian Grooms and Gunnar Martinsson for generously offering their time, support and guidance throughout the preparation and review of this manuscript. A special thank to Zydrunas Gimbutas for his suggestions on the Gram-Schmidt orthogonalization algorithm.

I will forever be thankful to my parents for the support and encouragement they provided me through my entire life.

Last but no least, thank you to my wife, Meng, for being there for me during all the ups and downs.

Contents

Tables

Figures

Chapter 1

Introduction

In this thesis, we develop what we call reduction algorithms for computing with multivariate mixtures in high dimensions. Using reduction algorithms, we develop a new adaptive method for solving equations of quantum chemistry and demonstrate its performance by solving the Hartree-Fock equations for two diatomic molecules. The reduction algorithm allows us to work with non-separated multivariate mixtures that are a far reaching generalization of multivariate separated representations [10, 11, 8] and can be used as a tool for solving multi-dimensional problems. These algorithms allow us to obtain solutions of PDEs in high dimensions as well as to address several problems in data science. The main contributions of this thesis are:

- We develop a fast algorithm for reducing the number of terms in a multivariate mixtures for a given accuracy. This method is based on a pivoted Cholesky factorization of the Gram matrix. We also consider two additional reduction algorithms for the same purpose. The first algorithm is based on orthogonalization of the multivariate mixture and have a similar performance as the approach based on Cholesky factorization. The second algorithm yields a better accuracy, but currently in high dimensions is only applicable to multivariate mixtures in a separated representation.

- We use reduction algorithm to develop a new adaptive numerical method for solving problems in quantum chemistry and demonstrate its performance by solving the Hartree-Fock equations for two diatomic molecules.

- We propose a new approach for solving multi-dimensional partial differential and integral equations in a functional form using non-separated multivariate mixtures. We apply it to two problems in high dimensions, the Poisson's equation and a second order elliptic equation with a variable coefficient.

- We present several examples of applying our reduction algorithm to kernel-type methods in data science. These examples include fast far-field kernel summation, kernel density estimation (KDE) for constructing a probability density function (PDF) and subdivision of a cloud of points into groups.

1.1 Background and novelty

We develop a fast algorithm to reduce the number of terms of a multivariate mixture for the purpose of solving differential and integral equations in multi-dimensions. A key example of multivariate mixtures that we consider in this thesis is a linear combination of multivariate Gaussians, which are also known as Radial Basis Functions (RBFs). While the multivariate mixtures and multivariate atoms that we consider are somewhat more general than RBFs (since the only requirement we impose is that their inner product can be computed efficiently), the novelty of this thesis is a numerical method for solving PDEs using such mixtures.

RBFs have been initially developed for solving scattered data interpolation problem for applications in topography [38]. Notable contributions were later made by M. J. D. Powell (see e.g. [58] and references therein). RBFs found applications in the numerical solution of PDEs (see e.g. [33, 34, 32] and references therein) as well as computer graphics, statistical learning theory, neural networks and signal and image processing (see e.g. references in [31]). While we do not address the problem of scattered data interpolation, we can compare our approach to previously developed methods that use RBFs for solving PDEs.

In [32] the authors use near-flat RBFs and formulate the problem for solving partial differential equations by enforcing both, the equation and the boundary conditions, at all nodes. The nodes for

RBFs are selected either as a structured grid or scattered nodes (see also [42, 24]). An alternative strategy for selecting nodes for RBFs is to use the so-called greedy algorithms [60, 42, 26], i.e. algorithmic strategy to optimize local performance (in terms of accuracy) with the expectation that such strategy is globally efficient. At issue is how to select an appropriate collection of functions so that the solution of a PDE (or that of an integral equation) can be well represented in such basis. In particular, one seeks an algorithm to remove functions that can be expressed as linear combinations of a subset of selected functions and to insert additional functions to improve accuracy of the solution. The key approach of greedy algorithms uses the strategy suggested for scattered data interpolation (see e.g. cross-validation in [31]). Namely, by excluding some nodes or inserting some additional RBFs, a comparison can be made using the rest of the functions to examine if a particular function can be removed or an additional function needs to be inserted. In its basic form, such approach is very expensive and various modifications were developed (see e.g. [51] and references therein).

Our approach to obtaining a basis for the solution differs substantially from the existing methods both conceptually and technically. We start by formulating an iteration for solving a PDE which is generally obtained by recasting a PDE as an integral equation using Green's functions. It is desirable that such iteration is convergent (as in the case of the integral form of the Hartree-Fock equations or the Kohn-Sham equations) but we also demonstrate that it is possible to work with iterations that are convergent only for some choices of parameters. *The key to our approach is that we use such iterations to generate the necessary basis functions.* It has been demonstrated (see [48, 49, 40]) that in the important example of the Hartree-Fock equations (as well as the Kohn-Sham equations of the Density Functional Theory), the corresponding PDEs can be converted into integral equations and solved by a convergent iteration. It has been shown in [40, 15, 16] that all integral operators in such formulation can be approximated within any user-selected finite accuracy via a linear combination of Gaussians. Using reduction algorithm described in Section 2.1, we introduce a new strategy for solving equations using multivariate atoms and, in particular, Gaussian atoms. We seek solution as a linear combination of Gaussian atoms and observe that applying all operations required by the iteration (when solving the Hartree-Fock equations or the Kohn-Sham equations) produces new Gaussian atoms, i.e. a linear combination of multivariate atoms of the same form. A obvious problem is that the number of such Gaussian atoms grows rapidly unless there is a way to remove the unnecessary (linear dependent) terms. For this purpose we use an algorithm that selects the "best" linearly independent terms after each operation within the iteration and compute new coefficients for the selected terms. Thus, we do not select any grid for Gaussian atoms beforehand and do not use greedy algorithms to achieve an acceptable local performance. Instead, we use the equation itself to provide a linear subspace in which we approximate the solution.

Our key example is finding solution of the Hartree-Fock equations (which are nonlinear) in two cases of small molecules. Note that solutions of the Hartree-Fock equations (as well as the Kohn-Sham equations and the Schrödinger equation) have cusps (that are not rotationally symmetric) at locations of nuclear centers and resolving these cusps is critically important to obtain correct energies.

While a transition to large molecules will require further work on accelerating the reduction algorithm of Section 2.1, we already use the basic approach with such acceleration described in Section 2.4 for solving the Hartree-Fock equations.

In Chapter 4 we show application of our approach to two additional PDEs in high dimensions, the Poisson's equation and a second order elliptic equation with a variable coefficient, both in the free space. Our purpose is to consider equations whose solutions do not have an efficient separated representation and demonstrate that our approach is capable of solving these problems.

We then turn to several additional applications of the reduction algorithm, including kernel density estimation (KDE) for constructing a PDF of a cloud of points, a far-field kernel summation method and the construction of equivalent sources for non- oscillatory kernels.

1.2 Motivation

1.2.1 Curse of dimensionality and separated representations

Many high-dimensional problems are difficult to solve by usual numerical methods since the cost increases exponentially with the dimension. When a typical algorithm in dimension one is extended to dimension d, its computational cost is taken to the power d. This effect has been dubbed "the curse of dimensionality"[5]. As an illustration, let us consider a spectral projection of a function of interest onto the span of N orthonormal basis functions,

$$g(x) = \sum_{j=1}^{N} \langle g, \psi_j \rangle \psi_j(x)$$

where $\langle \cdot, \cdot \rangle$ denotes an appropriately defined inner product. An straightforward extension of the one-dimensional spectral projection to d dimension is decomposing a multivariate function $g(x_1, x_2, \ldots, x_d)$ via a tensor product of orthonormal bases $\{\psi_{jk}(x_k)\}_{\substack{j=1,\ldots N\\ k=1,\ldots,d}}$:

$$g(x_1, x_2, \ldots, x_d) \quad = \quad \sum_{j_1=1}^{N} \sum_{j_2=1}^{N} \cdots \sum_{j_d=1}^{N} c_{j_1, j_2, \cdots, j_d} \left(\prod_{k=1}^{d} \psi_{j_1,k}(x_k) \right)$$

where

$$c_{j_1, j_2, \cdots, j_d} = \left\langle g, \prod_{k=1}^{d} \psi_{j_1,k}(x_k) \right\rangle .$$

The cost of this operation is $\mathcal{O}\left(N^d\right)$ and is prohibitively expensive for large d.

A number of problems in high dimensional spaces have been addressed by using separated representations [10, 15]. Recall that separated representation is a natural extension of the usual separation of variables as seeking an approximation

$$f(x_1, \ldots, x_d) = \sum_{l=1}^{r} s_l \phi_1^{(l)}(x_1) \cdots \phi_d^{(l)}(x_d) + \mathcal{O}(\epsilon), \tag{1.1}$$

where the functions $\phi_j^{(l)}(x_j)$ are normalized by the standard L^2-norm, $\|\phi_j^{(l)}\|_2 = 1$ and $s_l > 0$ are referred to as s-values. The number of terms, r, is called the separation rank of f and is assumed to be small. In this approximation the functions $\phi_j^{(l)}(x_j)$ are not fixed in advance but are optimized as to achieve the accuracy goal with (ideally) a minimal separation rank r. It is well understood that when the separation rank r is independent of d, the computation costs and storage requirements of standard algebraic operations in separated representations scale linearly in d [11]. For this reason, such representations are widely used for approximating high-dimensional functions [10, 11, 27, 28, 44, 59]. Importantly, a separated representation is not a projection onto a subspace, but rather a nonlinear method to track a function in a high-dimensional space while using a small number of parameters.

When using separated representations, common operations such as summations and multiplications, lead to new separated representations with separation ranks that may be larger than necessary. Therefore, a standard practice is to reduce the separation rank of a given separated representation without sacrificing much accuracy, for which the workhorse algorithm is Alternating Least Squares (ALS). This algorithm is one of the key tools in numerical multilinear algebra and was introduced originally for data fitting as PARAFAC model (PARAllel FACtor analysis) [41] and CANDECOMP (abbreviated in this thesis as CTD, canonical tensor decomposition) [23]. It has been used extensively in data analysis of (mostly) three-way arrays (see e.g. the reviews [65, 22], [50] and references therein). We note that any discretization of f in (1.1) leads to a d-dimensional tensor $\mathcal{U} \in \mathbb{R}^{M_1 \times \cdots \times M_d}$ yielding a canonical tensor decomposition (CTD) of separation rank r,

$$\mathcal{U}_{i_1,\ldots,i_d} = \sum_{l=1}^{r} \sigma_l \prod_{j=1}^{d} u_{i_j}^{(l)}, \tag{1.2}$$

where the s-values σ_l are chosen so that each vector $\mathbf{u}_j^{(l)} = \left\{ u_{i_j}^{(l)} \right\}_{i_j=1}^{M_j}$ has unit Frobenius norm $\|\mathbf{u}_j^{(l)}\|_F = 1$ for all j, l. However, ALS algorithm relies heavily on separated format in (1.1-1.2) and is not available for general multivariate mixtures.

1.2.2 Multivariate atoms

In this thesis, we consider multivariate functions that can be approximated via a linear combination of what we call multivariate atoms,

$$u\left(\mathbf{x}\right) = \sum_{l=1}^{r} c_l g_l\left(\mathbf{x}\right), \quad \mathbf{x} \in \mathbb{R}^d, \quad \left\|g_l\right\|_2 = 1, \tag{1.3}$$

so that the inner product between the atoms,

$$\langle g_l, g_{l'} \rangle = \int_{\mathbb{R}^d} g_l(\mathbf{x})\, g_{l'}(\mathbf{x})\, d\mathbf{x}, \tag{1.4}$$

can be computed efficiently. Importantly, the representation (1.3) are far more general than (1.1) since the multivariate atoms $g_l(\mathbf{x})$ do not necessarily admit a separated representation. When used for approximation of a multivariate function, (1.3) can yield a more efficient representation. As an illustration, let us consider the bivariate Gaussian mixture with two terms,

$$\begin{aligned} h(x_1, x_2) &= \exp\left(-\frac{x_1^2}{2} - \frac{x_2^2}{2}\right) + \exp\left(-x_1^2 - x_2^2 - x_1 x_2\right) \\ &= \exp\left(-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}_1^{-1}\mathbf{x}\right) + \exp\left(-\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}_2^{-1}\mathbf{x}\right), \end{aligned}$$

where $\mathbf{x} = (x_1, x_2)^T$, and

$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{pmatrix}.$$

A separated representation using one dimensional Gaussians, i.e.,

$$h(x_1, x_2) \approx \sum_{l=1}^{r} s_l \phi_1^{(l)}(x_1)\, \phi_2^{(l)}(x_2) + \mathcal{O}(\epsilon)$$

where

$$\phi_1^{(l)}(x_1) = e^{-\alpha_l x_1^2}, \quad \phi_2^{(l)}(x_2) = e^{-\beta_l x_2^2},$$

would require the separation rank $r \gg 2$ so that it can achieve a reasonable accuracy ϵ.

1.2.3 Gaussian atoms

A particularly important example are multivariate Gaussian atoms yielding a multivariate Gaussian mixture. In this case the atoms are

$$\begin{aligned} g(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) &= \left(\det(4\pi\boldsymbol{\Sigma}_l)\right)^{\frac{1}{4}} N(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) \\ &= \frac{1}{\left(\det(\pi\boldsymbol{\Sigma}_l)\right)^{1/4}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_l)^T \boldsymbol{\Sigma}_l^{-1} (\mathbf{x} - \boldsymbol{\mu}_l)\right), \end{aligned} \tag{1.5}$$

where $N\left(\mathbf{x},\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l\right)$ is the multivariate Gaussian distribution with the mean $\boldsymbol{\mu}_l$, the covariance $\boldsymbol{\Sigma}_l\in\mathbb{R}^{d\times d}$ is a symmetric positive definite matrix and $\|g_l\|_2=1$. The L^2-inner product between two multivariate Gaussian atoms is

$$\begin{aligned}&\int_{\mathbb{R}^d} g_l\left(\mathbf{x},\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l\right) g_{l'}\left(\mathbf{x},\boldsymbol{\mu}_{l'},\boldsymbol{\Sigma}_{l'}\right)d\mathbf{x}\\ =\;& 4^{\frac{d}{4}}\frac{\left(\det\left(\pi\boldsymbol{\Sigma}_l\right)\det\left(\pi\boldsymbol{\Sigma}_{l'}\right)\right)^{\frac{1}{4}}}{\det\left(\pi\left(\boldsymbol{\Sigma}_l+\boldsymbol{\Sigma}_{l'}\right)\right)^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_l-\boldsymbol{\mu}_{l'}\right)^T\left(\boldsymbol{\Sigma}_l+\boldsymbol{\Sigma}_{l'}\right)^{-1}\left(\boldsymbol{\mu}_l-\boldsymbol{\mu}_{l'}\right)\right).\end{aligned}$$

1.2.4 The use of Gaussians in numerical analysis

1.2.4.1 Gaussians in computational physics and quantum chemistry

The early use of Gaussian orbitals in electronic structure theory (instead of the more physical, Slater-type orbitals) was first proposed by Boys [21] in 1950. The principal reason for the use of Gaussian basis functions in molecular quantum chemical calculations is the explicit evaluation of integrals. The foundational papers of computational quantum chemistry [20, 53, 64] use a linear combination of Gaussians for exactly this reason showing that it is possible to reproduce both the correct cusp-type behavior and long-range decay of the solutions of relevant PDEs. However, in the process of solving equations, optimizing both coefficients and exponents of linear combinations of Gaussians via Newton's method (as was done in these papers) proved unsustainable as the number of variables in quantum chemistry problems grow rapidly with complexity of the molecules. This lead to replacing nonlinear approximations by Gaussian orbitals constructed off-line and used as a fixed basis to represent a solution. A drawback of this approach is the appearance of the so-called "basis error", an error due to a possible mismatch between the true solution and the linear subspace chosen to approximate it. As it was demonstrated in [9], it is possible to iteratively solve equations of quantum chemistry using new nonlinear algorithms which reduce the number of terms in intermediate representations without resorting to Newton's method. The results in [9] are based on using Slater-type orbitals, but it is clear that similar results can be obtained using representations via Gaussians which is demonstrated in Section .3.1.

1.2.4.2 Multiresolution analysis and MADNESS

Multiresolution analysis (MRA) based on multiwavelets has been a powerful numerical method for solving PDEs to high accuracy [3, 40, 68, 47, 7]. Recall that a multiresolution analysis decomposes the Hilbert space $L^2\left(\mathbb{R}^d\right)$, $d > 1$ into a chain of closed subspaces

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_n \subset \cdots .$$

The wavelet subspaces W_j are defined as an orthogonal complement of V_j in V_{j-1}, thus

$$V_n = V_0 \oplus_{j=0}^{n} W_j .$$

The MRA with multiwavelet bases [2] is capable of organizing functions and operators efficiently in terms of their proximity on a given scale and between different scales, and it provides a simple mechanism for truncation and adaptive refinement that can be used to maintain the desired accuracy. The adaptivity associated with the decomposition and refinement is key to representing both, global and local behavior via the the multiresolution hierarchy. On the other hand, the straightforward transition from multiresolution algorithms in one spatial dimension to those in dimensions two, three and higher yield algorithms that are too costly for practical applications. Besides using a multiresolution approach, a critical step has been the development of separable representations for the operators involved in the PDEs. Many key operators in mathematical physics depend only on the distance between interacting entities and, therefore, have radial kernels. As was demonstrated in [10, 11, 15, 16, 7], these operators (e.g. Poisson kernel and non-oscillatory Helmholtz kernel) can be efficiently represented via Gaussians leading to their separated representations and, as a consequence, to practical algorithms (see [40, 68, 69, 39]).

A successful application that incorporates the techniques of multiresolution analysis and separated representations is MADNESS (Multiresolution Adaptive Numerical Environment for Scientific Simulation), a high-level software environment for the solution of integral and differential equations in many dimensions with guaranteed precision. While initially MADNESS was designed to solve problems in computational chemistry [40, 68, 69], to date, it has been applied to solve problems in atomic and molecular physics, electrostatics, fluid dynamics, graph theory, materials science, nanoscience, nuclear physics, and solvation models [30, 35, 46, 66].

1.2.4.3 Radial basis functions

We already mentioned many applications of RBFs for scattered data interpolation, solving PDEs, computer graphics, statistical learning theory, neural networks and signal and image processing. Gaussian atoms is one of the possible choices of such functions and we refer for further examples to e.g. [34, 31] and references therein.

1.2.4.4 Gaussian Multiresolution Analysis

Multivariate Gaussian mixtures can achieve any finite accuracy when approximating functions as demonstrated by the introduction of an approximate Gaussian multiresolution analysis (GMRA) in [17]. The approximate GMRA is constructed by using the scaling functions

$$\left\{\phi_{j,k}\left(x\right)=2^{j/2}\phi\left(2^{j}x-k\right)=2^{j/2}\sqrt{\frac{\alpha}{\pi}}e^{-\alpha\left(2^{j}x-k\right)^{2}}\right\}_{j,k\in\mathbb{Z}}, \tag{1.6}$$

which are normalized so that the L^2-norm of functions $\phi_{j,k}$,

$$\left\|\phi_{j,k}\left(x\right)\right\|_{2}=\left(\frac{\alpha}{2\pi}\right)^{\frac{1}{4}},$$

does not depend on the scale j. The scaling function ϕ satisfies an approximate two-scale relation,

$$\left|\phi\left(x\right)-\sqrt{\frac{4\alpha}{3\pi}}\sum_{k\in\mathbb{Z}}e^{-\frac{\alpha}{3}k^{2}}\phi\left(2x-k\right)\right|\leq\epsilon\phi\left(x\right), \tag{1.7}$$

where the parameter α is chosen to achieve the desired accuracy,

$$\epsilon=\vartheta_{3}\left(0,e^{-\frac{3\pi^{2}}{4\alpha}}\right)-1, \tag{1.8}$$

and

$$\vartheta_{3}\left(z,q\right)=\sum_{n\in\mathbb{Z}}q^{n^{2}}e^{2inz}$$

is the Jacobi theta function. The approximate nature of the GMRA does not limit its applicability since any finite accuracy can be selected. For example, choosing $\alpha=1/5$ yields $\epsilon\approx2.22\cdot10^{-16}$

thus achieving double precision accuracy. Alternatively, as shown in [17], an exact multiresolution analysis can be constructed so that its scaling function is approximated by a Gaussian for any user-selected accuracy on any finite number of scales.

By its nature, in principle, any MRA allows construction of adaptive algorithms. The choice of an orthonormal basis makes adaptive algorithms numerically stable and, the particular choice of multiwavelets, makes it easier and more efficient to implement the hierarchical structure of the corresponding MRA. On the other hand, using Gaussians as basis functions presents an immediate difficulty since these functions are far from being orthogonal. Thus, in order to construct adaptive algorithms using these functions, an approach different from using the standard MRA decomposition and reconstruction tools has to be developed. Moreover, in order to work with functions of many variables, one has to avoid the usual spectral/pseudo-spectral approach of identifying beforehand the linear subspace for the solution; it has to be constructed in the process of solving equations. The reduction algorithms for multivariate mixtures developed in this thesis provide a numerical tool towards this goal.

1.2.4.5 Approximate approximations

Another numerical approach involving Gaussians was suggested in [55] (see also references therein). It involves constructing a quasi-interpolant, a linear combination of shifted Gaussians with coefficients taken to be the function values on an equally spaced grid,

$$M_{h,D}s\left(x\right) = \frac{1}{\sqrt{\pi D}} \sum_{m=-\infty}^{\infty} s\left(mh\right) e^{-\frac{(x-mh)^2}{Dh^2}}$$

where h is the step size, and D is a parameter. The approximation error then consists of two contributions: a contribution converging to zero as $h \to 0$ and a non-convergent part, the so-called saturation error. When used, one can choose appropriate h and D such that the approximation provides a prescribed error. While using function values is a desirable feature, it forces a significant oversampling if high accuracy is desired. In our view using a multiresolution basis where the scaling function is well approximated by Gaussians is a better alternative [17].

1.2.5 Summary

When solving PDEs, it is desirable to use adaptive algorithms to achieve a user-selected accuracy while minimize the computational cost. We introduce a new type of adaptive numerical algorithms that use a Basis Generating Iteration (BGI) for this purpose. The basic idea of such algorithms is simple: in the process of solving equations, we represent both *operators and functions* via Gaussians (or other functional atoms) and then compute the required integrals explicitly. The difficulty of this approach is then a rapid proliferation of terms in the resulting mixtures. For example, if an integral involves three Gaussian mixtures with 100 terms each, the resulting Gaussian mixture has 10^6 terms. However, in many practical applications most of these terms can be nearly linearly dependent and, thus, in order to use the resulting Gaussian mixture (as the final result or an intermediate result to be carried over for further computation), we need a fast algorithm to find the "best" linearly independent subset of the terms. We subdivide the process of solving PDEs as:

(1) Recast the PDEs as integral equations and construct an iteration to solve them.

(2) Represent both operators and functions via multivariate mixtures.

(3) Perform an iteration step by computing the required integrals explicitly to obtain a new multivariate mixture.

(4) Reduce the number of terms in the resulting mixture and repeat the above steps.

It is preferable that the iteration is convergent (as in the case of Hartree-Fock equations in Chapter 3.1) but it is not absolutely necessary (see example in Chapter 4). In this approach, we do not fix a representation of a solution in advance (i.e., choose the parameters of Gaussians in advance) since the iteration is generating the basis. In particular, we develop algorithms that adaptively select appropriate translations (shifts) and exponents of the Gaussian mixtures throughout the process of iteratively solving PDEs. The excessive number of functions generated by the integral equation is then "pruned" by our reduction algorithm.

