LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
FACULTY OF PHYSICS

New approaches to Numerical Integration in Quantum Chemistry using Quantics Tensor Cross Interpolation

Neue numerische Integrationsansätze in der Quantenchemie unter Verwendung von Quantics Tensor Cross Interpolation

Bachelor's Thesis
Chair of theoretical solid state physics

Author: Pirmin Finkel

Supervisors:
Prof. Dr. Jan von Delft
Markus Frankenbach

Submission Date: September 13, 2024

Acknowledgments

I would like to express my deepest gratitude to Professor Jan von Delft, whose support allowed me to undertake and complete this bachelor's thesis. His guidance and encouragement throughout the process were invaluable.

My sincere thanks also go to my supervisor, Markus Frankenbach, whose insightful comments and feedback consistently steered me in the right direction. His dedication and expertise significantly contributed to the quality of my work. I learned a lot from Markus not only about physics, but also on the practical aspects of scientific work. He explained me the structure of density field theory calculations and encouraged me to learn useful text editors as vim. Further he helped me with technical issues as fixing program bugs, compile packages or run code on the cluster. This project would not have been possible without him.

Further I would like to thank Philip Haupt and Andreea Filip, who helped me to fix issues with the *pyTCHint* code and compile the library properly. Whenever I had questions about the calculations, they responded quickly and reliable and helped me to understand the algorithm in all its details.

Also I would like to acknowledge helpful discussions with Marc Ritter, Anxiang Ge, Nepumuk Ritz, Benedikt Schneider, Marcel Gievers and Mathias Pelz. They constantly gave me feedback after the micro-updates and suggested solutions to technical problems of any kind.

My thanks also go to Hiroshi Shinaoka, who gave me his insights on the problem at the tensor4all conference in Vienna. He encouraged me to investigate small details further and shared his expertise about tensor algorithms.

I am profoundly grateful to my parents for their support and for providing me with the opportunity to pursue my studies in Physics. Their belief in my abilities and their constant encouragement have been my greatest motivation.

Lastly, I would like to thank all those who took the time to review my thesis and offer constructive feedback. Their contributions were essential in refining this work. I want to especially thank Naomi Brandl and Miriam Keim, who took their time to really dive into the theory of tensor networks and helped me find a pedagogical way to present the theory. Also I would like to thank Annika Bewersdorf, who constantly helped me reformulating the thesis and correcting grammar mistakes.

Thank you all for your support and assistance.

Contents

Chapter 1

Introduction

Numerical integration of multivariate functions is a computationally very costly operation and therefore, the bottleneck in calculations in numerous fields of science. Solving Feynman diagrams for electron phonon models [9] in many body field theory or computing the coulomb matrix in quantum chemistry [10] are just two examples among many. Alternative approaches to evaluate integrals numerically thus have many possible applications and allow algorithms, through a speedup in runtime, to investigate parameter spaces that were previously inaccessible.

By constructing a tensor from the integrated function with the quantics representation, one gains access to a class of powerful compression algorithms, which are collectively called 'tensor cross interpolation' (TCI). These algorithms decompose the input tensor into a tensor train (TT). If this tensor train has a low rank structure, further manipulations such as integration or Fourier transformation can easily be performed and TCI delivers an exponential speedup of these operations. The construction is rank revealing, terminating at the latest when the rank of the tensor train reaches the tensor rank. For low rank tensors, an accurate decomposition can thus be found by evaluating only a small number of grid points. For high rank tensors, the algorithm converges slowly against a high rank