1.3 Outline of the thesis

In Chapter 2, we introduce fast deterministic algorithms to reduce the number of terms of a linear combination of multivariate functions. We first describe a fast algorithm based on a pivoted Cholesky decomposition of the Gram matrix of the initial multivariate mixture (it is the main algorithm we use). While the accuracy of this approach is limited to about 1/2 of the available significant digits, it is advantageous in high dimensions since its complexity is dimension independent. We then introduce two additional reduction algorithms for the same purpose. We also describe a technique to accelerate the reduction algorithms.

In Chapter 3, we present a new adaptive algorithm for electronic structure calculations based on the reduction algorithm developed in Chapter 2. We demonstrate performance of the new algorithm by solving the Hartree-Fock equation for two diatomic molecules.

In Chapter 4, we present several examples of using our main algorithm for solving differential and integral equations. We represent solutions of the equations in a functional form and adaptively solve these equations. We start with the free space Poisson's equation with non-separable right hand side and then present an example of solving an elliptic problem with variable coefficients; we consider both examples in dimensions $d = 3$ through $d = 7$.

In Chapter 5, we first use our algorithm in dimension $d = 1$ to construct an efficient representation of the PDF of a cloud of points via kernel density estimation and compare it with results obtained via the standard approach. We then present an example of constructing PDFs in high dimensions.

In Chapter 6, we turn to kernel summation methods in high dimensions, consider far-field evaluation in such computations and explore the problem of constructing equivalent sources in a similar setup. We also illustrate how a reduction algorithm can be used to partition points into groups (in a hierarchical fashion if desired).

In Chapter 7, we conclude the thesis, present an outlook for expected applications and extensions of the tools and methods described here.

In Appendix A, we present some key identities for computing with multivariate Gaussians.

Chapter 2

Reduction Algorithms

In this chapter we describe fast deterministic algorithms to reduce the number of terms of a linear combination of multivariate functions of d variables by selecting the “best” subset of these functions that can, within a target accuracy, approximate the rest of them. We describe three types of algorithms for this purpose. The first algorithm is based on a pivoted Cholesky decomposition of the Gram matrix of the initial multivariate mixture; we assume that the entries of this matrix, i.e. the inner product of these functions, can be computed efficiently. This algorithm was mentioned in the discussion of tensor interpolative decomposition (tensor ID) of the canonical tensor representation in [18]. Due to the use of a Gram matrix, the accuracy of this approach is limited to about one half of the available significant digits (e.g. $10^{-7} \sim 10^{-8}$ when using double precision arithmetic). Nevertheless, this algorithm appears advantageous in high dimensions since its complexity is dimension independent and, if desired, full accuracy can be restored by performing some evaluations in higher precision. We obtain estimates of its accuracy and show that its computational complexity is $\mathcal{O}\left(r^2N\right)$, where N is the original number of terms and r is the number of skeleton terms obtained for a given accuracy.

We also consider an algorithm based on Gram-Schmidt orthogonalization of the multivariate functions. The computational cost of this algorithm $\mathcal{O}\left(r^3+r^2N\right)$ is also dimension independent. We observe that since the ill-conditioned Gram matrix appears in this algorithm, the resulting approximation loses 1/2 of available significant digits similar to the our main algorithm.

Finally, we describe an approach using frequency sampling that achieves full precision, but so far limited to low dimensions or mixtures in separated form. For this algorithm, we need access to the Fourier transform of the functions in the mixture. Since the Fourier transform is readily available for Gaussian atoms, we present this algorithm for the case of Gaussian mixtures and note that it can be used for any functional form that allows a rapid computation of the integrals involved.

2.1 Cholesky reduction

2.1.1 Introduction

We start with a linear combination of atoms of the form

$$u(\mathbf{x}) = \sum_{l=1}^{N} c_l g_l(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d \tag{2.1}$$

and, within a user-selected accuracy ϵ, seek a representation of the same form but with fewer terms. To be precise, we look for a partition of indices $I = \left[\widehat{I}, \widetilde{I}\right]$, where $\widehat{I} = [i_1, i_2, \cdots i_r]$ and $\widetilde{I} = [i_{r+1}, i_{r+2}, \cdots, i_N]$, and new coefficients $\tilde{c}_{i_m}, m = 1, \cdots r$, such that the function

$$\widetilde{u}(\mathbf{x}) = \sum_{j=1}^{r} \tilde{c}_{i_j} g_{i_j}(\mathbf{x}), \quad r \ll N, \tag{2.2}$$

approximates u,

$$u(\mathbf{x}) \approx \widetilde{u}(\mathbf{x}). \tag{2.3}$$

We present an algorithm based on a partial, pivoted Cholesky decomposition of the Gram matrix constructed using the atoms g_l in (2.1) and provide an estimate for the error in (2.3).

By analogy with the matrix Interpolative Decomposition (matrix-ID) (see e.g. [37]), we call the subset $\{g_{i_m}\}_{m=1}^{r}$ the skeleton terms and $\{g_{i_m}\}_{m=r+1}^{N}$ the residual terms. In order to identify the "best" subset of linear independent terms, we compute a pivoted Cholesky decomposition of the Gram matrix of the atoms of the multivariate mixture in (2.1). If the number of terms, N, is large then the cost of the full Cholesky decomposition is prohibitive. However, we show that we can terminate the Cholesky decomposition once the pivots are below a selected threshold. As a result, the complexity of the algorithm is $\mathcal{O}\left(r^2 N\right)$, where N is the initial number of terms and r is the number of selected (skeleton) terms. In fact, the final result will be the same as if we were to perform the full decomposition and then keep only the significant terms. This property is a consequence of the following lemma that can be found in e.g. [43, p.434, problem 7.1.P1].

2.1.2 A property of a positive semi-definite matrix

Lemma 1. Let $\mathbf{B} \in \mathbb{C}^{n\times n}$ be positive semi-definite, i.e. $\mathbf{B} = \mathbf{B}^*$ and $\mathbf{x}^*\mathbf{B}\mathbf{x} \geq 0$ for any $\mathbf{x} \in \mathbb{C}^n$. Then its diagonal entries b_{ii} are non-negative and the entries b_{ij} of $\mathbf{B}$ satisfy

$$|b_{ij}| \leq \sqrt{b_{ii}b_{jj}}. \tag{2.4}$$

In particular, assuming that the first i diagonal entries are in descending order and are greater or equal than the rest of the diagonal entries,

$$b_{11} \geq b_{22} \geq \cdots \geq b_{ii} \geq b_{i+1,i+1}, b_{i+2,i+2}, \dots b_{nn},$$

we have

$$|b_{ij}| \leq b_{ii}, \text{ for all } \; j \geq i. \tag{2.5}$$

Proof. Let $\{\mathbf{e_i}\}_{1\leq i\leq n}$ be the standard basis vectors, that is, $(\mathbf{e_i})_j = \delta_{ij}$. The diagonal entries are non-negative since, for any index i, $a_{ii} = \mathbf{e_i}^*\mathbf{B}\mathbf{e_i} \geq 0$. We now use the same approach to estimate the size of an off-diagonal entry $b_{ij} = |b_{ij}|\, e^{i\theta_{ij}}$. For the vector $\mathbf{x} = x_i\mathbf{e_i} + x_j\mathbf{e_j}$ we have

$$0 \leq \mathbf{x}^*\mathbf{B}\mathbf{x} = b_{ii}x_i\overline{x_i} + b_{ij}x_i\overline{x_j} + \overline{b_{ij}}\,\overline{x_i}x_j + b_{jj}x_j\overline{x_j}. \tag{2.6}$$

Setting

$$x_i = \left(\frac{b_{jj}}{b_{ii}}\right)^{1/4} \quad \text{and} \quad x_j = -e^{i\theta_{ij}}\left(\frac{b_{ii}}{b_{jj}}\right)^{1/4}$$

in (2.6), we obtain

$$0 \leq b_{ii}\left(\frac{b_{jj}}{b_{ii}}\right)^{1/2} - |b_{ij}| - |b_{ij}| + b_{jj}\left(\frac{b_{ii}}{b_{jj}}\right)^{1/2} = 2\sqrt{b_{ii}b_{jj}} - 2\,|b_{ij}|\,.$$

Thus, we arrive at

$$|b_{ij}| \leq \sqrt{b_{ii}b_{jj}} \leq \frac{b_{ii}+b_{jj}}{2}. \tag{2.7}$$

For the second part of the lemma, selecting $i \leq j$ implies that $b_{ii} \geq b_{jj}$ and, thus, (2.5) follows from the last inequality in (2.7). □

Lemma 1 implies

Corollary 1. Let G be a self-adjoint positive semi-definite matrix such that its Cholesky decomposition has monotonically decaying diagonal pivots. If we write its Cholesky decomposition as

$$G = \begin{pmatrix} L_r & 0 \\ W & Q \end{pmatrix} \begin{pmatrix} L_r^* & W^* \\ 0 & Q^* \end{pmatrix},$$

where L_r is an $r \times r$ lower triangular matrix with the smallest diagonal entry $\epsilon > 0$, then a partial Cholesky decomposition is of the form

$$G = \begin{pmatrix} L_r & 0 \\ W & 0 \end{pmatrix} \begin{pmatrix} L_r^* & W^* \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & QQ^* \end{pmatrix},$$

where all entries of the matrix QQ^* are less than ϵ.

Proof. After applying r steps of Cholesky decomposition, the matrix W does not change in the consecutive steps. The remaining matrix QQ^* is self-adjoint positive semi-definite and, due to the decay of the pivots, all of its diagonal entries are less than ϵ. Using Lemma 1, we conclude that all entries of QQ^* are less than ϵ. □

2.1.3 Error estimation

Let us organize the collection of atoms in (2.1) as

$$A = \left[g_1(\mathbf{x}), g_2(\mathbf{x}), \dots g_N(\mathbf{x})\right].$$

We can view A as a matrix with a gigantic number of rows resulting from a discretization of the argument $\mathbf{x} \in \mathbb{R}^d$. If we replace operations that require row-wise summation by the inner product between the atoms, we can use matrix notation in the sequel. Without loss of generality, to simplify notation, we assume that the first r atoms in A form the skeleton, that is, $A = (A_s \mid A_{ns})$, where A_s denotes the r skeleton and A_{ns} the $N - r$ non-skeleton atoms.

Given the vector of coefficients $c = [c_1, c_2, \dots c_N]^T$in (2.1), we want to find new coefficients $\tilde{c} = [\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_r]^T$ to approximate $u(\mathbf{x})$ by

$$\widetilde{u}(\mathbf{x}) = \sum_{l=1}^{r} \tilde{c}_l g_l(\mathbf{x}), \quad r \ll N, \tag{2.8}$$

and estimate the error $\|u-\widetilde{u}\|_2$ of the approximation. Note that we identify $u = Ac$ and $\widetilde{u} = A_s\tilde{c}$.

We first seek an approximation of all atoms via the skeleton atoms,

$$g_k(\mathbf{x}) \approx \sum_{i=1}^{r} p_{ik} g_i(\mathbf{x}), \quad k = 1, \ldots, N.$$

Selecting the coefficients p_{ik} as the solutions of the least squares problem, p_{ik} satisfy the normal equations,

$$\sum_{i=1}^{r} p_{ik} \langle g_i, g_{i'} \rangle = \langle g_k, g_{i'} \rangle, \quad k = 1, \ldots, N, \ i' = 1, 2, \ldots, r. \tag{2.9}$$

Introducing the matrix $P = \{p_{ik}\}_{\substack{i=1,\ldots r \\ k=1,\ldots N}}$, we write (2.9) as

$$A_s^* A_s P = A_s^* A \tag{2.10}$$

and observe that $P = (I_r \mid S)$, where I_r is the $r \times r$ identity matrix and S, an $(N-r) \times r$ matrix, which satisfies

$$A_s^* A_s S = A_s^* A_{ns}. \tag{2.11}$$

Setting

$$\tilde{c}_i = \sum_{k=1}^{N} p_{ik} c_k,$$

or

$$\tilde{c} = Pc,$$

we obtain from (2.10) that the coefficients $\tilde{c}$ solve the system of normal equations

$$A_s^* A_s \tilde{c} = A_s^* A c. \tag{2.12}$$

Theorem 1. Let the Gram matrix G,

$$G = \begin{pmatrix} A_s^* A_s & A_s^* A_{ns} \\ A_{ns}^* A_s & A_{ns}^* A_{ns} \end{pmatrix},$$

be such that its partial Cholesky decomposition has monotonically decaying pivots and is of the form

$$G = \begin{pmatrix} L_r & 0 \\ W & 0 \end{pmatrix} \begin{pmatrix} L_r^* & W^* \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & QQ^* \end{pmatrix}$$

where L_r is an $r \times r$ lower triangular matrix with the smallest diagonal entry $\epsilon > 0$. If the coefficients of the skeleton terms are computed via (2.12), then the difference between u in (2.1) and its approximation (2.8) can be estimated as

$$\|u - \widetilde{u}\|_2 = \|A_s \tilde{c} - Ac\|_2 \leq \|c\|_2 \sqrt{N-r}\, \epsilon^{1/2}. \tag{2.13}$$

Proof. We have

$$\|A_s \tilde{c} - Ac\|_2^2 = \|A_s Pc - Ac\|_2^2 = \langle c, \left(P^* A_s^* - A^*\right)\left(A_s P - A\right) c\rangle, \tag{2.14}$$

where the coefficient matrix P solves the normal equations (2.10) and, therefore,

$$A_s^* \left(A_s P - A\right) = 0, \tag{2.15}$$

as well as

$$A_s^* \left(A_s S - A_{ns}\right) = 0. \tag{2.16}$$

Using (2.15), we obtain

$$\left(P^* A_s^* - A^*\right)\left(A_s P - A\right) = A^* A - A^* A_s P \tag{2.17}$$

and proceed to compute $A^* A_s P$. We have $A^* = \begin{pmatrix} A_s^* \\ A_{ns}^* \end{pmatrix}$ and $A_s P = (A_s \mid A_s S)$, so that

$$A^* A_s P = \begin{pmatrix} A_s^* A_s & A_s^* A_s S \\ A_{ns}^* A_s & A_{ns}^* A_s S \end{pmatrix}.$$

Thus, we have

$$A^* A - A^* A_s P = \begin{pmatrix} 0 & A_s^* \left(A_{ns} - A_s S\right) \\ 0 & A_{ns}^* \left(A_{ns} - A_s S\right) \end{pmatrix}.$$

and, using (2.16), arrive at

$$A^* A - A^* A_s P = \begin{pmatrix} 0 & 0 \\ 0 & A_{ns}^* A_{ns} - A_{ns}^* A_s S \end{pmatrix}. \tag{2.18}$$

Equating the two expressions of the Gram matrix in the statement of the Theorem,

$$G = \begin{pmatrix} A_s^* A_s & A_s^* A_{ns} \\ A_{ns}^* A_s & A_{ns}^* A_{ns} \end{pmatrix} = \begin{pmatrix} L_r & 0 \\ W & 0 \end{pmatrix} \begin{pmatrix} L_r^* & W^* \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & QQ^* \end{pmatrix},$$

we obtain that

$$A_s^* A_s = L_r L_r^*, \tag{2.19}$$

$$A_{ns}^* A_s = W L_r^*, \quad A_s A_{ns}^* = L_r W^*, \tag{2.20}$$

and

$$A_{ns}^* A_{ns} = WW^* + QQ^*. \tag{2.21}$$

We observe that from (2.11) using (2.19) and (2.20), we arrive at

$$L_r L_r^* S = L_r W^*. \tag{2.22}$$

Next we show that the non-zero block of the matrix in the right hand side of (2.18) coincides with QQ^*. Using (2.20) and (2.22), we have

$$\begin{aligned} A_{ns}^* A_s S &= W L_r^* S \\ &= W \left(L_r^{-1} L_r \right) L_r^* S \\ &= W L_r^{-1} \left(L_r L_r^* S \right) \\ &= W L_r^{-1} \left(L_r W^* \right) = WW^*, \end{aligned}$$

where we used that L_r is non-singular. Hence, combining the last identity with (2.21), we obtain

$$A_{ns}^* A_{ns} - A_{ns}^* A_s S = WW^* + QQ^* - WW^* = QQ^*. \tag{2.23}$$

By (2.14), (2.17) , (2.18), and (2.23) we obtain

$$\begin{aligned} \|A_s \tilde{c} - Ac\|_2^2 &\leq \|c\|_2^2 \left\| \left(P^* A_s^* - A^* \right) \left(A_s P - A \right) \right\|_2 \\ &= \|c\|_2^2 \left\| A_{ns}^* A_{ns} - A_{ns}^* A_s S \right\|_2 \\ &= \|c\|_2^2 \left\| QQ^* \right\|_2 . \end{aligned}$$

Using Corollary 1, we estimate $\|QQ^*\|_2$ by its Frobenius norm,

$$\|QQ^*\|_2 \leq \|QQ^*\|_F \leq (N-r)\,\epsilon$$

which yields the desired estimate (2.13). □

Remark 1. The estimate (2.13) is tighter than the one obtained in [18, Theorem 3.1] since the dependence on the number of terms is $\mathcal{O}\left(N^{1/2}\right)$ instead of $\mathcal{O}\left(N^{3/4}\right)$. Yet, the estimate is still pessimistic since $\|QQ^*\|_2$ is usually significantly smaller than the Frobenius norm $\|QQ^*\|_F$. In practice, for N in the range 10^5-10^6, we did not observe the reduction of accuracy suggested by $\mathcal{O}\left(N^{1/2}\right)$.

2.1.4 Discussion

In Table 1 we present pseudo-code for the reduction Algorithm 1. Note that this algorithm is dimension independent (except, possibly, for the cost of computing the inner product which we always assume to be reasonable by the judicious choice of the functions in the mixture). As a consequence of Theorem 1 and Lemma 1, it is sufficient to generate only $N\times r$ entries of the Cholesky decomposition of the Gram matrix G implying that Algorithm 1 requires $\mathcal{O}\left(r^2N\right)$ operations.

2.2 Rank-revealing QR algorithm for reduction

We start by describing a rank-revealing QR algorithm based on the modified Gram-Schmidt approach for ordinary vectors. Given a set of N vectors v_i of length $M>N$, the rank-revealing QR algorithm can be used to identify the best subset of columns in the range of the matrix and, in its simplest form, cast as a pivoted version of the modified Gram-Schmidt.

Algorithm 1 Reduction algorithm using Gram matrix

Inputs: Coefficients c_l in representation of function $u(\mathbf{x}) = \sum_{l=1}^{N} c_l g_l(\mathbf{x})$ and error tolerance $10^{-14} \leq \epsilon < 1$. We assume that a subroutine to compute the inner product $\langle g_l, g_m \rangle$ is available.
Outputs: A pivot vector $I = \left[\widehat{I}, \widetilde{I}\right]$, where $\widehat{I} = [i_1, \ldots, i_r]$ contains indices of r skeleton terms and $\widetilde{I} = [i_{r+1}, \ldots i_N]$ indices of terms being removed from the final representation and the coefficients $\widetilde{c}_{i_m}, m = 1, \cdots r$, such that $\|u - \sum_{m=1}^{r} \widetilde{c}_{i_m} g_{i_m}\|_2 \leq \|c\|_2 \sqrt{N-r}\, \epsilon^{1/2}$ due to Theorem 1.

Stage 1: Pivoted Cholesky decomposition of the Gram matrix.

Initialization.

We maintain the diagonal $[d_1, d_2, \ldots, d_N]$ of the Cholesky factor L separately and initialize it as $\begin{bmatrix}1, & 1, & \cdots, & 1\end{bmatrix}$ (since all atoms have unit L^2-norm).
Set $r = 1$ and initialize a pivot vector as $I = [1, 2, \ldots, N]$.

```
for $l = 1, N$

    (1) Find the largest element of the diagonal and its index $i_j = \left\{i_j : \ d_{i_j} \geq d_{i_k}, \ \ k = l, \ldots, N\right\}$
        if $d_{i_j} < \epsilon$ goto Stage 2

    (2) Swap indices $i_j$ and $i_l$

    (3) Set the diagonal element of the matrix $L_{i_l,l} = (d_{i_l})^{1/2}$

        for $j = l+1, N$
            $L_{i_j,l} = \left(\langle g_{i_j}, g_{i_l} \rangle - \sum_{k=1}^{l-1} L_{i_l,k} L_{i_j,k}\right) / L_{i_l,l}$
            $d_{i_j} = d_{i_j} - L_{i_j,l}^2$
        end
        update: $r = r + 1$
end
```

Stage 2: Find new coefficients $\widetilde{c}_{i_m}, m = 1, \cdots, r$

```
    (1) Form a vector $b$ such that its $j$-th element is the inner product of
        $\sum_{i_m \in \widetilde{I}} c_{i_m} g_{i_m} = \sum_{m=r+1}^{N} c_{i_m} g_{i_m}$ and $g_{i_j}$.
        for $j = 1, r$
            $b(j) = \sum_{m=r+1}^{N} c_{i_m} \left(\sum_{k=1}^{j} L_{i_j,k} L_{i_m,k}\right)$
        end

    (2) Solve the linear system $\widehat{G}\widetilde{c} = b$, where $\widehat{G}_{jl} = \langle g_{i_j}, g_{i_l} \rangle = \sum_{k=1}^{N} L_{i_j k} L_{i_l k}$ and
        $\widetilde{c} = [\widetilde{c}_{i_1}, \widetilde{c}_{i_1}, \cdots, \widetilde{c}_{i_r}]^T$ using forward and backward substitution.

    (3) Add the original coefficients of the skeleton terms $c_{i_m}$ to $\widetilde{c}_{i_m}$ to get the new coefficients
        for $m = 1, r$
            $\widetilde{c}_{i_m} = \widetilde{c}_{i_m} + c_{i_m}$
        end
```

Algorithm 2 Rank-revealing QR algorithm

```
for $k = 1, N$
    (1) Compute $\tilde{v}_j = v_j - \langle v_j, v_k \rangle v_k$ for $j = k+1, \dots N$
    (2) Select $\tilde{v}_j$, $j = k+1, \dots N$, with the largest norm and set as $v_{k+1}$. Set $q_{k+1} = \frac{v_{k+1}}{\|v_{k+1}\|}$. Exit if the norm of $v_{k+1}$ is less than $\epsilon$.
end
```

For a given accuracy ϵ, if the resulting number of chosen vectors is r then the computational cost of this algorithm is at best $\mathcal{O}\left(r^2 N\right)$.

When constructing an algorithm for reducing the number of terms of multivariate mixtures, the difficulty is that the addition of two or more terms does not simplify and, thus, has to be maintained as a linear combination. For simplicity of indexing, we assume that the first r terms are the skeleton terms (we do not describe the swapping indices when pivoting). For a set of multivariate atoms $\{g_i(\mathbf{x})\}_{i=1}^N$, the Gram-Schmidt process works as

$$\begin{aligned}
\widetilde{\psi}_1(\mathbf{x}) &= g_1(\mathbf{x}), & \psi_1(\mathbf{x}) &= \frac{\widetilde{\psi}_1(\mathbf{x})}{\left\|\widetilde{\psi}_1\right\|} \\
\widetilde{\psi}_2(\mathbf{x}) &= g_2(\mathbf{x}) - p_{12}\widetilde{\psi}_1(\mathbf{x}), & \psi_2(\mathbf{x}) &= \frac{\widetilde{\psi}_2(\mathbf{x})}{\left\|\widetilde{\psi}_2\right\|} \\
\widetilde{\psi}_3(\mathbf{x}) &= g_3(\mathbf{x}) - p_{13}\widetilde{\psi}_1(\mathbf{x}) - p_{23}\widetilde{\psi}_2(\mathbf{x}), & \psi_3(\mathbf{x}) &= \frac{\widetilde{\psi}_3(\mathbf{x})}{\left\|\widetilde{\psi}_3\right\|} \\
&\vdots & &\vdots \\
\widetilde{\psi}_N(\mathbf{x}) &= g_N(\mathbf{x}) - \left(\sum_{i=1}^{N-1} p_{iN}\widetilde{\psi}_i(\mathbf{x})\right), & \psi_N(\mathbf{x}) &= \frac{\widetilde{\psi}_N(\mathbf{x})}{\left\|\widetilde{\psi}_N\right\|}
\end{aligned} \tag{2.24}$$

where

$$p_{ij} = \frac{\left\langle \widetilde{\psi}_i, g_j \right\rangle}{\left\langle \widetilde{\psi}_i, \widetilde{\psi}_i \right\rangle}, 1 \le i < j,$$

the functions $\widetilde{\psi}_i$ are orthogonal and ψ_i are orthonormal. We can also rewrite (2) as

$$\begin{aligned}
g_1(\mathbf{x}) &= p_{11}\widetilde{\psi}_1(\mathbf{x}), \\
g_2(\mathbf{x}) &= p_{12}\widetilde{\psi}_1(\mathbf{x}) + p_{22}\widetilde{\psi}_2(\mathbf{x}), \\
g_3(\mathbf{x}) &= p_{13}\widetilde{\psi}_1(\mathbf{x}) + p_{23}\widetilde{\psi}_2(\mathbf{x}) + p_{33}\widetilde{\psi}_3(\mathbf{x}), \\
&\vdots \\
g_N(\mathbf{x}) &= \sum_{i=1}^{N} p_{iN}\widetilde{\psi}_i(\mathbf{x}),
\end{aligned} \tag{2.25}$$

where $p_{ii} = 1$, $i = 1, \ldots, N$. On the other hand, we can obtain a set of coefficients q_{ij}, $1 \le i < j \le N$ such that

$$\begin{aligned}
\widetilde{\psi}_1(\mathbf{x}) &= q_{11}g_1(\mathbf{x}), \\
\widetilde{\psi}_2(\mathbf{x}) &= q_{12}g_1(\mathbf{x}) + q_{22}g_2(\mathbf{x}), \\
\widetilde{\psi}_3(\mathbf{x}) &= q_{13}g_1(\mathbf{x}) + q_{23}g_2(\mathbf{x}) + q_{33}g_3(\mathbf{x}), \\
&\vdots \\
\widetilde{\psi}_N(\mathbf{x}) &= \sum_{i=1}^{N} q_{iN}g_i(\mathbf{x}),
\end{aligned} \tag{2.26}$$

where $q_{ii} = 1$ for $i = 1, \ldots, N$.

Let us assume that the first k steps have been accomplished so that the coefficients p_{ij}, $1 \le i \le k$, $i \le j \le N$ in (2.25), the coefficients q_{ij},$1 \le i \le j \le k$ (2.26) and the norms of $\widetilde{\psi}_i$, $i = 1, \ldots, k+1$ are already available. Then at the $(k+1)$-th step, we first compute coefficients in (2.26) as

$$q_{i,k+1} = -\sum_{j=1}^{i} p_{j,k+1}q_{ij}, \quad 1 \le i \le k, \tag{2.27}$$

requiring $\mathcal{O}\left(k^2\right)$ operations at this step. Next we compute for $j = k+2, \ldots, N$

$$p_{k+1,j} = \frac{\left\langle \widetilde{\psi}_{k+1}, g_j \right\rangle}{\left\langle \widetilde{\psi}_{k+1}, \widetilde{\psi}_{k+1} \right\rangle},$$

where

$$\left\langle \widetilde{\psi}_{k+1}, g_j \right\rangle = \sum_{i=1}^{k+1} q_{i,k+1} \langle g_i, g_j \rangle. \tag{2.28}$$

Computing (2.28) is done for each index j so that it requires $\mathcal{O}(kN)$ operations at this step. Finally, we need to compute norms of $\widetilde{\psi}_j$ for $j = k+2, \cdots, N$. Since $\widetilde{\psi}_j = g_j - \sum_{i=1}^{k} p_{ik}\widetilde{\psi}_i$ are updated by subtracting $p_{k+1,j}\widetilde{\psi}_{k+1}$, $j = k+2, \cdots, N$, we compute the norm via

$$\begin{aligned}
& \left\| \widetilde{\psi}_j - p_{k+1j}\widetilde{\psi}_{k+1} \right\|^2 \\
= & \left\| g_j - \sum_{i=1}^{k} p_{ik}\widetilde{\psi}_i - p_{k+1j}\widetilde{\psi}_{k+1} \right\|^2 \\
= & \left\| g_j - \sum_{i=1}^{k} p_{ik}\widetilde{\psi}_i \right\|^2 + p_{k+1j}^2 \left\| \widetilde{\psi}_{k+1} \right\|^2 - 2p_{k+1j} \left\langle g_j - \sum_{i=1}^{k} p_{ik}\widetilde{\psi}_i, \widetilde{\psi}_{k+1} \right\rangle \\
= & \left\| g_j - \sum_{i=1}^{k} p_{ik}\widetilde{\psi}_i \right\|^2 + p_{k+1j}^2 \left\| \widetilde{\psi}_{k+1} \right\|^2 - 2p_{k+1j} \left\langle g_j, \widetilde{\psi}_{k+1} \right\rangle \qquad (2.29) \\
= & \left\| \widetilde{\psi}_j \right\|^2 + p_{k+1j}^2 \left\| \widetilde{\psi}_{k+1} \right\|^2 - 2p_{k+1j} \sum_{i=1}^{k+1} q_{il} \left\langle g_j, g_i \right\rangle . \qquad (2.30)
\end{aligned}$$

Note that at this stage functions $\widetilde{\psi}_j$, $j = k+2, \cdots, N$, are not orthogonal to $\widetilde{\psi}_{k+1}$. In the last line of (2.30), $\left\| \widetilde{\psi}_j \right\|^2$ and $p_{k+1j}^2 \left\| \widetilde{\psi}_{k+1} \right\|^2$ have been already computed, and evaluating $2p_{k+1j} \sum_{i=1}^{k+1} q_{il} \langle g_j, g_i \rangle$, $j = k+2, \cdots, N$ takes $\mathcal{O}(kN)$ operations.

In order to compute the coefficients of the new representation via skeleton terms, $\{\widetilde{c}_i\}_{i=1}^{r}$, such that

$$\sum_{i=1}^{N} c_i g_i(\mathbf{x}) \approx \sum_{j=1}^{r} \widetilde{c}_i g_i(\mathbf{x}),$$

we use the fact that

$$g_j(\mathbf{x}) = \sum_{k=1}^{j} p_{kj}\widetilde{\psi}_k(\mathbf{x}), \quad \widetilde{\psi}_j(\mathbf{x}) = \sum_{k=1}^{j} q_{kj} g_k(\mathbf{x})$$

and compute

$$\widetilde{c}_i = c_i + \sum_{j=r+1}^{N} c_j \left(\sum_{k=1}^{j} p_{kj} q_{ik} \right). \qquad (2.31)$$

Combining complexity estimates, if the number of skeleton terms is r, the resulting algorithm has a complexity $\mathcal{O}\left(r^3 + r^2 N\right)$, where the actual cost of $\mathcal{O}\left(r^3\right)$ part is not large if r is relatively small. Observing that this algorithm is designed to be used when $N \gg r$, we conclude that the overall cost is $\mathcal{O}\left(r^2 N\right)$. We observe that since the ill-conditioned Gram matrix, $\langle g_i, g_j \rangle$, appears in this algorithm and the computation of the norm is done via (2.30), the resulting approximation loses

1/2 of available digits as is the case in Algorithm 1. Possible improvement of this algorithm requires further investigation, and we plan to address it separately.

2.3 Reduction algorithm using frequency sampling

Due to the poor condition number of the Gram matrix, we lose half of the significant digits using Algorithm 1 (see examples in [18] and [59]). In order to identify "best" linear independent terms we can design a matrix with a better condition number if we use a family of "dual" functions for computing inner products instead of functions of the mixture. In the case of Gaussians (which are well localized), a natural set of such "dual" functions are exponentials with purely imaginary exponents (which are global functions); computing the inner product with them reduces to computing their Fourier transform. Therefore, as representatives of Gaussian atoms we can then use frequency vectors, i.e. samples of their Fourier transforms. The sampling strategy should be sufficient to differentiate between all Gaussian atoms; it is fairly straightforward to achieve this in low dimensions or if the functions admit a separated representation. Currently, we do not know how to do it efficiently in high dimensions. Naively it appears to require the construction of a sample matrix with $\mathcal{O}(N \times Ndr)$ entries and additional work is required to understand how to lower this number. Alternatively, in dimensions $d = 1, 2, 3$ it is sufficient to use $\mathcal{O}\left(N \times r^d\right)$ samples if we were to use the straightforward generalization of the algorithm in dimension $d = 1$ described below. In all cases, the last step in this approach is to compute the matrix ID of the sample matrix.

We present a deterministic algorithm in dimension $d = 1$ and note that its extension to functions in separated form in high dimensions can follow the approach in [18]. We consider a univariate Gaussian atom mixture

$$u(x) = \sum_{l=1}^{N} c_l g_l(x), \quad x \in \mathbb{R}, \tag{2.32}$$

where

$$g_l(x) = \frac{1}{\pi^{\frac{1}{4}} \sigma_l^{\frac{1}{2}}} e^{-\frac{(x-\mu_l)^2}{2\sigma_l^2}}, \quad \|g_l\|_2 = 1,$$

Algorithm 3 Reduction algorithm using rank-revealing QR

Inputs: Coefficients c_l in representation of function $u\left(\mathbf{x}\right)=\sum_{l=1}^{N}c_l g_l\left(\mathbf{x}\right)$ and error tolerance ϵ. We assume that a subroutine to compute the inner product $\langle g_l, g_m\rangle$ is available.

Outputs: A set of skeleton terms $g_{i_m}\left(\mathbf{x}\right)$ and their new coefficients $\widetilde{c}_{i_m}, m=1,\cdots r$ yielding an approximation $\widetilde{u}\left(\mathbf{x}\right)=\sum_{m=1}^{N}\tilde{c}_{i_m}g_{i_m}\left(\mathbf{x}\right)$ such that $\left|u\left(\mathbf{x}\right)-\widetilde{u}\left(\mathbf{x}\right)\right|=\mathcal{O}\left(\sqrt{\epsilon}\right)$.

```
Stage 1: Orthogonalization of $g_l$, $l=1,\cdots,N$.

Initialization.  We initialize of $\left\|\widetilde{\psi}_i\right\|=1$, for $i=1,\ldots,N$ and $p_{ii}=q_{ii}=1$ for $i=1,\ldots,N$.

for $k=1,N$

    (1) Compute $q_{i,k}$ for $1\leq i\leq k$ via (2.27).

    (2) Compute $p_{k,j}$ for $j=k+1,\ldots N$ via (2.28) to update $\widetilde{\psi}_j=g_j-p_{kj}g_k$

    (3) Update norm of $\widetilde{\psi}_j$ for $j=k+1,\ldots N$ via (2.30).

    (4) Select $\widetilde{\psi}_j$, $j=k+1,\ldots N$, with the largest norm and set as $g_{k+1}$. Exit if the norm of $g_{k+1}$
        is less than $\epsilon$.

end

Stage 2: Find new coefficients $\widetilde{c}_{i_m}, m=1,\cdots,r$.
We compute new coefficients via (2.31).
```

and seek the best linear independent subset as in (2.2). Defining the Fourier transform of f as

$$\hat{f}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\, e^{-ix\xi} dx,$$

we obtain

$$\hat{g}_l(\xi) = \frac{\sigma_l^{\frac{1}{2}}}{\pi^{\frac{1}{4}}} e^{-\frac{\sigma_l^2 \xi^2}{2}} e^{-i\mu_l \xi}.$$

In fact, for multivariate Gaussian atoms in any dimensions, the Fourier transform can be computed explicitly (see Appendix).

We set the highest frequency ξ_{high} of $\{\hat{g}_l(\xi)\}_{l=1}^{N}$ as

$$\xi_{high} = \sqrt{\frac{-2\log \frac{\pi^{\frac{1}{4}} 10^{-16}}{\sigma^{\frac{1}{2}}}}{\sigma^2}}, \quad \sigma = \min_{l=1,\cdots,N} \sigma_l,$$

such that

$$|\hat{g}_l(\xi)| < 10^{-16} \;\text{ for } \xi > \xi_{high} \text{ and } l = 1, \cdots, N.$$

We also set the lowest frequency $\xi_{low} = 10^{-2} \sim 10^{-3}$, a positive value obtained experimentally. We then sample the interval $[\xi_{low}, \xi_{high}]$ using frequencies $\xi_k, k = 1, \cdots r_p$, equally spaced on a logarithmic scale,

$$\xi_k = e^{\left(\log \xi_{low} + k \frac{\xi_{high} - \xi_{low}}{r_p}\right)},$$

where r_p is the number of samples. We choose $r_p > r$, where r is the expected final number of terms. In our setup, the column

$$\begin{bmatrix} \hat{g}_l(\xi_1) \\ \hat{g}_l(\xi_2) \\ \vdots \\ \hat{g}_l(\xi_{r_p}) \end{bmatrix}$$

serves as a representative of the Gaussian g_l. In this way, we reduce the problem to that of using the matrix ID. Specifically, given frequencies ξ_k, $k = 1, \cdots, r_p$, we construct a $r_p \times N$ sample matrix

Y,

$$Y=\begin{bmatrix}\hat{g_1}\left(\xi_1\right) & \hat{g_2}\left(\xi_1\right) & \cdots & \hat{g_N}\left(\xi_1\right)\\ \hat{g_1}\left(\xi_2\right) & \hat{g_2}\left(\xi_2\right) & \cdots & \hat{g_N}\left(\xi_2\right)\\ \vdots & \vdots & \ddots & \vdots\\ \hat{g_1}\left(\xi_{r_p}\right) & \hat{g_2}\left(\xi_{r_p}\right) & \cdots & \hat{g_N}\left(\xi_{r_p}\right)\end{bmatrix},$$

and compute its matrix ID (see e.g. [37]). We obtain a partition of indices $I=\left[\widehat{I},\widetilde{I}\right]$, where $\widehat{I}=[i_1,\ldots,i_r]$ and $\widetilde{I}=[i_{r+1},\ldots i_N]$ denote the skeleton and residual terms respectively. We also obtain a matrix X such that

$$Y=Y_{\left[:,\widehat{I}\right]}X,$$

where X is a $r\times N$ matrix that satisfies $X_{\left[:,\widehat{I}\right]}=I_r$. We then compute the new coefficients as

$$\tilde{c}_{i_m}=c_{i_m}+\sum_{n=r+1}^{N}c_{i_n}X_{mn},\;\; m=1,2,\ldots,r,$$

and use them to approximate

$$u\left(\mathbf{x}\right)\approx\sum_{m=1}^{r}\tilde{c}_{i_m}g_{i_m}\left(\mathbf{x}\right).$$

The accuracy of this approximation appears to be the same as the accuracy of matrix ID. Unfortunately, for multivariate Gaussian atoms, the size of the matrix Y appears to grow too fast with the dimension d (except in the case of separated representations where such dependence is linear). While our approach via frequency vectors can be extended in a straightforward manner to dimensions $d=2,3$, we note that the complexity of the Algorithm 1 via a Gram matrix is dimension independent and it would be of interest to construct a dimension independent algorithm yielding high accuracy approximations.

2.4 An acceleration technique for reduction algorithms

Recall that the complexity of Algorithm 1 is $\mathcal{O}\left(r^2N\right)$, where N is the initial number of terms and r is the number of skeleton terms. This algorithm obviously slows down when the number of skeleton terms, r, is large, since the computational cost grows quadratically in r. On the other

Algorithm 4 Reduction using frequency sampling (dimension $d = 1$)

Inputs: a function $u(\mathbf{x}) = \sum_{l=1}^{N} c_l g_l(\mathbf{x})$, number of frequencies samples r_p and error tolerance $\epsilon > 0$. We assume that a subroutine to compute the Fourier transform $\hat{g}_l(\xi)$ is available.

Outputs: an index set $I = \left[\widehat{I}, \widetilde{I}\right]$, where $\widehat{I} = [i_1, \ldots, i_r]$ contains indices of r skeleton terms and $\widetilde{I} = [i_{r+1}, \ldots i_N]$ contains indices of terms being removed from the final representation, and coefficients $\widetilde{c}_{i_m}, m = 1, \cdots r$, such that $u(\mathbf{x}) \approx \sum_{m=1}^{r} \widetilde{c}_{i_m} g_{i_m}(\mathbf{x})$ with accuracy of the matrix ID.

```
(1) set $\xi_{low} \in \left[10^{-2}, 10^{-3}\right]$ and compute $\xi_{high} = \sqrt{\frac{-2\log\frac{\pi^{\frac{1}{4}}10^{-16}}{\sigma^{\frac{1}{2}}}}{\sigma^2}}$ where $\sigma = \min_{l=1,\cdots,N}\{\sigma_l\}$.
(2) initialize $\xi_k = e^{\left(\log\xi_{low} + k\frac{\xi_{high}-\xi_{low}}{r_p}\right)}, k = 1, \cdots, r_p$.
(3) construct matrix $Y$ such that $Y_{kl} = \hat{g}_l(\xi_k)$.
(4) compute a matrix ID of $Y$ to obtain a index partition $\left[\widehat{I}, \widetilde{I}\right]$ and a matrix $X$ such that $Y = Y_{\left[:,\widehat{I}\right]} X$
(5) compute new coefficients via $\tilde{c}_{i_m} = c_{i_m} + \sum_{n=r+1}^{N} c_{i_n} X_{mn}$, for $m = 1, \ldots r$ .
```

hand, if two sets of functions are known to be linearly independent in advance, attempting to reduce their combination would be wasteful. In many practical applications it is expected that a function that models a complicated system will have a multivariate mixture representation with a relatively large number of terms. We note that usually such representation can be split into subgroups of terms so that each group reflects local behavior of the function. For instance, in electronic structure calculations, approximation of the electron orbitals require many terms to resolve the cusps near the nuclei centers but Gaussians associated with these terms decay rapidly away from the nuclear center. Thus, these terms will be nearly orthogonal to similar terms at other nuclear centers. This suggests an acceleration technique for the reduction algorithm based on a hierarchical subdivision of the terms of the initial mixture.

As an illustration, let us consider a multivariate mixture with N terms where the number of linearly independent terms is r. Let us assume that we can subdivide these terms into m groups (equally, for simplicity). The number of linearly independent terms in each group in at most r/k, where $k > 1$. The cost of performing the reduction algorithm on one group is $\mathcal{O}\left(\frac{N}{m}\left(\frac{r}{k}\right)^2\right)$ and, thus, the total cost to reduce all the groups is $\mathcal{O}\left(m\frac{N}{m}\left(\frac{r}{k}\right)^2\right) = \mathcal{O}\left(\left(\frac{r}{k}\right)^2 N\right)$. Clearly, an additional reduction may be needed if we want to reduce to r terms. Depending on k, we get a significant speed-up factor (in our examples of solving Hartree-Fock equations, $k \approx 7$). We note that (while we did not use it in our computations) that the reduction of independent groups of terms is trivially parallel.

In the thesis we use this acceleration technique in the computations for solving the Hartree-Fock equations in Chapter 3, as well as the Poisson's equation in Chapter 4. We plan to do a careful speed comparison and further develop applications of this technique. .

2.5 Numerical Example

In order to compare the performance of Algorithm 1, 3 and 4 in dimension $d = 1$, we consider a univariate Gaussian mixture (2.32), choose $N = 10000$, and sample c_l, σ_l and μ_l from uniform distributions $\mathcal{U}(-1,1)$, $\mathcal{U}(0,20)$, and $\mathcal{U}(-5,5)$, respectively. We apply Algorithms 1, 3 and 4 to reduce the number of terms in the Gaussian mixture and display the original function and the errors of approximations in Figure 2.1.

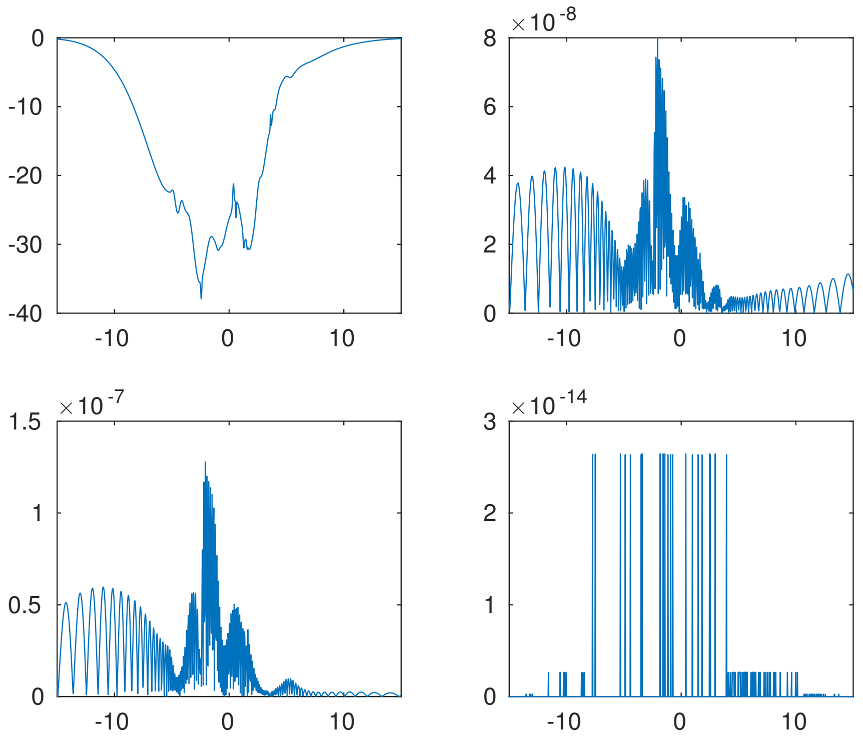

Figure 2.1: A function represented via a Gaussian mixture with 10^4 terms (top left). Relative errors obtained by using Algorithm 1 yielding 121 terms (top right), Algorithm 3 yielding 122 terms (bottom left) and Algorithm 4 yielding 273 terms (bottom right).

Chapter 3

Adaptive algorithm for electronic structure calculations

3.1 Hartree-Fock equation for diatomic molecules

In this Chapter, we present a new adaptive method for electronic structure calculations based on algorithms for reduction of multiresolution multivariate mixtures described in Chapter 2. While we represent solutions using a linear combination of Gaussians (which has a long history in quantum chemistry), these Gaussians are not selected in advance as a basis but are generated in the process of solving equations. Thus, we avoid the so-called "basis error" usually associated with methods using Gaussians. Our approach can also be characterized as a "gridless" or "meshfree" method.

Using Gaussians to find solutions of quantum chemistry problems has its origins in seminal papers [20, 53, 64] motivated by the fact that integrals involving these functions can be evaluated efficiently. In these papers the authors proposed to use linear combinations of Gaussians whose exponents and coefficients were found (or optimized) via Newton's method in order to capture the correct behavior near the nuclear cusps and the correct rate of decay. However, this approach proved unsustainable as problems became larger. Instead, the construction of a basis for spatial orbitals has been performed off-line and the resulting sets of functions were then used as a fixed basis with unknown coefficients, leading to the so-called "basis error" if the actual solution is not well approximated within the linear span of such basis.

Using bases of so constructed sets of Gaussians have revolutionized computational quantum chemistry in spite of the absence of a systematic way for controlling accuracy (and not providing guaranteed error bounds). As a result, selecting a basis set became an art form requiring insight into the underlying solution. However, once a basis set is selected, the accuracy of the solution obtained using such basis is ultimately limited. The limitation on accuracy should be understood in the context of approximate equations that are being solved, e.g. the Hartree-Fock equations or the Kohn-Sham equations of density functional theory (DFT) (see e.g. [57]), making it difficult to separate the impact of approximate equations vs. approximate methods of solving them. Thus, adaptive methods are highly desirable in this field.

The advent of multiresolution analysis (see e.g. [25]) laid a conceptual foundation for adaptive methods but it took some time before practical adaptive algorithms were developed using multiwavelets [2], see [40, 68, 69]. A central element of the success of the multiresolution algorithms can be traced to the fact that physically significant (integral) operators arising in problems of quantum chemistry are naturally represented by radial kernels which can be accurately approximated by a linear combination of Gaussians. The key advantage of using approximations via Gaussians is that they yield a representation in which operators are efficiently applied in each dimension separately. Without such separated representations multiresolution operators would be too expensive to be used in dimensions three and higher. Multiresolution methods systematically refine numerical grids (or basis functions) in the vicinity of the cusp-type singularities while using a relatively few grid points (or basis functions) elsewhere. These methods have proven successful in efficiently computing highly accurate solutions, achieving guaranteed error bounds and eliminating the basis error. This approach has been implemented at Oak Ridge National Laboratory as a software package MADNESS (A Multiresolution, ADaptive Numerical Environment for Scientific Simulation), see [39], and is now considered the most accurate approach in this field [45] (c.f. a mixed-basis method using plane-waves and atom-centered radial polynomials [56], interlocking multi-center grids [62], etc.).

However, since multiwavelets are piecewise polynomials, they do not resemble the spatial orbitals and a large number of basis functions is needed to represent solutions, e.g. an individual orbital may require $\approx 2 \cdot 10^5$ basis functions in 3D. Moreover, such local refinement schemes do not take advantage of the essential simplicity of the spatial orbitals far from the nuclei and require boundary conditions to limit the computational domain. While adaptive multiresolution methods are sufficiently fast to be used within one-particle theories of quantum chemistry, due to computational costs advancement towards two particle theories or solving Schrödinger's equation had a limited success (see e.g. [12, 13]).

Thus, constructing new adaptive schemes to compete with MADNESS is of interest. As it was demonstrated in [9], it is possible to iteratively solve equations of Quantum Chemistry using new nonlinear algorithms which reduce the number of terms in intermediate representations without resorting to Newton's method. The results in [9] are based on using Slater-type orbitals. In this thesis, we present an adaptive method using linear combinations of Gaussians to represent the solutions. We demonstrate the new approach by solving the Hartree-Fock equations for Helium Hydride (HeH+) and Lithium Hydride (LiH) so that we can compare the resulting representations of solutions with those in [40, 9]. As in [40, 9], we formulate the problem using integral equations and use a convergent iteration to solve them. We use linear combinations of Gaussians to represent not only operators and potentials but also the functions on which they operate. As a result, all integrals can be computed explicitly and exactly by simply updating the parameters of the Gaussians. The computational effort thus moves from that of approximating and computing integrals to that of maintaining a reasonable number of terms in intermediate representations of solutions within the iterative scheme. We use algorithm 1 to reduce the number of terms after each operation by finding the "best" linearly independent terms to represents solutions thus maintaining a reasonably small number of them. There is no underlying grid to maintain (thus, "gridless" or "meshfree" method) and there is no need to impose boundary conditions as in [40].

3.2 Separated representations of the Green's functions

In this section, we start by reviewing a key result developed in [15, 16]. It essentially says that the power functions $r^{-\alpha}$, $\alpha > 0$, can be well represented by a sum of Gaussians with near-optimal numbers of parameters.

Theorem 2. (Beylkin and Monzón [16]) For any $\alpha > 0$, $\delta < 0$, and $1/e \geq \epsilon > 0$, there exist a step size h and integers M and N such that

$$\left|r^{-\alpha} - G_F\left(r; M, N, h\right)\right| \leq r^{-\alpha}\epsilon, \quad \text{for all } \delta \leq r \leq 1,$$

where

$$G_F\left(r; M, N, h\right) = \frac{h}{\Gamma\left(\alpha/2\right)} \sum_{n=M+1}^{N} e^{\alpha h n/2} e^{-e^{hn} r^2}. \tag{3.1}$$

The error estimates is based on discretizing the integral

$$\frac{1}{r^\alpha} = \frac{1}{\Gamma\left(\frac{\alpha}{2}\right)} \int_{-\infty}^{\infty} e^{-r^2 e^t + \frac{\alpha}{2}t} dt$$

using trapezoidal rule as

$$\frac{1}{\Gamma\left(\frac{\alpha}{2}\right)} \int_{-\infty}^{\infty} e^{-r^2 e^t + \frac{\alpha}{2}t} dt \approx \frac{h}{\Gamma\left(\frac{\alpha}{2}\right)} \sum_{n \in \mathbb{Z}} e^{\alpha h n/2} e^{-e^{hn} r^2}, \tag{3.2}$$

where the step size h satisfies

$$h \leq \frac{2\pi}{\log 3 + \frac{\alpha}{2}\log(\cos 1)^{-1} + \log \epsilon^{-1}}$$

and ϵ is any user-selected accuracy. For a given accuracy ϵ, power α and a range of values r, the infinite sum (3.2) is then truncated due to the exponential or super exponential decay of the integrand at $\pm\infty$, to yield a finite sum approximation in that range. This approximation provides an analytic construction of a multiresolution separated representation for the Poisson kernel in any dimensions and the Coulomb potential. For example, setting $\alpha = 1$ in (3.1), we obtain the approximation of the Poisson kernel in $\mathbb{R}^3$ as a sum of Gaussians.

Another function of interest in this thesis is the Green's function for the non-oscillatory Helmholtz equations,

$$G\left(\mu, r\right) = \left(\frac{\mu}{r}\right)^p K_p\left(\mu r\right),$$

where K_p is the modified Bessel function of the second kind. Similar results can be obtained by discretizing the integral representation

$$2^{p+1}\left(\frac{\mu}{r}\right)^p K_p\left(\mu r\right) = \int_{-\infty}^{\infty} e^{-r^2 e^{-t}/4 - \mu^2 e^{-t} + pt} dt.$$

Note that in this representation the variables μ and r are separated and the integrand has a super-exponential decay at $\pm\infty$.

3.3 The Helium Hydride Ion HeH+

As our first example, we solve the Hartree-Fock equation

$$\left(-\frac{1}{2}\Delta + V - 4\pi\Delta^{-1}\left(|\phi|^2\right)\right)\phi = E\phi, \tag{3.3}$$

with the potential

$$V(\mathbf{r}) = \frac{Z_1}{\|\mathbf{r}-\mathbf{R}_1\|} + \frac{Z_2}{\|\mathbf{r}-\mathbf{R}_2\|}.$$

For the Helium Hydride Ion, HeH+, we have $Z_1 = -1$, $Z_2 = -2$ and $\mathbf{R}_1 = (0,0,-0.7)$ and $\mathbf{R}_2 = (0,0,0.7)$.

As in [40, 68, 69, 9], our basic approach involves recasting (3.3) as an integral equation which we solve iteratively. The iteration (see below) is convergent and while the rate of convergence is not quite quadratic, it is sufficiently fast so that only a few dozen iterations are required (see discussion in [40] and reference therein). In contrast to [40, 68, 69, 9], instead of using multiwavelet bases, or [9] where Slater-type orbitals are used, we represent $\phi(\mathbf{r})$ as a Gaussian mixture in (2.1) that is constructed via Basis Generating Iteration. In each step of the iterative solution, the number of terms in representing $\phi(\mathbf{r})$ grows and we use Algorithm 1 to control their number. To be precise, we write (3.3) as

$$\left(-\Delta + \mu^2\right)\phi = -2V_\phi\phi, \tag{3.4}$$

where

$$\mu^2 = -2E$$

and

$$V_\phi = V - 4\pi\Delta^{-1}\left(|\phi|^2\right).$$

We approximate both, the potential $V(\mathbf{r})$ by discretizing the integral

$$\begin{aligned} V(\mathbf{r}) &= \frac{Z_1}{\|\mathbf{r}-\mathbf{R}_1\|} + \frac{Z_2}{\|\mathbf{r}-\mathbf{R}_2\|} \\ &= \frac{Z_1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-\|\mathbf{r}-\mathbf{R}_1\|^2 e^t + \frac{t}{2}}dt + \frac{Z_2}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-\|\mathbf{r}-\mathbf{R}_2\|^2 e^t + \frac{t}{2}}dt \end{aligned}$$

as

$$V_\infty(\mathbf{r},h) = \frac{Z_1 h}{\sqrt{\pi}} \sum_{l_1\in\mathbb{Z}} e^{\frac{hl_1}{2}} e^{-e^{hl_1}\|\mathbf{r}-\mathbf{R}_1\|^2} + \frac{Z_2 h}{\sqrt{\pi}} \sum_{l_2\in\mathbb{Z}} e^{\frac{hl_2}{2}} e^{-e^{hl_2}\|\mathbf{r}-\mathbf{R}_2\|^2} \tag{3.5}$$

and the Green's function $G(\mu,\mathbf{r})$ for bound state (non-oscillatory) Helmholtz equation

$$\left(-\Delta+\mu^2\right)\phi = f$$

by discretizing the integral

$$\begin{aligned} G(\mu,\mathbf{r}) &= \frac{1}{4\pi}\frac{e^{-\mu\|\mathbf{r}\|}}{\|\mathbf{r}\|} \\ &= (4\pi)^{-\frac{3}{2}} \int_{-\infty}^{\infty} e^{-\frac{\|\mathbf{r}\|^2 e^t}{4} - \mu^2 e^{-t} + \frac{t}{2}} dt \end{aligned}$$

as

$$G_\infty(\mu,\mathbf{r},h) = (4\pi)^{-\frac{3}{2}}\, h \sum_{l\in\mathbb{Z}} e^{-\mu^2 e^{-hl} + \frac{hl}{2}} e^{-\frac{\|\mathbf{r}\|^2 e^{hl}}{4}}, \tag{3.6}$$

where the step size h is selected to achieve the desired accuracy ϵ. We then truncate the sums (3.5) and (3.6) in the same manner as described in Section 3.2 and obtain

$$\widetilde{V}(\mathbf{r}) = \frac{Z_1 h}{\sqrt{\pi}} \sum_{l_1=M_1}^{N_1} e^{\frac{hl_1}{2}} e^{-e^{hl_1}\|\mathbf{r}-\mathbf{R}_1\|^2} + \frac{Z_2 h}{\sqrt{\pi}} \sum_{l_2=M_2}^{N_2} e^{\frac{hl_2}{2}} e^{-e^{hl_2}\|\mathbf{r}-\mathbf{R}_2\|^2}$$

and

$$\widetilde{G}_\mu(\mathbf{r}) = (4\pi)^{-\frac{3}{2}}\, h \sum_{l=M}^{N} e^{-\mu^2 e^{-hl} + \frac{hl}{2}} e^{-\frac{\|\mathbf{r}\|^2 e^{hl}}{4}}.$$

Note that $\widetilde{G}_0(\mathbf{r})$ approximates the Green's function for the Poisson's equation

$$-\Delta\phi = f.$$

We solve (3.4) via the iteration,

$$\begin{aligned} V_\phi^{(n+1)} &\leftarrow V - 4\pi\widetilde{G}_0 * \left(\left|\phi^{(n)}\right|^2\right) \\ \phi^{(n+1)} &\leftarrow -2\widetilde{G}_{\mu^{(n)}} * \left(V_\phi^{(n+1)}\phi^{(n)}\right), \\ E^{(n+1)} &\leftarrow E^{(n)} - \frac{\left\langle \phi^{(n)} - \phi^{(n+1)}, V_\phi^{(n+1)}\phi^{(n)} \right\rangle}{\left\|\phi^{(n+1)}\right\|}, \\ \phi^{(n+1)} &\leftarrow \frac{\phi^{(n+1)}}{\left\|\phi^{(n+1)}\right\|}, \\ \mu^{(n+1)} &\leftarrow \sqrt{-2E^{(n+1)}}, \quad n = 0,1,\cdots \end{aligned} \tag{3.7}$$

where $*$ denotes convolution. Since all operators and functions are approximated as Gaussian mixtures, all operations such as multiplication and convolution are computed analytically.

For the initial guess of the solution, we use a single Gaussian centered at the origin with the orbital energy $E = -2$. We verify that, for this choice of parameters, after 19 iterations, the orbital energy $E = -1.66054711273118$ is computed with six significant digits, and with 3.4×10^{-6} absolute error. The comparison is made using the MADNESS [29] software yielding the orbital energy $E = -1.6605437$. The total number of term to represent the orbital $\phi(\mathbf{r})$ is 1738. The number of terms is significantly smaller than $\approx 2 \cdot 10^5$ that required by multiresolution method [40, 68, 69] and by a factor ≈ 3 larger than that in [9], where the number of Slater-type orbitals is 637. However, our approach is much simpler than that in [9] and, for this reason, is easier to extend to work with large molecules.

In Figure 3.1, we display the spatial orbital $\phi(\mathbf{r})$ on the line $\mathbf{r} = (0, 0, x)$ connecting the two nuclei locations $\mathbf{R}_1$ and $\mathbf{R}_2$. Note that both cusps are not symmetric (clearly visible for one of them).

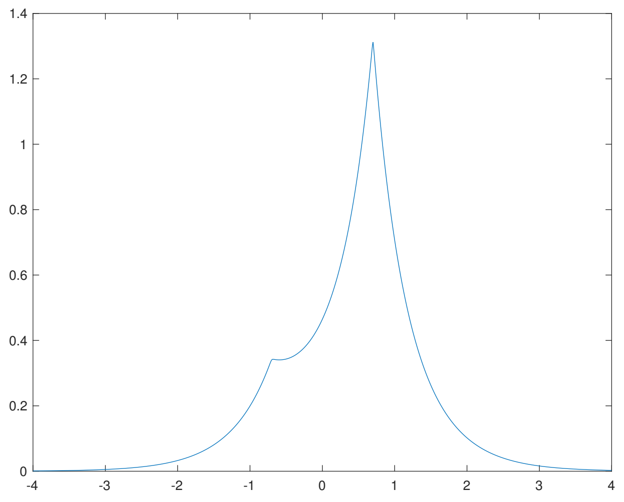

Figure 3.1: Plot of the spatial orbital $\phi(\mathbf{r})$ for Helium Hydride, HeH+ on the line $\mathbf{r} = (0, 0, x)$ connecting the two nuclear centers $\mathbf{R}_1 = (0, 0, -0.7)$ and $\mathbf{R}_2 = (0, 0, 0.7)$.

3.4 Lithium Hydride Ion LiH

For our second example, we consider the Hartree-Fock equations for the Lithium Hydride, LiH. The Hartree-Fock equations in this case are

$$F\phi_j(\mathbf{r}) = E_j\phi(\mathbf{r}), \quad j = 1, 2,$$

where $F = -\frac{1}{2}\Delta + V + 2J - K$,

$$J\phi_j = \phi_j\left(-4\pi\Delta^{-1}\left(|\phi_1|^2 + |\phi_2|^2\right)\right),$$

$$K\phi_j = \phi_1\left(-4\pi\Delta^{-1}\left(\phi_1^*\phi_j\right)\right) + \phi_2\left(-4\pi\Delta^{-1}\left(\phi_2^*\phi_j\right)\right),$$

and

$$V(\mathbf{r}) = \frac{Z_1}{\|\mathbf{r} - \mathbf{R}_1\|} + \frac{Z_2}{\|\mathbf{r} - \mathbf{R}_2\|}.$$

For Lithium Hydride Ion LiH, we have $Z_1 = -3$, $Z_2 = -1$ and $\mathbf{R}_1 = (-3.15/2, 0, 0)$ and $\mathbf{R}_2 = (3.15/2, 0, 0)$. We solve this equation using the same iteration as described in Section 3.3, with the two modifications that are standard procedures for solving the Hartree-Fock equations with two or more orbitals. First, the spatial orbitals are orthogonalized after each iteration. Second, the orbital energies are obtained via solving the eigenvalues of the 2×2 matrix $H_{ij} = \langle \mathcal{F}\phi_i, \phi_j \rangle$ after each iteration.

To be precise, we choose an initial guess of the orbitals, $\phi_j^{(0)}(\mathbf{r})$, $j = 1, 2$, without specifying the corresponding orbitals energies $E_j^{(0)}$ $j = 1, 2$. We orthonormalize $\phi_j^{(0)}(\mathbf{r})$ and then compute the initial energies $E_j^{(0)}$ as the eigenvalues of the 2×2 matrix $H_{ij}^{(0)} = \left\langle \mathcal{F}\phi_i^{(0)}, \phi_j^{(0)} \right\rangle$. Setting $\left(\mu_j^{(0)}\right)^2 = -2E_j^{(0)}$, $j = 1, 2$, we then iterate the orbitals via

$$\begin{pmatrix} \phi_1^{(n+1)} \\ \phi_2^{(n+1)} \end{pmatrix} \leftarrow \begin{pmatrix} -2G_{\mu_1^{(n)}} * (V + 2J - K)\phi_1^{(n)} \\ -2G_{\mu_2^{(n)}} * (V + 2J - K)\phi_2^{(n)} \end{pmatrix}, \quad n = 0, 1, \cdots \tag{3.8}$$

After each iteration, $\phi_j^{(n+1)}(\mathbf{r})$ are orthogonalized, orbital energies $E_j^{(n+1)}$ are updated via solving the eigenvalues of the 2×2 matrix $H_{ij}^{(n+1)} = \left\langle \mathcal{F}\phi_i^{(n+1)}, \phi_j^{(n+1)} \right\rangle$ and $\mu_j^{(n+1)}$ are computed via $\left(\mu_j^{(n+1)}\right)^2 = -2E_j^{(n+1)}$.

For the Hartree-Fock equations with n orbitals, the total electron energy is defined as

$$E = \sum_{j=1}^{n} \left(E_j + \left\langle \left(-\frac{1}{2}\Delta + V \right) \phi_j, \phi_j \right\rangle \right),$$

and the total energy, E_{tot}, is calculated by adding the nucleus–nucleus repulsion energies to the total electron energy. For Lithium Hydride, LiH, we have

$$E_{tot} = \sum_{j=1}^{2} \left(E_j + \left\langle \left(-\frac{1}{2}\Delta + V \right) \phi_j, \phi_j \right\rangle \right) + \frac{Z_1 Z_2}{\|\boldsymbol{R}_1 - \boldsymbol{R}_2\|}.$$

For the initial guess of the spatial orbitals, we use Gaussian mixtures approximating

$$\phi_1(\mathbf{r}) \approx \frac{1}{\sqrt{\pi}} e^{-\|\mathbf{r}-\mathbf{R}_1\|} \quad \text{and} \quad \phi_2(\mathbf{r}) \approx \frac{1}{\sqrt{\pi}} e^{-\|\mathbf{r}-\mathbf{R}_2\|},$$

which are then orthogonalized. After 15 iterations, the computed orbital energies are $E_1 = -2.45174443318099$ and $E_2 = -0.297831582093949$ and agree to five and four significant digits with the values $E_1 = -2.451763$ and $E_2 = -0.297823$ computed by the MADNESS software, and have absolute errors of 1.9×10^{-5} and 8.9×10^{-6}. The computed total energy $E_{tot} = -7.98693757026112$ agrees to five significant digits with the value $E_{tot} = -7.9869364$ computed in MADNESS, and has an absolute error of 1.2×10^{-6}. The total number of term to represent the orbitals $\phi_1(\mathbf{r})$ and $\phi_2(\boldsymbol{r})$ is 3126 and 3981. The number of terms is again much smaller (about 100 times smaller) than that obtained using MADNESS software and by a factor ≈ 3 larger than that in [9], where the number of terms to represent the Slater-type orbitals is 1282 and 1327.

In Figure 3.2 we display the spatial orbitals $\phi_1(\mathbf{r})$ and $\phi_2(\mathbf{r})$ on the line $\mathbf{r} = (x, 0, 0)$ connecting the two nuclei locations $\mathbf{R}_1$ and $\mathbf{R}_2$.

3.5 Discussion

Our implementation of the examples in 3.3 and 3.4 are written in Fortran90 and compiled with the Intel Fortran Compiler version 18.0.3. All computations are performed on a single core (without parallelization) of an Intel i7-6700 CPU at 3.4 GHz on a 64-bit Linux workstation with

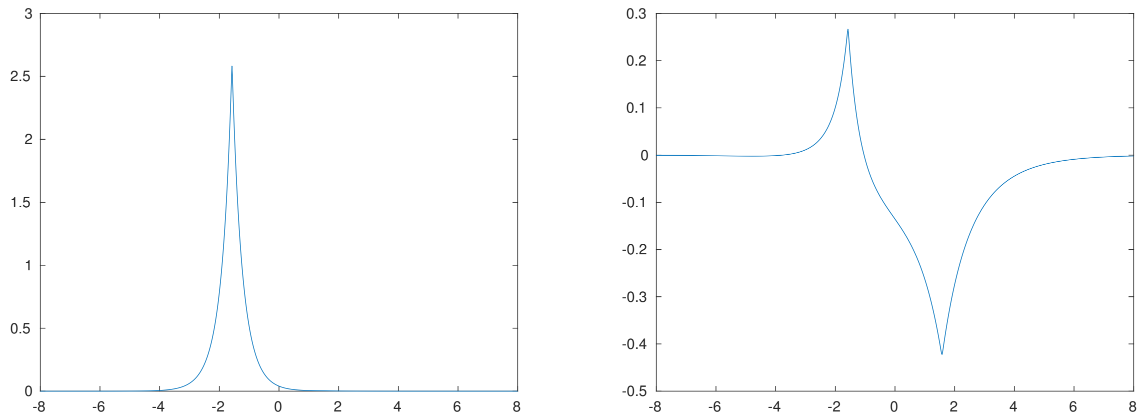

Figure 3.2: Plot the spatial orbitals for Lithium Hydride, LiH, $\phi_1(\mathbf{r})$ (left) and $\phi_2(\boldsymbol{r})$ (right) on the line $\mathbf{r} = (x, 0, 0)$ connecting the two nuclei locations $\mathbf{R}_1 = (-3.15/2, 0, 0)$ and $\mathbf{R}_2 = (3.15/2, 0, 0)$.

64 GB of RAM. Currently it takes about 6.4 minutes and 28.4 minutes to solve the Hartree-Fock equation for HeH+ and LiH, respectively. We made no attempt to optimize our implementation.

The reduction algorithm with splitting Gaussian atoms into groups (see Section 2.4) is trivially parallel but we did not take advantage of this in the current version of our code. Since there are about hundred groups and the main cost is in reduction, this acceleration factor should bring the timing to close to that of MADNESS (which is highly efficient code well optimized during more than 10 years of its development). We plan to do a careful speed comparison with other existing methods separately.

Chapter 4

Reduction algorithms for solving differential and integral equations

4.1 Poisson equation in free space in high dimensions

The Poisson's equation

$$-\Delta u(\mathbf{x}) = f\left(\mathbf{x}\right), \quad \mathbf{x} \in \mathbb{R}^d, \tag{4.1}$$

arises in numerous applications in nearly all field of physics and computational chemistry (see e.g. [36]). Reduction Algorithm 1 allows us to solve this equation in dimensions $d \geq 3$ assuming that the charge distribution $f\left(\mathbf{x}\right)$ is given by, e.g. , a linear combination of multivariate Gaussian atoms. We obtain the solution via

$$u(\mathbf{x}) = \int_{\mathbb{R}^d} G\left(\mathbf{x}-\mathbf{y}\right) f(\mathbf{y}) d\mathbf{y}, \tag{4.2}$$

where the free-space Green's function for (4.1) is given by the radial function

$$G(\mathbf{x}) = C_d \|\mathbf{x}\|^{2-d}, \quad C_d = \frac{\Gamma\left(\frac{d}{2}+1\right)}{d\left(d-2\right)\pi^{\frac{d}{2}}}, \tag{4.3}$$

where $\|\cdot\| = \|\cdot\|_2$ is the standard l^2-norm.

4.1.1 Approximation of the Green's via Gaussians

In order to evaluate the integral (4.2), we approximate the Green's function G via a linear combination of Gaussians (see e.g. [40, 15, 16]). The error estimates in [16, Theorem 3] are based on discretizing the integral

$$\frac{1}{r^{d-2}} = \frac{1}{\Gamma\left(\frac{d-2}{2}\right)} \int_{-\infty}^{\infty} e^{-r^2 e^t + \frac{d-2}{2}t} dt$$

as

$$G_\infty(r,h) = \frac{C_d h}{\Gamma\left(\frac{d-2}{2}\right)} \sum_{l\in\mathbb{Z}} e^{hl(d-2)/2} e^{-e^{hl} r^2}, \tag{4.4}$$

where the step size h satisfies

$$h \le \frac{2\pi}{\log 3 + \frac{d-2}{2}\log(\cos 1)^{-1} + \log\epsilon^{-1}} \tag{4.5}$$

and ϵ is any user-selected accuracy. Then [16, Theorem 3] implies

$$|G(r) - G_\infty(r,h)| \le \epsilon\, G(r), \quad \text{for all } r > 0. \tag{4.6}$$

4.1.2 Solution method and error estimation

To estimate the error of approximating the solution u in (4.2) using the series (4.4) instead of the Green's function (4.3), we first prove the following lemma.

Lemma 2. For any $d \ge 3$, $e^{-1} \ge \epsilon > 0$, and f nonnegative in (4.1), there exist a step size h such that

$$\left| u(\mathbf{x}) - \int_{\mathbb{R}^d} G_\infty(\|\mathbf{x}-\mathbf{y}\|, h) f(\mathbf{y}) d\mathbf{y} \right| \le \epsilon\, |u(\mathbf{x})|, \quad \text{for all } \ \mathbf{x} \ne 0. \tag{4.7}$$

Proof. From (4.2) and (4.6), we have

$$\begin{aligned}
\left| u(\mathbf{x}) - \int_{\mathbb{R}^d} G_\infty(\|\mathbf{x}-\mathbf{y}\|, h) f(\mathbf{y}) d\mathbf{y} \right| &= \left| \int_{\mathbb{R}^d} \left[G(\mathbf{x}-\mathbf{y}) - G_\infty(\|\mathbf{x}-\mathbf{y}\|, h)\right] f(\mathbf{y}) d\mathbf{y} \right| \\
&\le \int_{\mathbb{R}^d} |G(\mathbf{x}-\mathbf{y}) - G_\infty(\|\mathbf{x}-\mathbf{y}\|, h)|\, |f(\mathbf{y})|\, d\mathbf{y} \\
&\le \epsilon \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y})\, |f(\mathbf{y})|\, d\mathbf{y} \\
&= \epsilon \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y})\, f(\mathbf{y}) d\mathbf{y} \\
&= \epsilon\, u(\mathbf{x}) \\
&= \epsilon\, |u(\mathbf{x})|.
\end{aligned}$$

□

In our examples, we always consider functions f represented in the form

$$f(\mathbf{x}) = \sum_{l=1}^{N} c_l g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right), \tag{4.8}$$

for some Gaussian atoms as in (1.5). In particular,

$$u(\mathbf{x}) = \sum_{l=1}^{N} c_l \int_{\mathbb{R}^d} G\left(\mathbf{x}-\mathbf{y}\right) g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right) d\mathbf{y}$$

and

$$u^{+}(\mathbf{x}) = \sum_{l=1}^{N} |c_l| \int_{\mathbb{R}^d} G\left(\mathbf{x}-\mathbf{y}\right) g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right) d\mathbf{y}$$

are both bounded. We assume that

$$\left\|u^{+}\right\|_{L^\infty} \leq c\left\|u\right\|_{L^\infty} \tag{4.9}$$

for a moderate size constant c. This assumption prevents representations of u that involve large coefficients c_l of opposite signs. We then have

Lemma 3. Let $d \geq 3$, $e^{-1} \geq \epsilon > 0$, and f as in (4.8). If (4.9) holds, then there exist a step size h such that

$$\left|\int_{\mathbb{R}^d} G\left(\mathbf{x}-\mathbf{y}, h\right) f(\mathbf{y}) d\mathbf{y} - \int_{\mathbb{R}^d} G_\infty\left(\left\|\mathbf{x}-\mathbf{y}\right\|, h\right) f(\mathbf{y}) d\mathbf{y}\right| \leq \epsilon\, c\left|u(\mathbf{x})\right|, \quad \text{for all} \quad \mathbf{x} \neq 0. \tag{4.10}$$

Proof. It follows from Lemma 2 that

$$\left|u(\mathbf{x}) - \tilde{u}(\mathbf{x})\right| \leq \epsilon u^{+}(\mathbf{x})$$

where

$$\tilde{u}(\mathbf{x}) = \int_{\mathbb{R}^d} G_\infty\left(\left\|\mathbf{x}-\mathbf{y}\right\|, h\right) f(\mathbf{y}) d\mathbf{y}.$$

Therefore, using (4.9) , we have

$$\left\|u(\mathbf{x}) - \tilde{u}\right\|_{L^\infty} \leq \epsilon u^{+}(\mathbf{x}) \leq \epsilon\, c\left\|u\right\|_{L^\infty}.$$

$\square$

In practice, when using (4.4), we truncate the sum

$$\begin{aligned} G_{M,N}(r) &= \frac{C_d h}{\Gamma\left(\frac{d-2}{2}\right)} \sum_{l=M}^{N} e^{hl(d-2)/2} e^{-e^{hl} r^2} \qquad (4.11) \\ &= \frac{C_d h}{\Gamma\left(\frac{d-2}{2}\right)} \sum_{l=1}^{N_{terms}} e^{h(M+l-1)(d-2)/2} e^{-e^{h(M+l-1)} r^2} \end{aligned}$$

so that the removed terms contribute less than ϵ and we limit the range of r to some interval of the form $[\delta, R]$; the resulting approximation has $N_{terms} = N - M + 1$.

In our computations we select the accuracy range of $G_{M,N}$ to be $\left[10^{-7}, 10^{10}\right]$ with $\epsilon = 10^{-14}$. The number of terms in (4.11) depends on the dimension only weakly (see Table 4.1).

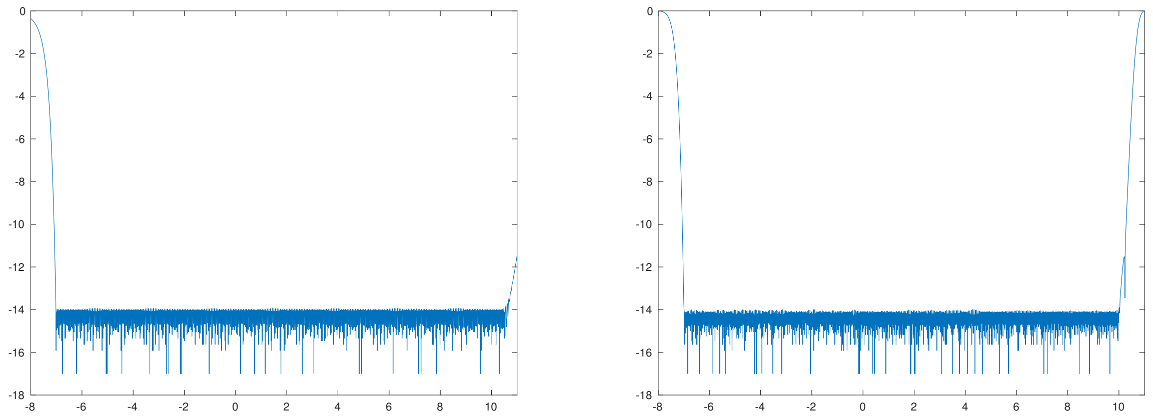

Figure 4.1: Plot of the error $\log_{10}\left|1 - r^{\beta} G_{M,N}(r)\right|$. Shown on left: $\beta = 1$, $G_{M,N}(r)$ has 298 terms and achieves relative accuracy $= 10^{-14}$ in the interval $\left[10^{-7}, 10^{10}\right]$. Shown on right: $\beta = 5$, $G_{M,N}(r)$ has 338 terms and achieves relative accuracy $= 10^{-14}$ in the interval $\left[10^{-7}, 10^{10}\right]$.

As an illustration, we first demonstrate our approach for a single Gaussian,

$$f(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}. \qquad (4.12)$$

Using (4.11), we approximate the solution u by

$$\begin{aligned} u_{\epsilon}(\mathbf{x}) &= \frac{C_d h}{\Gamma\left(\frac{d-2}{2}\right)} \sum_{l=1}^{N_{terms}} e^{h(M+l-1)(d-2)/2} \int_{\mathbb{R}^d} e^{-e^{h(M+l-1)} \|\mathbf{x}-\mathbf{y}\|^2} f(\mathbf{y}) d\mathbf{y} \\ &= \frac{C_d h}{\Gamma\left(\frac{d-2}{2}\right)} \sum_{l=1}^{N_{terms}} e^{h(M+l-1)(d-2)/2} \int_{\mathbb{R}^d} e^{-e^{h(M+l-1)} \|\mathbf{x}-\mathbf{y}\|^2} e^{-\frac{1}{2}(\mathbf{y}-\boldsymbol{u})^T \boldsymbol{\Sigma}^{-1} (\mathbf{y}-\boldsymbol{\mu})} d\mathbf{y} \qquad (4.13) \end{aligned}$$

Evaluating the integral explicitly (see Appendix A for details), we obtain

$$u_\epsilon(\mathbf{x}) = \sum_{l=1}^{N_{terms}} c_l g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right) \tag{4.14}$$

where

$$c_l = \frac{C_d h \pi^{\frac{3d}{4}}}{\Gamma\left(\frac{d-2}{2}\right)} e^{-h(M+l-1)(d+2)/2} \frac{(\det \Sigma)^{\frac{1}{2}}}{(\det \Sigma_l)^{\frac{1}{4}}},$$

$$\boldsymbol{\Sigma}_l = \Sigma + \begin{bmatrix} \frac{1}{2e^{h(M+l-1)}} & & & 0 \\ & \frac{1}{2e^{h(M+l-1)}} & & \\ & & \ddots & \\ 0 & & & \frac{1}{2e^{h(M+l-1)}} \end{bmatrix},$$

and

$$\boldsymbol{\mu}_l = \boldsymbol{\mu}.$$

The number of terms in the representation of u_ϵ is excessive and we apply Algorithm 1 to reduce their number and obtain our final approximation as

$$\tilde{u}(\mathbf{x}) = \sum_{m=1}^{\widetilde{N}} \tilde{c}_{i_m} g_{i_m}\left(\mathbf{x}, \boldsymbol{\mu}_{i_m}, \boldsymbol{\Sigma}_{i_m}\right), \tag{4.15}$$

where $\tilde{c}_{i_m}$ are the new coefficients and $i_m \in \widehat{I}$ and (see Algorithm 1).

Remark 2. The representation of the kernel in (4.11) can be obtained for a large spatial range since the number of terms N_{terms} is proportional to the logarithm of the range. For this reason our approach can work where employing the Fast Fourier Transform is not an option due to the the size of the domain, c.f. [67].

In order to demonstrate the performance of our approach, we choose the right hand side f to be a Gaussian mixture with 100 terms,

$$f(\mathbf{x}) = \sum_{i=1}^{100} c_{f_i} e^{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_{f_i}\right)^T \boldsymbol{\Sigma}_{f_i}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_{f_i}\right)}.$$

In the Gaussian mixture f, the coefficients c_{f_i} and means $\boldsymbol{\mu}_{f_i}$ are sampled from a one and a d-dimensional standard normal distributions respectively. The symmetric positive definite matrices

$\mathbf{\Sigma}_{f_i}$ are constructed as

$$\mathbf{\Sigma}_{f_i} = U_i^T U_i + \frac{1}{10} I_d, \tag{4.16}$$

where U_i is a $d \times d$ matrix of standard normally distributed numbers and I_d is the $d \times d$ identity matrix. We obtain u_ϵ in (4.14) and apply Algorithm 1 to reduce the number of terms to get $\tilde{u}$ in (4.15). The results are displayed in Table 4.2, where we show the dimension of the problem, d, the number of terms, N_{terms}, in the approximation of the Green's function, the number of terms, N_{tot}, in the solution u_ϵ before reduction and its accuracy, the number of terms, $\widetilde{N}$, in the solution $\widetilde{u}$ after reduction and its accuracy, and, finally, the relative error between u_ϵ and $\widetilde{u}$.

4.1.3 Error verification

We use the following approach to estimate the accuracy of the solution $\tilde{u}$ of (4.1). We define the error

$$h_{error}(\mathbf{x}) = -\Delta \tilde{u}(\mathbf{x}) - f(\mathbf{x}),$$

and, to ascertain its size, compute the value of h_{error} at a collection of points in all principle directions of the right hand side $f(\mathbf{x})$. To be precise, we first solve the eigenvalue problem for all matrices Σ_{f_i} , $i = 1, \ldots, 100$,

$$\Sigma_{f_i} = \sum_{j=1}^{d} \lambda_j^{(i)} \mathbf{v}_j^{(i)} \left(\mathbf{v}_j^{(i)}\right)^T.$$

Here the eigenvectors $\mathbf{v}_j^{(i)}$ identify principle directions for each Gaussian in f so that we can select an appropriate set of samples along those directions. We note that the actual range of the eigenvalues of matrices Σ_{f_i}, $\left\{\lambda_j^{(i)}\right\}_{\substack{i=1,..,100\\ j=1,\ldots,d}}$, is $\left[\frac{1}{10}, 40\right]$. Next, for each pair of $\left\{\lambda_j^{(i)}, \mathbf{v}_j^{(i)}\right\}$, we find an interval $\left[-s_j^{(i)}, s_j^{(i)}\right]$ by solving

$$\left| e^{-\frac{1}{2}\left(s_j^{(i)}\mathbf{v}_j^{(i)}\right)^T \mathbf{\Sigma}_{f_i}^{-1}\left(s_j^{(i)}\mathbf{v}_j^{(i)}\right)} \right| = \left| e^{-\frac{\left(s_j^{(i)}\right)^2}{2\lambda_j^{(i)}}} \right| = 10^{-10} \Leftrightarrow s_j^{(i)} = \left(-2\lambda_j^{(i)} \log 10^{-10}\right)^{1/2}$$

and generate and equally-spaced grid in $\left[-s_j^{(i)}, s_j^{(i)}\right]$ as

$$s_{jk}^{(i)} = k \frac{2 s_j^{(i)}}{N_{s_j^{(i)}}}, \quad k = 0, \ldots N_s.$$

Finally, we select sample points

$$\mathbf{x}_{jk}^{(i)} = s_{jk}^{(i)} \mathbf{v}_j^{(i)} + \boldsymbol{\mu}_{f_i}$$

and evaluate $h_{error}\left(\mathbf{x}_{jk}^{(i)}\right)$ for $i = 1, \dots, 100$, $j = 1, \dots, d$ and $k = 1, \dots, N_s$.

4.1.4 Numerical results

In our experiment, we select $N_s = 1$ and $N_s = 100$, and report the resulting errors in Table 4.1 and 4.2. We also report the required CPU time, T, for reducing the number of terms in u_ϵ in seconds for the examples when $N_s = 100$. For these two tables, we use the notation $\|f\|_\infty = \max_{i,j,k}\left|f\left(\mathbf{x}_{jk}^{(i)}\right)\right|$.

Our code are implemented in Fortran90 and compiled with the Intel Fortran Compiler version 18.0.3. The computations are performed on a single core (without parallelization) of an Intel i7-6700 CPU at 3.4 GHz on a 64-bit Linux workstation with 64 GB of RAM. Currently it takes minutes (see Table 4.2) to solve the Poisson's equation in multi-dimensions with no attempt of optimization of the code. The reduction algorithm with splitting Gaussian atoms into groups (see Section 2.4) is trivially parallel but we did not take advantage of this in the current version of our code. We expect that the parallelization would speed up our initial implementation by a factor of between 10 and 100.

d	N_{terms}	N_{tot}	$\frac{\|-\Delta u_\epsilon - f\|_\infty}{\|f\|_\infty}$	$\widetilde{N}$	$\frac{\|-\Delta \widetilde{u} - f\|_\infty}{\|f\|_\infty}$	$\frac{\|u_\epsilon - \widetilde{u}\|_\infty}{\|u_\epsilon\|_\infty}$
3	345	278	$1.2e-13$	183	$3.4e-8$	$5.0e-10$
4	397	333	$1.7e-14$	223	$2.8e-7$	$3.6e-9$
5	386	323	$8.0e-15$	204	$5.6e-8$	$1.2e-9$
6	343	311	$4.6e-14$	194	$3.0e-7$	$7.0e-9$
7	354	328	$2.3e-14$	206	$6.9e-8$	$2.1e-9$

Table 4.1: Number of terms and relative errors of solving Poisson's equation in dimensions $d = 3, \dots, 7$ where the forcing term is a single randomly generated multivariate Gaussian. The number of Gaussians to represent the Green's function in (4.11) is N_{terms}, the number of terms of u_ϵ in (4.14) after truncation of coefficients to 10^{-10} is N_{tot} and, after applying Algorithm 1, the number of terms of $\tilde{u}$ in (4.15) is $\widetilde{N}$.

d	N_{terms}	N_{tot}	$\frac{\|-\Delta u_\epsilon - f\|_\infty}{\|f\|_\infty}$	$\widetilde{N}$	$\frac{\|-\Delta \widetilde{u} - f\|_\infty}{\|f\|_\infty}$	$\frac{\|u_\epsilon - \widetilde{u}\|_\infty}{\|u_\epsilon\|_\infty}$	T
3	345	27918	$7.6e-14$	4407	$7.4e-5$	$4.2e-7$	$8.6e1$
4	397	32872	$5.8e-14$	5156	$5.5e-5$	$9.3e-7$	$1.8e2$
5	386	31660	$4.6e-14$	5773	$1.6e-5$	$1.3e-7$	$2.4e2$
6	343	31557	$2.8e-14$	6772	$1.1e-6$	$2.2e-8$	$2.8e2$
7	354	32706	$1.5e-14$	7623	$8.1e-7$	$2.4e-8$	$3.8e2$

Table 4.2: Number of terms, relative errors and CPU time (in seconds) of solving Poisson's equation in dimensions $d = 3, \ldots, 7$ where the forcing term is a linear combination of 100 randomly generated multivariate Gaussians. The information displayed in each column is described in Table 4.1.

4.2 Second order elliptic equation with a variable coefficient

We consider the second order linear elliptic equation with a variable coefficient,

$$-\nabla \cdot (a(\mathbf{x}) \nabla u(\mathbf{x})) + k^2 u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d \tag{4.17}$$

where $d \geq 3$ and $k > 0$. We assume that the variable coefficient $a(\boldsymbol{x})$ is of the form

$$a(\mathbf{x}) = 1 + e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_a)^t \Sigma_a^{-1} (\mathbf{x}-\boldsymbol{\mu}_a)}, \tag{4.18}$$

such that $\max_{\mathbf{x}} |a(\mathbf{x})| / \min_{\mathbf{x}} |a(\mathbf{x})| = 2$, and choose the forcing function to be

$$f(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_f)^t \Sigma_f^{-1} (\mathbf{x}-\boldsymbol{\mu}_f)}, \quad \|f\|_\infty = 1. \tag{4.19}$$

4.2.1 Approximation of the Green's function via Gaussians

The free space Green's function for the problem with a constant coefficient

$$-\Delta u(\mathbf{x}) + k^2 u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d$$

is

$$G(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} \left(\frac{k}{\|\mathbf{x}\|} \right)^{\frac{d}{2}-1} K_{\frac{d}{2}-1}(k\|\mathbf{x}\|), \tag{4.20}$$

where $K_{\frac{d}{2}-1}$ is the modified Bessel function of the second kind of order $\frac{d}{2}-1$. We approximate the Green's function (4.20) by discretizing the integral

$$G(\mathbf{x}) = (4\pi)^{-\frac{d}{2}} \int_{-\infty}^{\infty} e^{-\frac{\|\boldsymbol{x}\|^2 e^t}{4} - k^2 e^{-t} + \left(\frac{d}{2}-1\right)t} dt$$

as

$$G_{\infty}(r) = (4\pi)^{-\frac{d}{2}} h \sum_{l\in\mathbb{Z}} e^{-\frac{r^2 e^{hl}}{4} - k^2 e^{-hl} + \left(\frac{d}{2}-1\right)hl},$$

where the step size h is selected to achieve the desired accuracy ϵ. We then truncate the sum in the same manner as described in Section (3.2) and obtain

$$\begin{aligned} G_{M,N}(r) &= (4\pi)^{-\frac{d}{2}} h \sum_{l=M}^{N} e^{-\frac{r^2 e^{hl}}{4} - k^2 e^{-hl} + \left(\frac{d}{2}-1\right)hl} \\ &= (4\pi)^{-\frac{d}{2}} h \sum_{l=1}^{N_{terms}} e^{-\frac{r^2 e^{h(l+M-1)}}{4} - k^2 e^{-h(l+M-1)} + \left(\frac{d}{2}-1\right)h(l+M-1)}, \end{aligned}$$

where the number of terms $N_{terms} = M - N + 1$ in $G_{M,N}$ weakly depends on the dimension (see Table 4.4). In our computation, $k = 1$ and we select the accuracy range of $G_{M,N}$ to be $\left[10^{-7}, 10^{2}\right]$ with $\epsilon = 10^{-10}$.

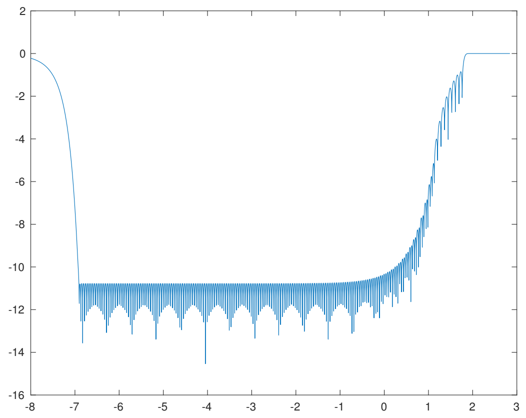
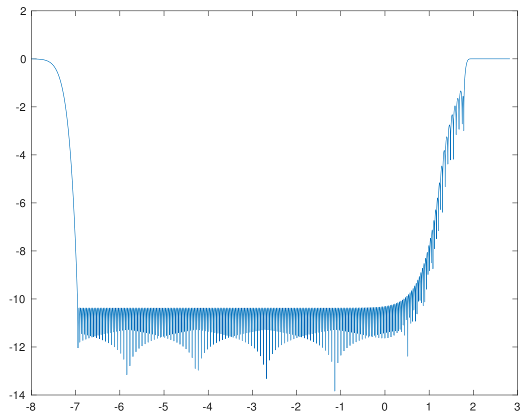

Figure 4.2: Plot of the error $\log_{10}\left|\frac{G(r)-G_{M,N}(r)}{G(r)}\right|$. Shown on left: $d = 3$, $G_{M,N}(r)$ has 104 terms and achieves relative accuracy $= 10^{-10}$ in the interval $\left[10^{-7}, 10^{2}\right]$. Shown on right: $d = 7$, $G_{M,N}(r)$ has 127 terms and achieves relative accuracy $= 10^{-10}$ in the interval $\left[10^{-7}, 10^{2}\right]$.

4.2.2 Solution method

We rewrite (4.17) as an integral equation,

$$u(\mathbf{x}) - \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y}) \nabla \cdot \left((a(\mathbf{y}) - 1) \nabla u(\mathbf{y})\right) d\mathbf{y} = \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y}) f(\mathbf{y}) d\mathbf{y}. \tag{4.21}$$

To solve (4.21), our approach has two steps. First we look for a set of Gaussian atoms to be used as a basis to approximate the solution (4.21). In this step we perform several iterations of the integral equation (without computing the coefficients corresponding to each atom) followed by reduction. Our approach is based on the following argument. It can be shown that, in a multiresolution basis, for a finite accuracy $\epsilon > 0$, the non-standard form (see [6]) of the Green's function for (4.21) is banded on all scales. This implies that the set of basis functions to represent the solution is fully determined by the size of the bands of the Green's function and the right hand side. This suggests that if we try to solve the integral equation (4.21) iteratively, the functions that this iteration will generate should be sufficient to approximate the solution due to the interaction between the essential supports of the functions involved (even if the fixed-point iteration does not converge).

Once the necessary Gaussian atoms are identified, we look for the solution in the form

$$\tilde{u}(\mathbf{x}) = \sum_{l=1}^{\widetilde{N}} c_l g_l(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l),$$

for some coefficients c_l, $l = 1, \cdots \widetilde{N}$, to be determined; we then substitute this ansatz for $\tilde{u}$ into either the integral equation or the differential equation and derive a system of linear algebraic equations for c_l by computing appropriate inner products.

To demonstrate our approach, we rewrite (4.21) as

$$u(\mathbf{x}) = \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y}) \nabla \cdot ((a(\mathbf{y})-1)\nabla u(\mathbf{y}))\, d\mathbf{y} + \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y}) f(\mathbf{y})\, d\mathbf{y}$$

and setup the iteration,

$$\begin{aligned} u_{n+1}(\mathbf{x}) &= u_n(\mathbf{x}) + \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y}) \nabla \cdot ((a(\mathbf{y})-1)\nabla u_n(\mathbf{y}))\, d\mathbf{y} \qquad (4.22) \\ u_0(\mathbf{x}) &= \int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{y}) f(\mathbf{y})\, d\mathbf{y} \end{aligned}$$

We perform one (or several) iteration(s), without computing coefficients, producing a large number of Gaussian atoms to represent the solution. Using Algorithm 1 after each iteration, we reduce the number of atoms by removing linearly dependent terms. In this way we find a basis of Gaussian atoms for the solution of the equation (4.17).

Once the set of Gaussian atoms is selected, to find the coefficients c_l, $l = 1, \cdots \widetilde{N}$, we use the weak formulation of (4.17). Substituting the selected Gaussian atoms with unknown coefficients into (4.17), we solve the resulting linear system

$$\sum_{l=1}^{\widetilde{N}} c_l \left\langle -\nabla \cdot \left(a\left(\mathbf{x}\right) \nabla g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right)\right), g_k\left(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right\rangle = \left\langle f\left(\mathbf{x}\right), g_k\left(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right\rangle, \ k = 1, \cdots, \widetilde{N}. \quad (4.23)$$

The inner products $\left\langle -\nabla \cdot \left(a\left(\mathbf{x}\right) \nabla g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right)\right), g_k\left(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right\rangle$ are computed explicitly using integration by parts and the fact that

$$\nabla_{\boldsymbol{x}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})} = -\nabla_{\boldsymbol{\mu}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})},$$

leading to integrals involving only Gaussians (the result is then differentiated with respect to the shift parameter $\boldsymbol{\mu}$).

We solve the linear system (4.23) using the SVD to obtain an approximate solution

$$\widetilde{u}\left(\mathbf{x}\right) = \sum_{l=1}^{\widetilde{N}} c_l g_l\left(\mathbf{x}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right).$$

4.2.3 Error verification

In order to verify that $\widetilde{u}$ is an approximate solution of (4.17), we note that

$$h_{error}\left(\mathbf{x}\right) = -\nabla \cdot \left(a\left(\mathbf{x}\right) \nabla \tilde{u}\left(\mathbf{x}\right)\right) + k^2 \tilde{u}\left(\mathbf{x}\right) - f\left(\mathbf{x}\right)$$

is a combination of Gaussians and products of Gaussians with low degree polynomials. We compute inner products of h_{error} with a collection of exponentials, i.e., explicitly compute the Fourier transform of h,

$$\widehat{h}_{error}\left(\xi\right) = \frac{1}{\left(2\pi\right)^{\frac{d}{2}}} \int_{\mathbb{R}^d} \left(-\nabla \cdot \left(a\left(\mathbf{x}\right) \nabla \tilde{u}\left(\mathbf{x}\right)\right) + k^2 \tilde{u}\left(\mathbf{x}\right) - f\left(\mathbf{x}\right)\right) e^{-i\pi \mathbf{x} \cdot \xi} d\mathbf{x}. \quad (4.24)$$

The Fourier transform, $\widehat{h}_{error}$ is computed explicitly since all functions in (4.24) are represented via Gaussians or products of Gaussians and low degree polynomials. We then evaluate $\widehat{h}_{error}$ for selected vector arguments ξ, which we call frequency vectors. To select these vectors, we use the principal directions of the matrices $\mathbf{\Sigma}_l$ of the Gaussian atoms in the representation of $\widehat{h}_{error}$. To this end, we solve the eigenvalue problem

$$\mathbf{\Sigma}_l = \sum_{j=1}^{d} \lambda_j^{(l)} \mathbf{v}_j^{(l)} \left(\mathbf{v}_j^{(l)}\right)^T$$

and select the frequency vectors along the principal directions of Σ_l. In our experiment, we choose $s_{min} = 10^{-5}$ and

$$s_{max} = \left(-2\log\left(10^{-10}\right)/\lambda_{min}\right)^{\frac{1}{2}},$$

where $\lambda_{min} = \min_{l,j} \lambda_j^{(l)}$, such that

$$e^{-\frac{1}{2}\left(sv_j^{(l)}\right)^T \mathbf{\Sigma}\left(sv_j^{(l)}\right)} = e^{-\frac{\lambda_j^{(l)} s^2}{2}} \leq 10^{-10}$$

for $s \in [s_{min}, s_{max}]$, $l = 1, \dots n$ and $j = 1, \dots d$. We then sample s_i,$i = 1, \dots, N_s$, using a logarithmic scale on the interval $[s_{min}, s_{max}]$

$$s_k = e^{\log\left(s_{min} + k\frac{s_{max} - s_{min}}{N_s - 1}\right)},$$

and select the frequency vectors $\boldsymbol{\xi}_j^{(l)}$ with components

$$\left(\boldsymbol{\xi}_j^{(l)}\right)_k = s_k v_j^{(l)}.$$

4.2.4 Numerical results

We notice that the number of terms in the solution $\widetilde{u}$ grows significantly with the dimension, if the matrices $\mathbf{\Sigma}_a$ and $\mathbf{\Sigma}_f$ are not related (they are effectively random) and/or the range of their eigenvalues is large. In our first experiment, we select matrices $\mathbf{\Sigma}_a$ and $\mathbf{\Sigma}_f$ in (4.18)-(4.19) to be

$$\Sigma_a = UD_aU^T, \quad \text{and} \quad \Sigma_f = UD_fU^T,$$

where U is a $d \times d$ random unitary matrix and D_a and D_f are $d \times d$ diagonal matrices. We set the first two diagonal entries of D_a and D_f to be 0.1 and 20, and sample the other diagonal entry/entries from a uniform distribution $\mathcal{U}(0.1, 20)$. A random permutation is applied after all diagonal entries are generated. In the second experiment, we construct matrices $\mathbf{\Sigma}_a$ and $\mathbf{\Sigma}_f$ to be

$$\Sigma_a = U_aD_aU_a^T, \quad \text{and} \quad \Sigma_f = U_fD_fU_f^T$$

where U_a and U_f are $d \times d$ random unitary matrices, D_a and D_f are $d \times d$ diagonal matrices. We set the first two diagonal entries of D_a and D_f to be 0.1 and 1, and sample the other diagonal entry/entries from a uniform distribution $\mathcal{U}(0.1, 1)$. Again we randomly permute the diagonals of D_a and D_f. We also notice that if the centers μ_a and μ_f are far away (no overlapping essential supports), then solving (4.17) is effectively the same as solving the Poisson's equation. In our tests, we select $\mu_f = \mathbf{0}$ and $\mu_a = (1, 0, \ldots, 0)^T$ so that $\|\mu_a - \mu_f\|_2 = 1$. The results are displayed in Tables 4.3 and 4.4 where we show the dimension of the problem, d, the number of terms N_{terms} in the approximation of the Green's function, the number of terms N_{tot} obtained by performing one iteration in (4.22), and the number of terms $\widetilde{N}$ in the solution $\widetilde{u}$ after reduction and its accuracy.

dimension	N_{terms}	N_{tot}	$\widetilde{N}$	error
3	104	5985	1184	$1.8e-6$
4	109	11624	2649	$3.5e-6$
5	117	16014	3640	$3.8e-6$
6	121	22466	4377	$8.0e-5$
7	127	31522	5573	$2.5e-5$

Table 4.3: Results of solving a second order elliptic equation with a variable coefficient where the principle directions of matrices Σ_a and Σ_f in 4.18 and 4.19 are aligned and their eigenvalues are in the range $(0.1, 20)$.

dimension	N_{terms}	N_{tot}	$\widetilde{N}$	error
3	104	15348	2586	$8.1e-7$
4	109	32688	6883	$1.9e-6$
5	117	40110	11793	$2.4e-6$
6	121	64664	18205	$6.7e-5$
7	127	75059	22966	

Table 4.4: Results of solving a second order elliptic equation with a variable coefficient where the principle directions of matrices Σ_a and Σ_f in 4.18 and 4.19 are not aligned and their eigenvalues are in the range $(0.1, 1)$. We note that the accuracy estimation in dimension $d = 7$ is computationally expensive and we skipped it.

Chapter 5

Kernel Density Estimation

We describe a new approach to Kernel Density Estimation (KDE) based on Algorithm 1. KDE is a non-parametric method for constructing the PDF of data points used in cluster analysis, classification, and machine learning. Note that the standard KDE construction is practical only in low dimensions, $d = 1, 2, 3$ as it requires a Fourier transform of the data points, the cost of which grows exponentially with dimension. Our approach avoids using the Fourier transform and is applicable in high dimensions. We note that a randomized approach that can be used for KDE estimation was recently suggested in [54].

The essence of KDE is to associate a smooth PDF f with data points $\mathbf{x}_j \in \mathbb{R}^d$, $j = 1, \dots, N$,

$$f(\mathbf{x}, h) = \frac{1}{N} \sum_{j=1}^{N} K_h(\mathbf{x} - \mathbf{x}_j), \tag{5.1}$$

where $K_h(\mathbf{x}) = K(\mathbf{x}/h)/h$ and K is a nonnegative function with zero mean and $\int_{\mathbb{R}^d} K(\mathbf{x})\, d\mathbf{x} = 1$. In what follows, we use a multivariate Gaussian as the kernel K. A naive implementation of (5.1) would require N evaluations of the kernel K_h for each point $\mathbf{x}$ so that the computational cost of using this approach in a straightforward manner is prohibitive if N is large. We note that the selection of the parameter h, the so-called bandwidth or scale parameter, is a well recognized delicate issue and, in our one dimensional example, we use h computed within Mathematica^{TM} implementation of KDE.

In our approach, for a user selected target accuracy ϵ, we seek a *subset* of linear independent terms in (5.1) and express the rest of the terms as their linear combinations. Thus, by removing redundant terms in the representation of $f(\mathbf{x}, h)$, we construct

$$F(\mathbf{x}, h) = \sum_{\ell=1}^{r} a_\ell K_h(\mathbf{x} - \mathbf{x}_{j_\ell}), \tag{5.2}$$

where $r \ll N$ and

$$|f(\mathbf{x}, h) - F(\mathbf{x}, h)| \leq \epsilon.$$

In other words, with accuracy ϵ, we obtain an approximation of the function f by a function F with a small number of terms. This reduction algorithm seeking a subset of linear independent terms in (5.1) can be used for any kernel in $\mathbb{R}^d$ such that the cost of evaluating the multidimensional inner product

$$g_{ij} = \int_{\mathbb{R}^d} K_h(\mathbf{x} - \mathbf{x}_i) K_h(\mathbf{x} - \mathbf{x}_j) \, d\mathbf{x}$$

is reasonable (e.g. does not depend on the dimension d or depends on it mildly). For multivariate Gaussians, the values g_{ij} are available explicitly.

The computational cost of our algorithm is $\mathcal{O}(r^2 N)$ and is independent of the underlying dimension d. Here N is the original number of data points and r is the final number of terms in the chosen linearly independent subset.

5.1 A comparison in dimension $d = 1$

In low dimensions, the standard approach to KDE relies on using the Fast Fourier transform to both, assist in estimating the bandwidth parameter h and in constructing a more efficient representation of (5.1) on an equally spaced grid (see e.g. [63, Section 3.5]). Implementations of this approach can be found in many packages in dimensions $d = 1, 2$, e.g. Matlab, Mathematica, etc. While this approach is appropriate in low dimensions, an extension of this algorithm to high dimensions is prevented by the "curse of dimensionality". Thus, in high dimensions, only values at selected points can be computed (see [19] and Matlab implementation of KDE in high dimensions).

In order to illustrate our approach we provide a simple example with a bimodal distribution in dimension $d = 1$. Although this example is in one variable, it allows us to emphasize the differences between the existing KDE methods and our approach. We generate test data by using two normal distributions with means $\mu_1 = 0$ and $\mu_2 = 4$. The exact PDF of this data is given by

$$g(x) = \frac{1}{2}\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} + \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\frac{(x-4)^2}{\sigma^2}}\right), \tag{5.3}$$

where $\sigma = 1$. We then use KDE implemented in Mathematica^{TM} with the Gaussian kernel K. Using $N = 10^5$ data samples drawn from (5.3) (so that the initial sum (5.1) has N terms), the scaling parameter was set to $h = 0.20121412622314902019$. The true distribution (5.3) and the approximation error obtained by Mathematica^{TM} by reducing (5.1) from $N = 10^5$ terms to 241 Gaussian terms centered on an equally spaced grid are displayed in Figure 5.1. Using our algorithm with the same parameter h, we reduce (5.1) from $N = 10^5$ terms to 102 terms centered at a selected *subset of the original data points.* The error of the resulting approximation is displayed in Figure 5.1, where we also show the difference between (5.1) and (5.2). The main point here is that while the standard approach in high dimensions becomes impractical (as it requires a multidimensional grid), our approach proceeds unchanged since the cost of the reduction algorithm is dimension independent.

Remark 3. We selected a much higher accuracy for reduction than the difference between the original and estimated PDFs in order to illustrate the fact that the accuracy limit of Algorithm 1 of about 7 to 8 digits is more than sufficient for this application. Note that to achieve a comparable accuracy for estimation of a PDF via KDE one needs $\approx 10^{16}$ points since the accuracy improves as $\mathcal{O}\left(1/\sqrt{N}\right)$.

Remark 4. Using KDE in high dimensions requires an additional assumption that points are located in a vicinity of a low dimensional manifold (see comments in e.g. [61, Section 1.5.3] and/or Figure 5 in [54]).

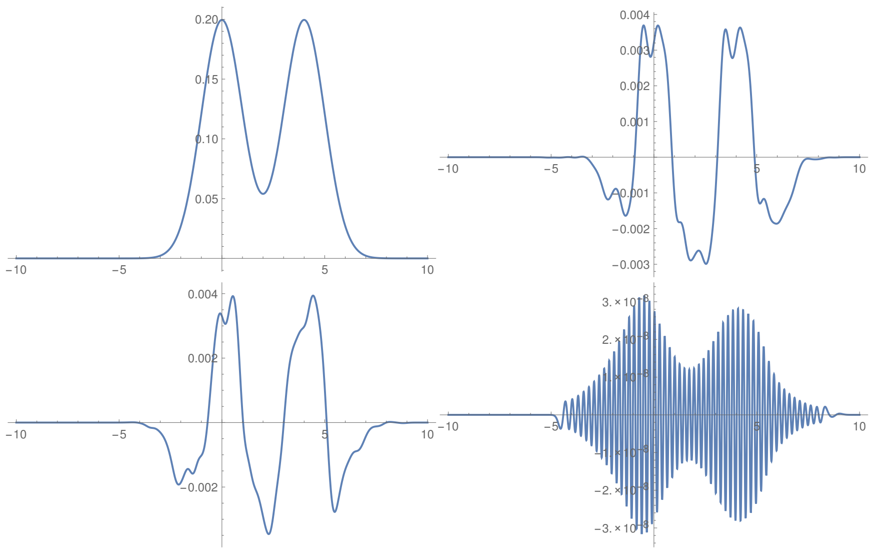

Figure 5.1: The distribution (5.3) (top left) and the error of its estimation with 241 term centered at an equally spaced grid obtain using KDE in Mathematica^{TM} (top right). The error of estimating (5.3) using Algorithm 1 with 102 terms centered at a selected subset of the *original* data points (bottom left). We also show the difference between the definition of the PDF in (5.1) and its reduced version in (5.2) obtained using Algorithm 1 where accuracy was set to 10^{-7} (bottom right).

5.2 An example in high dimensions

We generate $N = 10^5$ samples from a two dimensional Gaussian distributions with a PDF

$$g(\mathbf{y}) = \frac{1}{2}\left(\frac{1}{2\pi}e^{-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{y}-\boldsymbol{\mu}_1)} + \frac{1}{2\pi}e^{-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\mathbf{y}-\boldsymbol{\mu}_2)}\right)$$

where $\boldsymbol{\mu}_1 = (0,0)$, $\boldsymbol{\mu}_2 = (3,3)$ and

$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 2 & 0 \\ 0 & 0.5 \end{pmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

We pad these samples with zeros so that they belong to a d-dimensional space and denote them by $\{\mathbf{y}_i\}_{i=1}^N$. We then apply a random rotation matrix U to obtain the test data $\{\mathbf{x}_i\}_{i=1}^N$. As the bandwidth parameter, we set

$$h = \left(\frac{4}{2d+1}\right)^{\frac{1}{d+4}} N^{-\frac{1}{d+4}},$$

a value that minimizes the mean integrated square error when the underlying distribution is standard normal (see e.g. [63]). In our case, since the intrinsic dimension of the test data is 2, we set $h = 0.14142135623730950488$. The initial kernel density estimator using all test data is

$$f(\mathbf{x}) = \frac{1}{Nh^d}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{d}{2}}}e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T\left(h^2\boldsymbol{I}_d\right)^{-1}(\mathbf{x}-\mathbf{x}_i)}, \tag{5.4}$$

where $\boldsymbol{I}_d$ is the d-by-d identity matrix. We then reduce the number of terms in (5.4) using Algorithm 1 and obtain a sum of Gaussians with fewer terms,

$$\widetilde{f}(\mathbf{x}) = \sum_{j=1}^{\widetilde{N}} c_j \frac{1}{(2\pi)^{\frac{d}{2}}}e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_j)^T\left(h^2\boldsymbol{I}_d\right)^{-1}(\mathbf{x}-\mathbf{x}_j)}.$$

In our experiments, we choose dimension $d = 2, \ldots 16$, and the error threshold $\epsilon = 10^{-1}$ in Algorithm 1. The number of terms after reduction is about 2000 for all $d = 2, \ldots, 16$, and the approximation error is about 1.2×10^{-3} (see Figure 5.2).

In order to compare our approximation with the true PDF $g(\mathbf{x})$, we consider a point $\mathbf{y} = (y_1, y_2, 0, \ldots, 0) \in \mathbb{R}^d$ such that, under the rotation U, we have $\mathbf{x} = U\mathbf{y}$. Evaluating (5.4) at $\mathbf{x}$, we obtain

$$\begin{aligned} f(\mathbf{x}) &= f(U\mathbf{y}) \\ &= \frac{1}{Nh^d}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{d}{2}}}e^{-\frac{1}{2}(U(\mathbf{y}-\mathbf{y}_i))^T\left(h^2\boldsymbol{I}_d\right)^{-1}(U(\mathbf{y}-\mathbf{y}_i))} \\ &= \frac{1}{Nh^d}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{d}{2}}}e^{-\frac{1}{2}(\mathbf{y}-\mathbf{y}_i)^T\left(h^2\boldsymbol{I}_d\right)^{-1}(\mathbf{y}-\mathbf{y}_i)} \\ &= c_d\frac{1}{Nh^2}\sum_{i=1}^{N}\frac{1}{2\pi}e^{-\frac{\left(y_1-y_1^{(i)}\right)^2+\left(y_2-y_2^{(i)}\right)^2}{2h^2}}, \end{aligned}$$

where $c_d = h^{2-d}(2\pi)^{\frac{2-d}{2}}$ and $y_j^{(i)}$ denote the j-th component of $\mathbf{y}_i$. Notice that the kernel density estimator for the PDF of the original distribution in two dimensions is

$$g(\mathbf{y}) \approx \frac{1}{Nh^2}\sum_{i=1}^{N}\frac{1}{2\pi}e^{-\frac{\left(y_1-y_1^{(i)}\right)^2+\left(y_2-y_2^{(i)}\right)^2}{2h^2}} = \frac{1}{c_d}f(\mathbf{x})$$

Therefore, to assure the size of the approximation error, a set of grid points $\mathbf{y}_{ij}$, $i, j = 1, \ldots, 64$, is generated in $[-8, 8] \times [-8, 8]$. We pad $\mathbf{y}_{ij}$ with zeros and apply the same rotation matrix U to get a set of points $\mathbf{x}_{ij}$. In Figure 5.2 (for dimension $d = 16$) we show the PDF g and the errors between g and the KDE estimate f before reduction and $\widetilde{f}$ after reduction as well as the difference between f and $\widetilde{f}$.

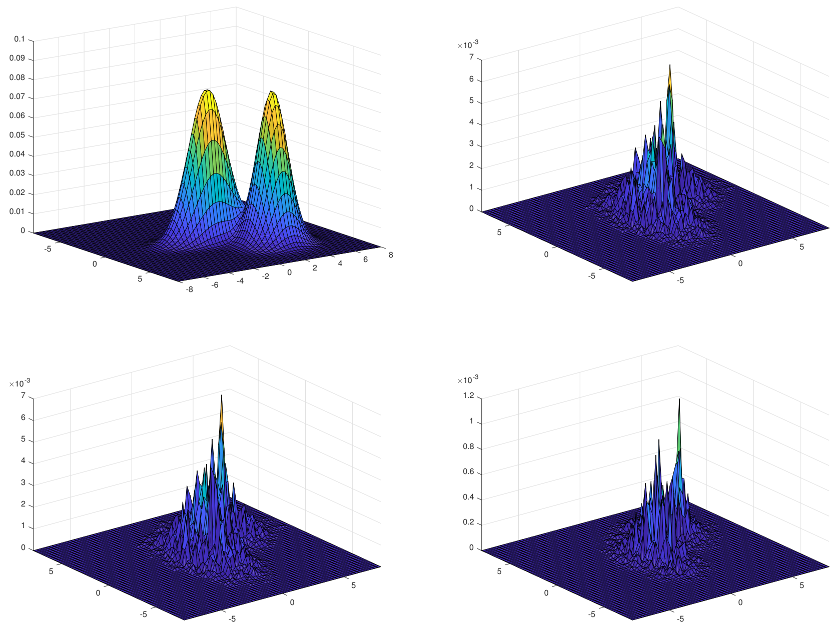

Figure 5.2: The PDF $g\left(\mathbf{y}_{ij}\right)$ (top left) and the difference between g and f (with rescaling by $1/c_d$) $\left|g\left(\mathbf{y}_{ij}\right)-\frac{1}{c_d}f\left(\mathbf{x}_{ij}\right)\right|$ (top right) on the grid $\mathbf{y}_{ij}$, $i,j=1,\ldots 64$. The difference between g and $\widetilde{f}$ (with rescaling by $1/c_d$), i.e., $\left|g\left(\mathbf{y}_{ij}\right)-\frac{1}{c_d}\widetilde{f}\left(\mathbf{x}_{ij}\right)\right|$ (bottom left) and the difference between f and $\widetilde{f}$ (both with rescaling by $1/c_d$) , i.e., $\frac{1}{c_d}\left|f\left(\mathbf{x}_{ij}\right)-\widetilde{f}\left(\mathbf{x}_{ij}\right)\right|$ (bottom right) on the grid $\mathbf{y}_{ij}$, $i,j=1,\ldots 64$.

Chapter 6

Far-field summation in high dimensions

Given a large set of points in dimension $d \gg 3$ with pairwise interaction via a non-oscillatory kernel, our approach provides a deterministic algorithm for fast summation in the far-field setup (i.e. where two groups of points are separated). Recently a randomized algebraic approach in a similar setup was suggested in [54]. Instead, we use Algorithm 1 as a tool to rapidly evaluate

$$g_m = \sum_{n=1}^{N} f_n K\left(\mathbf{x}_m, \mathbf{y}_n\right), \;\; m = 1, \ldots, M,$$

where M and N are large and $K\left(\mathbf{x}, \mathbf{y}\right)$ is a non-oscillatory kernel with a possible singularity at $\mathbf{x} = \mathbf{y}$ (recall that kernels of mathematical physics typically have this type of singularity).

Let us consider sources $\{(\mathbf{y}_n, f_n)\}_{n=1}^{N}$ and targets $\{(\mathbf{x}_m, g_m)\}_{m=1}^{M}$ occupying two distinct d-dimensional balls. Specifically, we assume that $\|\mathbf{y}_n - \mathbf{y}_c\| \leq r_s$ and $\|\mathbf{x}_m - \mathbf{x}_c\| \leq r_t$, where $\boldsymbol{y}_c$, $\boldsymbol{x}_c$ and r_s, r_t are the centers and the radii of the balls. We also assume that sources and targets are separated, i.e., $r \leq \|\mathbf{x}_m - \mathbf{y}_n\| \leq R$. The separation of sources and targets implies that in the evaluation of the kernel we are never close to a possible singularity of the kernel at $\mathbf{x} = \mathbf{y}$. Separation of sources and targets allows us to use a non-singular approximation of the kernel and reduce the problem to finding the best linearly independent subsets of sources (or targets) as defined by such approximate kernel. We also assume that the sources are located on a low-dimensional manifold embedded in high-dimensional space (see Remark 7 below).

In order to use Algorithm 1, we need to define an inner product and make sure that its definition incorporates the assumption of separation of sources and targets. We illustrate this using the example of the Poisson kernel (4.3),

$$K(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^{-d+2}$$

which we use without standard normalization. Approximating $K(\mathbf{x},\mathbf{y})$ in the same manner as shown in Section 4.1 (c.f. (4.4)), we obtain

$$\left| K(\mathbf{x},\mathbf{y}) - \sum_{l\in\mathbb{Z}} w_l e^{-\tau_l \|\mathbf{x}-\mathbf{y}\|^2} \right| \leq \epsilon K(\mathbf{x},\mathbf{y}), \tag{6.1}$$

Since sources and targets are separated, we can further reduce the number of terms in (6.1). In particular, we drop terms with sufficiently large exponents that produce a negligible contribution in the interval $[r,R]$ as well as replace terms with small exponents using algorithm described in [16]. As a result, we define

$$\widetilde{K}(\mathbf{x},\mathbf{y}) = \sum_{l=L_0}^{L_1} w_l e^{-\tau_l \|\mathbf{x}-\mathbf{y}\|^2},$$

$$\left| K(\mathbf{x},\mathbf{y}) - \widetilde{K}(\mathbf{x},\mathbf{y}) \right| < \widetilde{\epsilon}, \text{ for } r \leq \|\boldsymbol{x}-\boldsymbol{y}\| \leq R,$$

where $\widetilde{\epsilon}$ is slightly larger than ϵ. We obtain

$$|g_m - \widetilde{g}_m| \leq \widetilde{\epsilon},$$

where

$$\widetilde{g}_m = \sum_{n=1}^{N} f_n \widetilde{K}(\mathbf{x}_m,\mathbf{y}_n), \quad m = 1,\ldots,M.$$

Since $\|\boldsymbol{x}-\boldsymbol{x}_c\| \leq r_t$ implies $r \leq \|\boldsymbol{x}-\boldsymbol{y}_n\| \leq R$**,** we define the inner product as an integral over the ball $\|\boldsymbol{x}-\boldsymbol{x}_c\| \leq r_t$,

$$\langle \widetilde{K}(\cdot,\mathbf{y}_n), \widetilde{K}(\cdot,\mathbf{y}_{n'})\rangle_d = \int_{\|\boldsymbol{x}-\boldsymbol{x}_c\|\leq r_t} \widetilde{K}(\mathbf{x},\mathbf{y}_n)\,\widetilde{K}(\mathbf{x},\mathbf{y}_{n'})\,d\mathbf{x}. \tag{6.2}$$

The inner product (6.2) can be reduced to a one dimensional integral. In this case, we have

$$\begin{aligned}
&\langle \widetilde{K}(\cdot,\mathbf{y}_n), \widetilde{K}(\cdot,\mathbf{y}_{n'})\rangle_d \\
= &\sum_{l,l'=L_0}^{L_1} w_l w_{l'} \int_{\|\boldsymbol{x}-\boldsymbol{x}_c\|\le r_t} e^{-\tau_l\|\mathbf{x}-\mathbf{y}_n\|^2} e^{-\tau_{l'}\|\mathbf{x}-\mathbf{y}_{n'}\|^2} d\mathbf{x} \\
= &\sum_{l,l'=L_0}^{L_1} w_l w_{l'} e^{-\frac{\tau_l\tau_{l'}}{\tau_l+\tau_{l'}}\|\mathbf{y}_n-\mathbf{y}_{n'}\|^2} \int_{\|\mathbf{x}-\mathbf{x}_c\|\le r_t} e^{-(\tau_l+\tau_{l'})\left\|\mathbf{x}-\frac{\tau_l\mathbf{y}_n+\tau_{l'}\mathbf{y}_{n'}}{\tau_l+\tau_{l'}}\right\|^2} d\mathbf{x} \\
= &\sum_{l,l'=L_0}^{L_1} \widetilde{w}_{ll'}^{nn'} I_{ll'}^{nn'},
\end{aligned}$$

where

$$\widetilde{w}_{ll'}^{nn'} = w_l w_{l'} e^{-\frac{\tau_l\tau_{l'}}{\tau_l+\tau_{l'}}\|\mathbf{y}_n-\mathbf{y}_{n'}\|^2}, \quad \widetilde{\tau}_{ll'} = \tau_l + \tau_{l'}, \quad \widetilde{\mathbf{y}}_{ll'}^{nn'} = \frac{\tau_l\mathbf{y}_n + \tau_{l'}\mathbf{y}_{n'}}{\tau_l+\tau_{l'}},$$

and

$$I_{ll'}^{nn'} = \int_{\|\mathbf{x}-\mathbf{x}_c\|\le r_t} e^{-\widetilde{\tau}_{ll'}\left\|\mathbf{x}-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|^2} d\mathbf{x}.$$

Using spherical coordinates in dimensions $d \ge 3$, we obtain

$$\begin{aligned}
I_{ll'}^{nn'} &= \int_{\|\mathbf{x}-\mathbf{x}_c\|\le r_t} e^{-\widetilde{\tau}_{ll'}\left\|\mathbf{x}-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|^2} d\mathbf{x} \\
&= e^{-\widetilde{\tau}_{ll'}\left\|\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|^2} \int_{\|\mathbf{z}\|\le r_t} e^{-\widetilde{\tau}_{ll'}\|\mathbf{z}\|^2} e^{-2\widetilde{\tau}_{ll'}\left\langle \mathbf{z},\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\rangle} d\mathbf{z} \\
&= e^{-\widetilde{\tau}_{ll'}\left\|\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|^2} \Omega_{d-2} \int_0^{r_t} \left(\int_0^{\pi} e^{-2\widetilde{\tau}_{ll'} r\left\|\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|\cos\theta} \sin^{d-2}\theta d\theta \right) e^{-\widetilde{\tau}_{ll'}r^2} r^{d-1} dr \\
&= e^{-\widetilde{\tau}_{ll'}\left\|\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|^2} \frac{2\pi^{\frac{d}{2}}}{\left(\widetilde{\tau}_{ll'}\left\|\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\|\right)^{\frac{d-2}{2}}} \int_0^{r_t} I_{\frac{d-2}{2}}\left(2\widetilde{\tau}_{ll'}\left\|\mathbf{x}_c-\widetilde{\mathbf{y}}_{ll'}^{nn'}\right\| r\right) e^{-\widetilde{\tau}_{ll'}r^2} r^{\frac{d}{2}} dr, \quad (6.3)
\end{aligned}$$

where Ω_d is the surface area of the d-dimensional sphere embedded in $(d+1)$-dimensional space, i.e.,

$$\Omega_d = \frac{2\pi^{\frac{d+1}{2}}}{\Gamma\left(\frac{d+1}{2}\right)}$$

and I_d is the d-th order modified Bessel function of the first kind (see [1, Eq. 9.6.18]). While in odd dimensions d the integrand in (6.3) can be extended from $[0, r_t]$ to $[-r_t, r_t]$ as a smooth function (in order to use the trapezoidal rule), such extension is not available in even dimensions. Because of this, we choose to use quadratures on $[0, r_t]$ developed in [14] for any d (alternatively, one can use quadratures from [4]).

Remark 5. It is an important observation that the selection of the inner product for finding a linearly independent subset of functions is not limited to the standard one defined in (6.2). Observing that (6.2) approaches to zero as d increases, in all of our experiments in dimensions $d = 3, \ldots 128$, we use (6.2) where we set $d = 3$. Thus, the inner product $\langle \cdot, \cdot \rangle_3$ no longer corresponds to the integral between the functions $\widetilde{K}(\mathbf{x}, \mathbf{y}_n)$ and $\widetilde{K}(\mathbf{x}, \mathbf{y}_{n'})$. However, since we use inner products only to identify the best linearly independent subset of sources (skeleton sources) and compute the coefficients to replace the rest of the terms as linear combinations of these skeleton sources, there are many choices of inner products that will produce similar results.

Associating with sources functions $\{K(\mathbf{x}, \mathbf{y}_n)\}_{n=1}^{N}$, we use Algorithm 1 to find the skeleton terms (i.e. the skeleton sources) with indices $\widehat{I} = \{n_k\}_{k=1}^{r_s}$ which allows us to express the rest of the source functions as

$$\left| \widetilde{K}(\mathbf{x}, \mathbf{y}_n) - \sum_{k=1}^{r_s} \widetilde{f}_{n_k} \widetilde{K}(\mathbf{x}, \mathbf{y}_{n_k}) \right| \leq \epsilon_1, \quad n \notin \widehat{I}.$$

Remark 6. Using Algorithm 1 to find the skeleton sources requires $\mathcal{O}\left(r_s^2 N\right)$ operations and computing interactions between skeleton sources and targets requires $\mathcal{O}\left(r_s M\right)$ operations. Clearly, instead of working with sources, we can work with targets. If targets are located on a low-dimensional manifold, we can associate the functions $\{K(\mathbf{x}_m, \mathbf{y})\}_{m=1}^{M}$ with targets and use Algorithm 1 to find the skeleton targets. In such case, the computational cost becomes $\mathcal{O}\left(r_t^2 M + r_t N\right)$, where r_t is the number of skeleton targets.

Remark 7. If sources are chosen from a random distribution in $\mathbb{R}^d$ rather than located in a small neighborhood of a low-dimensional manifold, the expected distance between two sources $\|\mathbf{y}_n - \mathbf{y}_{n'}\|$ becomes increasingly large as the dimension d increases (see e.g. comments in [61, Section 1.5.3] and examples in [54]). As a result, the functions of variable $\mathbf{x}$, $\widetilde{K}(\mathbf{x}, \mathbf{y}_n)$ and $\widetilde{K}(\mathbf{x}, \mathbf{y}_{n'})$, are effectively linearly independent as d becomes large so that in order to have compressibility, the sources must have a low intrinsic dimension. Therefore, the assumption that sources are located in a small neighborhood of a low-dimensional manifold is not specific to our approach.

6.1 Skeleton sources

We illustrate our approach using sources located on a two-dimensional manifold embedded in a high-dimensional space. For our example, we generate points $\{\mathbf{y}_n\}_{n=1}^N$, $\mathbf{y}_n \in \mathbb{R}^d$ so that the first two coordinates are random variables drawn from the two-dimensional standard normal distribution and the remaining coordinates are set to zero. Next we apply a random rotation and rescale the points so that $\|\mathbf{y}_n\| \leq 1$ for all $n = 1, \ldots, N$. For targets, we draw points $\{\mathbf{x}_m\}_{m=1}^M$ from the d-dimensional standard normal distribution and rescale them so that $\|\mathbf{x}_m\| \leq 1$. We then shift the first component of $\{\mathbf{y}_n\}_{n=1}^N$ by 2 and that of $\{\mathbf{x}_m\}_{m=1}^M$ by -2 so that sources and targets are well separated. Finally, we select the coefficients of sources, $\{f_n\}_{n=1}^N$, from the uniform distribution $\mathcal{U}(0,1)$. In all tests we set $N = 10^4$ and $M = 10^3$. In Table 6.1 we report the actual minimal and maximum distances between sources and targets (dist$_{near}$ and dist$_{far}$), the number of skeleton sources r_s, and the relative error of the approximation,

$$err = \frac{\max_{m=1,\ldots,M} |g_m - \widetilde{g}_m|}{\max_{m=1,\ldots,M} |g_m|}, \tag{6.4}$$

for selected dimensions $3 \leq d \leq 128$. For dimensions $d = 3, 4, 5$, we use the Poisson kernel $\|\mathbf{x} - \mathbf{y}\|^{-d+2}$ while for $d \geq 8$, we use the kernel $\|\mathbf{x} - \mathbf{y}\|^{-1}$ (the fast decay of $\|\mathbf{x} - \mathbf{y}\|^{-d+2}$ results in a negligible interaction between sources and targets in our setup).

d	dist$_{near}$	dist$_{far}$	r_s	error
3	2.5310	5.6317	22	$2.6573e-07$
4	2.8922	5.3544	27	$1.5657e-07$
5	3.2019	5.0864	29	$3.0440e-08$
8	3.1261	5.1186	23	$3.2212e-08$
16	3.1881	5.1049	25	$2.5774e-08$
32	3.5261	5.0242	25	$1.1001e-08$
64	3.6577	4.7520	24	$1.4346e-09$
128	3.8410	4.4825	25	$1.7492e-09$

Table 6.1: Skeleton sources selected using the inner product in (6.2) in dimension d. We report the actual minimal and maximum distances between sources and targets (dist$_{near}$ and dist$_{far}$), the number of skeleton sources r_s, and the relative error of the approximation.

6.2 Equivalent sources

In this example, we consider a similar setting as in Example 6.1 for $d = 2, 3$. We want to replace true sources $\{\mathbf{y}_n\}_{n=1}^N$ located inside a ball by equivalent sources on its boundary so that we reproduce their interaction with the targets within a selected accuracy. We expect the number of equivalent sources on the boundary to be significantly smaller than the number of original true sources so that pairwise interactions with targets can be computed rapidly. We note that such strategy is used in many numerical algorithms (see e.g. [70]) and here we demonstrate that our reduction algorithm can solve this problem.

We combine an initial set of candidate equivalent sources (note that their number will be reduced by the procedure) with the true sources and compute the Cholesky decomposition of their Gram matrix. We use Algorithm 1 with the inner product defined in (6.2) and modify the pivoting strategy to first pivot only among the candidate equivalent sources until we run out of significant pivots; only then we switch to pivot among the true sources. Finally, we compute new coefficients in the usual way (see Algorithm 1) noting that, initially, the candidate equivalent sources had zero coefficients. This approach allows us to (i) obtain the minimal number of equivalent sources and (ii) remove as many of the true sources as possible (we do not preclude the possibility of some of the true sources to remain).

To examine the performance of our approach, we draw source and target points from the d-dimensional standard normal distribution (where $d = 2, 3$), rescale and translate these points so that targets are located in a ball of radius 1 centered at $\mathbf{x}_c$ and sources are located in a ball of radius 0.9 centered at $\mathbf{y}_c$. We choose

$$\mathbf{x}_c = (-2, 0)\,, \mathbf{y}_c = (2, 0)\,, \text{for } d = 2$$

and

$$\mathbf{x}_c = (-2, 0, 0)\,, \mathbf{y}_c = (2, 0, 0)\,, \text{for } d = 3$$

to make sure sources and targets are well separated. Next we pick locations for the candidate equivalent sources on the surface of the ball of radius 1 centered at $\mathbf{y}_c$. In dimension $d = 2$, we pick

$$\mathbf{z}_k = \mathbf{y}_c + (\cos\theta_k, \sin\theta_k), \quad k = 1, \ldots, K,$$

where the angles θ_k are equally spaced on $[0, 2\pi]$ with step size $\frac{2\pi}{K}$. In dimension $d = 3$ we pick

$$\mathbf{z}_{kl} = \mathbf{y}_c + (\cos\theta_k \sin\phi_l, \sin\theta_k \sin\phi_l, \cos\phi_l), \quad k = 1, \ldots, K, \ l = 1, \ldots, L,$$

where the angles θ_k are equally spaced on $[0, 2\pi]$ with step size $\frac{2\pi}{K}$ and the angles ϕ_l are the Gauss-Legendre nodes on $[0, \pi]$. In our experiments we choose a relatively small number of true sources and targets ($N, M = 1000$) so that the result can be clearly visualized (see Figure 6.1 and 6.2). Note that the number of sources can be significantly higher since the algorithm is linear in this parameter. We demonstrate the results in Figure 6.1 and 6.2, where we display the original sources and targets, indicate both, candidate equivalent sources and selected equivalent sources obtained by Algorithm 1.

6.3 Partitioning of points into groups

Algorithm 1 can be used to subdivide scattered points into groups. Indeed, if a set of points (seeds) are specified beforehand then, like in Voronoi decomposition, all points can be split into groups by their proximity to the seeds, i.e. a point belongs to a group associated with a given seed if it is the closest to it among all seeds. The question then becomes how to choose such seeds. There are several algorithms, e.g. Lloyd's algorithm [52], that select such seeds, usually by an iterative procedure to optimize some properties of sought subdivision. We would like to point out that Algorithm 1 can be used to generate initial seeds using linear dependence (which is a proxy for distances between points).

Specifically, let us associate with a point a Gaussian centered at that point. The scale parameter of the Gaussians can be selected sufficiently large (so that the Gaussian is sufficiently flat) to cover the whole set of points. We can then use Algorithm 1 to select the seeds. Since the first term that Algorithm 1 selects is arbitrary, we introduce an additional point as a mean of all points,

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i,$$

and associate an additional Gaussian with this point to start Algorithm 1.

The seeds are the first significant pivots produced by the algorithm and our choice of their number depends on the goals of the subdivision. By its nature, the reduction Algorithm 1 tends to push these seeds far away from each other. We observe that groups with a small number of points appear to contain outliers (see e.g. Figure 6.3), so that the resulting subdivision can be helpful in identifying them. Since the computational cost of Algorithm 1 is $\mathcal{O}\left(r^2N\right)$, where N is the original number of points and r is the number of seeds, as long as the number of groups we are seeking is small, this algorithm is essentially linear. We note that we can subdivide the resulting groups further and, in a hierarchical fashion, build a tree structure. In this paper we simply illustrate the use of Algorithm 1 for subdivision of points into groups and plan to develop applications of this approach elsewhere.

For the example in Figure 6.3 we use the two dimensional distribution of points described in Section 5.2. We choose the bandwidth parameter $h = 200$ when selecting 4 seeds and $h = 16$ when selecting 10 seeds in order to obtain the corresponding subdivisions of the set. Observe that outliers tend to be associated with linearly independent terms and, thus, form a group with a small number of points. Using Algorithm 1 to subdivide scattered points into groups requires further analysis and we plan to address it elsewhere.

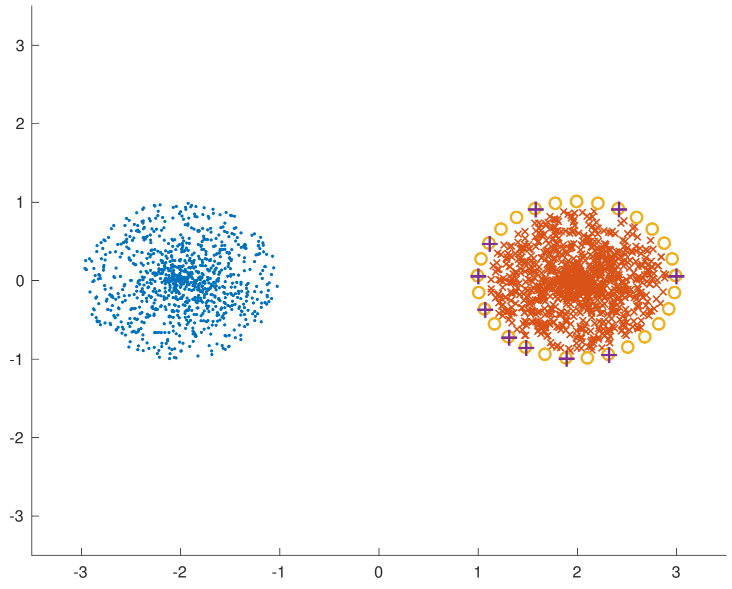

Figure 6.1: Example in dimension $d = 2$. We display $M = 1000$ targets (marked with a dot on the left), $N = 1000$ sources (marked with an "x" on the right), and $K = 30$ candidate sources (marked with a circle on the right). Algorithm 1 selects 10 equivalent sources from the 30 candidate sources (marked with a $+$). The relative approximation error in (6.4) is $1.3e - 07$.

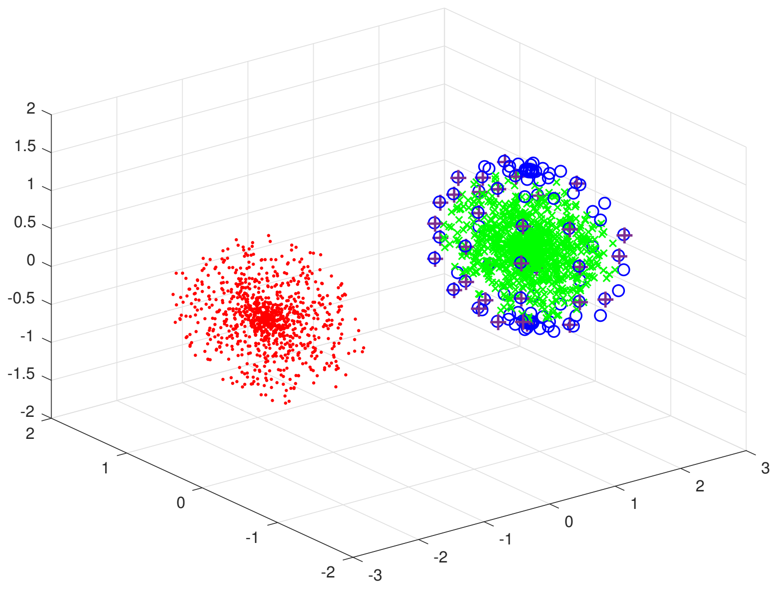

Figure 6.2: Example in dimension $d = 3$. We display $M = 1000$ targets (marked with a dot on the left), $N = 1000$ sources (marked with an "x" on the right), and $K \times L = 10 \times 10$ candidate sources (marked with a circle on the right). Algorithm 1 selects 35 equivalent sources from the 100 candidate sources (marked with a +). The relative approximation error in (6.4) is $8.7e - 08$.

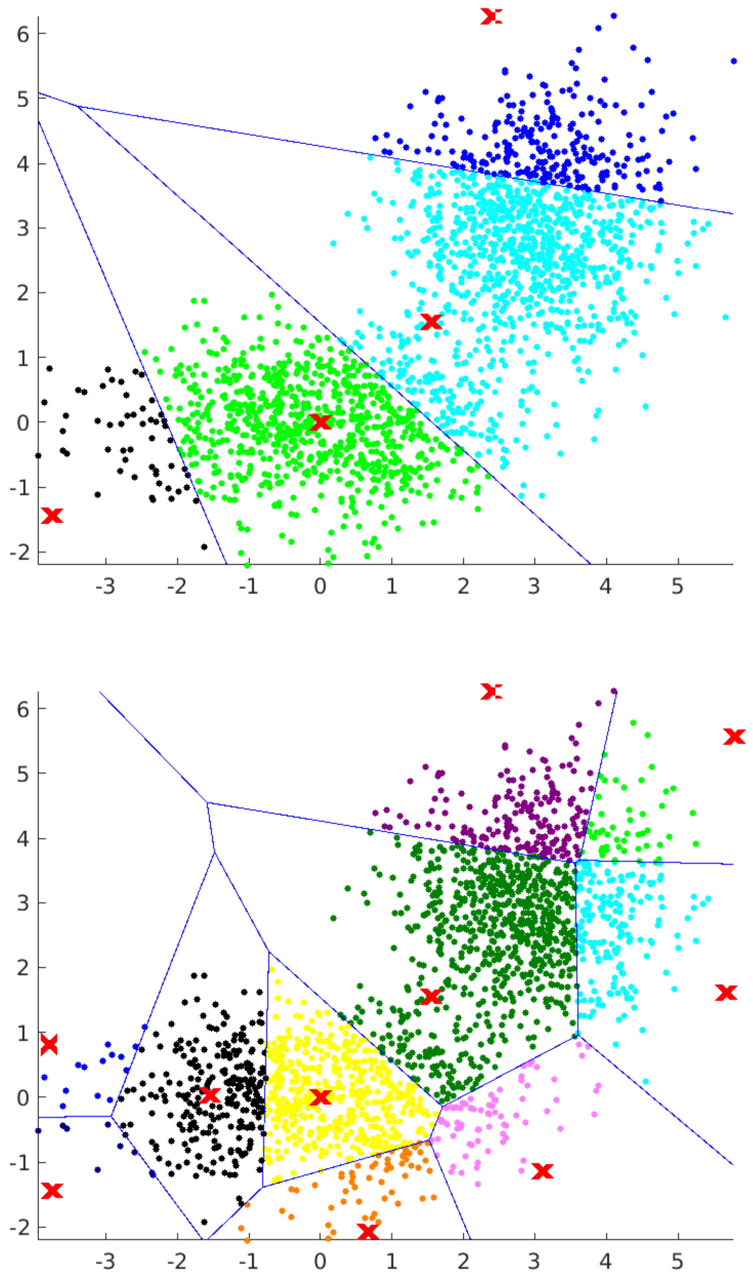

Figure 6.3: Subdivision of 2, 000 points into groups using seeds (marked by "x") produced by Algorithm 1. Illustrated are subdivisions into four groups (top) and into ten groups (bottom). Note that groups with a small number of points are likely to contain outliers.

Chapter 7

Conclusions and further work

In this thesis, we presented a fast algorithm for reducing the number of terms in non-separated multivariate mixtures. The algorithm is based on a partial, pivoted Cholesky decomposition of the Gram matrix. The resulting accuracy is limited to about one half of the available significant digits due to ill-conditioning of the Gram matrix. However, this algorithm is advantageous in high dimensions since it the requires $\mathcal{O}\left(r^2 N\right)$ operations, where N is the initial number of terms in the multivariate mixture and r is the number of selected linearly independent terms. We also consider two additional reduction algorithms for the same purpose. The first algorithm is based on orthogonalization of the multivariate mixture and have a similar performance as the approach based on Cholesky factorization. The second algorithm yields a better accuracy, but currently in high dimensions is only applicable to multivariate mixtures in a separated representation.

We proposed a novel adaptive numerical algorithm for solving partial differential and integral equations from quantum chemistry. The key features of this algorithm are: i) it uses a so-called Basis Generating Iteration to generate necessary basis functions; ii) the excessive number of basis functions are then reduced by the reduction algorithm. We demonstrated the performance of this approach by solving the Hartree-Fock equations in two cases of small molecules. In both examples, the cusp behaviors of the solutions are well resolved and energies are obtained with good accuracy.

The reduction algorithm also allows us to work with non-separated multivariate mixtures that are a far reaching generalization of multivariate separated representations [10, 11, 8] and can be used as a tool for solving multi-dimensional problems. In particular, we solve the Poisson's equation and a linear second order elliptic equation with a variable coefficients. We also address several problems in statistics and data science. We consider the kernel density estimation (KDE) approach for constructing a PDF of a cloud of points, a far-field kernel summation method and the construction of equivalent sources for non-oscillatory kernels (used in both, computational physics and data science) and, finally, show how to use the new algorithm to produce seeds for subdividing a cloud of points into groups. Further work is required to develop new numerical methods that use non-separated multivariate mixtures in applications. We now discuss possible applications and extensions of techniques illustrated in this thesis.

Reduction Algorithms

The accuracy of our reduction algorithm is limited to a half of the available significant digits, while the algorithm using frequency sampling can achieve higher accuracy but its computational cost is expensive in high dimensions. Future work on this algorithm should improve its computational cost. In our implementation, the most costly computation is constructing the sample matrix of the Fourier transform. Ideally, the number of frequency samples should be sufficient to differentiate between all multivariate atoms while remain small in high dimensions. To speed-up the computation, algorithms should use a more efficient (randomized) sample strategy which takes advantage of the geometric structure of the multivariate atoms.

Differential and integral equations with boundary conditions

In Chapter 4, we proposed a new algorithm to solve differential and integral equations in free space. A new Gaussian-type bases using the Jacobi theta functions can be constructed by periodizing Gaussian atoms. Using a linear combination of these function, we can design bases that satisfy periodic, Dirichlet or Neumann boundary conditions in simple domains. A possible future is to solve similar equations in simple domains with boundary conditions.

Electronic structure calculations

In Chapter 3, we demonstrated that our reduction algorithm can be used as a tool for solving Hartree-Fock equations diatomic molecules. There are many possibilities for future work on this topic. Our approach can be competitive with adaptive methods using multiwavelet bases (see [39]). One possible topic is to extend the new adaptive method for more complicated molecules, that use either Gaussians and/or related functions (e.g. products of Gaussians and polynomials) rather than multiwavelets since such representations are more efficient than those used in [39].

Fast kernel summation in general setup

In a general setup of fast kernel summation methods, sources and targets are not separated as it is assumed in the far-field setup. In such case, all existing fast algorithms are based on a hierarchical subdivision of sources and targets. As was demonstrated in Section 6.3, our algorithm can be used to organize scattered points into groups in high dimensions. An future research topic is to extend this approach to construct a fast algorithm in high dimensions that combines the hierarchical subdivision and far-field compression. Another research topic of practical interest is how to extend our approach to the problems with oscillatory kernels (e.g. the Helmholtz kernel).

Kernel density estimation

In practical construction of the KDE, the so-called bandwidth or scale parameter h is commonly chosen through cross-validation on some objective function of interest. This in turn requires a search over many values of h. A possible improvement is to select the bandwidth adaptively so that the construction of the KDE is done in a multiresolution fashion.

Bibliography

[1] Milton Abramowitz and Irene A. Stegun. Handbook of Mathematical Functions, volume 55 of Applied Math Series. National Bureau of Standards, 1964.

[2] B. Alpert. A class of bases in L^2 for the sparse representation of integral operators. SIAM J. Math. Anal, 24(1):246–262, 1993.

[3] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi. Adaptive solution of partial differential equations in multiwavelet bases. J. Comput. Phys., 182(1):149–190, 2002.

[4] Bradley K. Alpert. Hybrid Gauss-trapezoidal quadrature rules. SIAM J. Sci. Comput., 20(5):1551–1584, 1999.

[5] Richard Bellman. Adaptive Control Processes: a Guided Tour. Princeton Univ. Press, Princeton, New Jersey, 1961.

[6] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms, I. Comm. Pure Appl. Math., 44(2):141–183, 1991. Yale Univ. Technical Report YALEU/DCS/RR-696, August 1989.

[7] G. Beylkin, G. Fann, R. J. Harrison, C. Kurcz, and L. Monzón. Multiresolution representation of operators with boundary conditions on simple domains. Appl. Comput. Harmon. Anal., 33:109–139, 2012. http://dx.doi.org/10.1016/j.acha.2011.10.001.

[8] G. Beylkin, J. Garcke, and M. J. Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. SIAM Journal on Scientific Computing, 31(3):1840–1857, 2009.

[9] G. Beylkin and T. S. Haut. Nonlinear approximations for electronic structure calculations. Proc. R. Soc. A, 469(20130231), 2013.

[10] G. Beylkin and M. J. Mohlenkamp. Numerical operator calculus in higher dimensions. Proc. Natl. Acad. Sci. USA, 99(16):10246–10251, August 2002.

[11] G. Beylkin and M. J. Mohlenkamp. Algorithms for numerical analysis in high dimensions. SIAM J. Sci. Comput., 26(6):2133–2159, July 2005.

[12] G. Beylkin, M. J. Mohlenkamp, and F. Pérez. Preliminary results on approximating a wavefunction as an unconstrained sum of Slater determinants. Proceedings in Applied Mathematics and Mechanics, 7(1):1010301–1010302, 2007. Special Issue: Sixth International Congress on Industrial Applied Mathematics (ICIAM07) and GAMM Annual Meeting, Zürich 2007.

[13] G. Beylkin, M. J. Mohlenkamp, and F. Pérez. Approximating a wavefunction as an unconstrained sum of Slater determinants. Journal of Mathematical Physics, 49(3):032107, 2008.

[14] G. Beylkin and L. Monzón. On generalized Gaussian quadratures for exponentials and their applications. Appl. Comput. Harmon. Anal., 12(3):332–373, 2002.

[15] G. Beylkin and L. Monzón. On approximation of functions by exponential sums. Appl. Comput. Harmon. Anal., 19(1):17–48, 2005.

[16] G. Beylkin and L. Monzón. Approximation of functions by exponential sums revisited. Appl. Comput. Harmon. Anal., 28(2):131–149, 2010.

[17] G. Beylkin, L. Monzón, and I. Satkauskas. On computing distributions of products of random variables via Gaussian multiresolution analysis. Submitted to Appl. Comput. Harmon. Anal., 2016. See arXiv:1611.08580.

[18] D. J. Biagioni, D. Beylkin, and G. Beylkin. Randomized interpolative decomposition of separated representations. Journal of Computational Physics, 281:116–134, 2015.

[19] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. The Annals of Statistics, 38(5):2916–2957, 2010.

[20] S. F. Boys. The integral formulae for the variational solution of the molecular many-electron wave equations in terms of gaussian functions with direct electronic correlation. Proceedings of the Royal Society of London Series A- Mathematical and Physical Sciences, 258(1294):402, 1960.

[21] S.F. Boys. Electronic wavefunctions. I. A general method of calculation for stationary states of any molecular system. Proc R. Soc. London Ser. A, 200:542, 1950.

[22] R. Bro. Parafac. Tutorial and Applications. Chemometrics and Intelligent Laboratory Systems, 38(2):149–171, 1997.

[23] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition. Psychometrika, 35:283–320, 1970.

[24] Y. Chen, S. Gottlieb, A. Heryudono, and A. Narayan. A reduced radial basis function method for partial differential equations on irregular domains. Journal of Scientific Computing, 66(1):67–90, 2016.

[25] I. Daubechies. Ten Lectures on Wavelets. CBMS-NSF Series in Applied Mathematics. SIAM, 1992.

[26] S. De Marchi. Geometric greedy and greedy points for {RBF} interpolation. Proceedings of the 9th CMMSE (Gijon, Spain, July 2009), 2:381–392, 2009.

[27] A. Doostan and G. Iaccarino. Breaking the curse of dimensionality for a class of pdes with stochastic inputs. Technical report, Center for Turbulence Research, Stanford University, 2008.

[28] A. Doostan and G. Iaccarino. A least-squares approximation of partial differential equations with high-dimensional random inputs. Journal of Computational Physics, 228(12):4332–4345, 2009.

[29] G. Fann, R. Harrison, G. Beylkin, R. Hartman-Baker, J. Jia, W. A. Shelton, and S. Sugiki. MADNESS applied to density functional theory in chemistry and physics. J. of Physics, Conference Series, 78:012018, 2007. doi:10.1088/1742-6596/78/1/012018.

[30] G. Fann, J. Pei, R. J. Harrison, J. Jia, M. Ou, W. Nazarewciz, N. Schunck, and W. A. Shelton. Fast multiresolution methods for density functional theory in nuclear physics. Physics: Conference Series 180, page 012080, 2009.

[31] G. E. Fasshauer. Meshfree approximation methods with MATLAB, volume 6. World Scientific, 2007.

[32] B. Fornberg and N. Flyer. Solving PDEs with radial basis functions. Acta Numer., 24:215–258, 2015.

[33] B. Fornberg, E. Larsson, and N. Flyer. Stable computations with gaussian radial basis functions. SIAM Journal on Scientific Computing, 33(2):869–892, 2011.

[34] Bengt Fornberg and Natasha Flyer. A primer on radial basis functions with applications to the geosciences. SIAM, 2015.

[35] J. Fosso-Tande and R.J. Harrison. Implicit solvation models in a multiresolution multiwavelet basis. Chemical Physics Letters, 561-562:179–184, 2013.

[36] L. Genovese, T. Deutsch, A. Neelov, S. Goedecker, and G. Beylkin. Efficient solution of Poisson's equation with free boundary conditions. J. Chem. Phys., 125(7), 2006.

[37] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2):217–288, 2011.

[38] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. Journal of geophysical research, 76(8):1905–1915, 1971.

[39] R. J. Harrison, G. Beylkin, F. A. Bischoff, J. A. Calvin, G. I. Fann, J. Fosso-Tande, D. Galindo, J.R Hammond, R. Hartman-Baker, J. C Hill, et al. Madness: A multiresolution, adaptive numerical environment for scientific simulation. SIAM J. Sci. Comput., 38(5):S123–S142, 2016. see also arXiv preprint arXiv:1507.01888.

[40] R.J. Harrison, G.I. Fann, T. Yanai, Z. Gan, and G. Beylkin. Multiresolution quantum chemistry: basic theory and initial applications. J. Chem. Phys., 121(23):11587–11598, 2004.

[41] R. A. Harshman. Foundations of the Parafac procedure: model and conditions for an "explanatory" multi-mode factor analysis. Working Papers in Phonetics 16, UCLA, 1970.

[42] YC Hon, R. Schaback, and X. Zhou. An adaptive greedy algorithm for solving large rbf collocation problems. Numerical Algorithms, 32(1):13–25, 2003.

[43] R. A. Horn and C. R. Johnson. Matrix analysis. Cambridge University Press, Cambridge, second edition, 2013.

[44] M. B. Horowitz, A. Damle, and J. W. Burdick. Linear Hamilton Jacobi Bellman equations in high dimensions. In IEEE 53rd Int. Conference on Decision and Control (CDC), pages 5880–5887. IEEE, 2014. http://dx.doi.org/10.1109/CDC.2014.7040310.

[45] S. R. Jensen, S. Saha, J. A. Flores-Livas, W. Huhn, V. Blum, S. Goedecker, and Luca Frediani. The elephant in the room of density functional theory calculations. The Journal of Physical Chemistry Letters, 8(7):1449–1457, 2017.

[46] J. Jia, J. Hill, G. Fann, and R. J. Harrison. Multiresolution fast methods for a periodic 3-d navier-stokes solver. Proceedings of the Eighth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, October 16-19, 2009, Wuhan, Hubei, China, 2009.

[47] B.A. Jones, G. Beylkin, G. H. Born, and R.S. Provence. A multiresolution model for small-body gravity estimation. Celest. Mech. Dyn. Astr., 111:309–335, 2011. http://dx.doi.org/10.1007/s10569-011-9374-y.

[48] M. H. Kalos. Monte Carlo calculations of the ground state of three- and four-body nuclei. Phys. Rev. (2), 128:1791–1795, 1962.

[49] M. H. Kalos. Monte Carlo integration of the Schrödinger equation. Trans. New York Acad. Sci. (2), 26:497–504, 1963/1964.

[50] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. SIAM Review, 51(3):455–500, 2009.

[51] L. Ling. A fast block-greedy algorithm for quasi-optimal meshless trial subspace selection. SIAM Journal on Scientific Computing, 38(2):A1224–A1250, 2016.

[52] S. Lloyd. Least squares quantization in pcm. IEEE transactions on information theory, 28(2):129–137, 1982.

[53] J.V.L. Longstaff and K. Singer. The use of gaussian (exponential quadratic) wave functions in molecular problems. ii. wave functions for the ground state of the hydrogen atom and of hydrogen molecule. Proc. R. Soc. London Ser. A-Math., 258(1294):421, 1960.

[54] W. B. March and G. Biros. Far-field compression for fast kernel summation methods in high dimensions. Applied and Computational Harmonic Analysis, 43(1):39–75, 2017.

[55] V. Maz'ya and G. Schmidt. Approximate approximations, volume 141 of Mathematical Surveys and Monographs. American Mathematical Society, Providence, RI, 2007.

[56] F. A. Pahl and N. C. Handy. Plane waves and radial polynomials: a new mixed basis. Molecular Physics, 100:3199–3224, 2002.

[57] R. G. Parr and W. Yang. Density-Functional Theory of Atoms and Molecules. Number 16 in The International Series of monographs on Chemistry. Oxford Univ. Press, New york, 1989.

[58] M. J. D. Powell. The theory of radial basis function approximation in 1990. University of Cambridge. Department of Applied Mathematics and Theoretical Physics, 1990.

[59] M. Reynolds, A. Doostan, and G. Beylkin. Randomized alternating least squares for canonical tensor decompositions: application to a PDE with random data. SIAM J. Sci. Comput., pages A2634–A2664, 2016.

[60] R. Schaback and H. Wendland. Adaptive greedy techniques for approximate solution of large {RBF} systems. Numerical Algorithms, 24(3):239–254, 2000.

[61] D. W. Scott. Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons, 2015.

[62] T. Shiozaki and S. Hirata. Grid-based numerical hartree-fock solutions of polyatomic molecules. Phys. Rev. A, 76:040503, Oct 2007.

[63] B. W. Silverman. Density estimation for statistics and data analysis, volume 26. CRC press, 1986.

[64] K. Singer. The use of gaussian (exponential quadratic) wave functions in molecular problems. i. General formulae for the evaluation of integrals. Proc. R. Soc. London Ser. A-Math., 258(1294):412, 1960.

[65] G. Tomasi and R. Bro. A comparison of algorithms for fitting the PARAFAC model. Comput. Statist. Data Anal., 50(7):1700–1734, 2006.

[66] N. Vence, R. Harrison, and P. Krstic. Attosecond electron dynamics: A multiresolution approach. Physical Review A, 85(3):033403, 2012.

[67] F. Vico, L. Greengard, and M. Ferrando. Fast convolution with free-space green's functions. Journal of Computational Physics, 323:191–203, 2016.

[68] T. Yanai, G.I. Fann, Z. Gan, R.J. Harrison, and G. Beylkin. Multiresolution quantum chemistry: Analytic derivatives for Hartree-Fock and density functional theory. J. Chem. Phys., 121(7):2866–2876, 2004.

[69] T. Yanai, G.I. Fann, Z. Gan, R.J. Harrison, and G. Beylkin. Multiresolution quantum chemistry: Hartree-Fock exchange. J. Chem. Phys., 121(14):6680–6688, 2004.

[70] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. Journal of Computational Physics, 196(2):591–626, 2004.

Appendix A

Appendix

As mentioned in the paper, computing with multivariate Gaussian mixtures is particularly convenient since all common operations result in explicit integrals. We present below the key identities for multivariate Gaussians using the standard L^1 normalization,

$$N(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\det(2\pi\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

However, when computing integrals with Gaussians atoms, it is convenient to normalize them to have unit L^2-norm.

A.0.0.1 Convolution of two normal distributions

$$\int_{\mathbb{R}^d} N(\mathbf{x}-\mathbf{y}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) N(\mathbf{y}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\, d\mathbf{y} = N(\mathbf{x}, \boldsymbol{\mu}_1+\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2). \tag{A.1}$$

A.0.0.2 Sum of two quadratic forms

Consider vectors $\mathbf{x}$, $\mathbf{a}$, and $\mathbf{b}$ and two symmetric positive definite matrices $\mathbf{A}$ and $\mathbf{B}$. We have

$$(\mathbf{x}-\mathbf{a})^T \mathbf{A}(\mathbf{x}-\mathbf{a}) + (\mathbf{x}-\mathbf{b})^T \mathbf{B}(\mathbf{x}-\mathbf{b}) = (\mathbf{x}-\mathbf{c})^T(\mathbf{A}+\mathbf{B})(\mathbf{x}-\mathbf{c}) + (\mathbf{a}-\mathbf{b})^T \mathbf{C}(\mathbf{a}-\mathbf{b}),$$

where

$$\mathbf{c} = (\mathbf{A}+\mathbf{B})^{-1}(\mathbf{A}\mathbf{a}+\mathbf{B}\mathbf{b})$$

and

$$\mathbf{C} = \mathbf{A}(\mathbf{A}+\mathbf{B})^{-1}\mathbf{B} = \left(\mathbf{A}^{-1}+\mathbf{B}^{-1}\right)^{-1}.$$

A.0.0.3 Product of two normal distributions

We have

$$N\left(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) N\left(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right) = N\left(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right) \cdot N\left(\mathbf{x}, \boldsymbol{\mu}_c, \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}\right) \tag{A.2}$$

where

$$\boldsymbol{\mu}_c = \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1} \left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right).$$

A.0.0.4 Inner product of two normal distributions

It follows that

$$\int_{\mathbb{R}^d} N\left(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) N\left(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right) d\mathbf{x} = N\left(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right). \tag{A.3}$$

Indeed, from (A.1) we have

$$\begin{aligned}
\int_{\mathbb{R}^d} N\left(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) N\left(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right) d\mathbf{x} &= \int_{\mathbb{R}^d} N\left(2\boldsymbol{\mu}_1 - \mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) N\left(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right) d\mathbf{x} \\
&= N\left(2\boldsymbol{\mu}_1, \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right) \\
&= N\left(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right).
\end{aligned}$$

Alternatively, from (A.2) we have

$$\begin{aligned}
\int_{\mathbb{R}^d} N\left(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) N\left(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right) d\mathbf{x} &= N\left(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right) \int_{\mathbb{R}^d} N\left(\mathbf{x}, \boldsymbol{\mu}_c, \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}\right) d\mathbf{x} \\
&= N\left(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right).
\end{aligned}$$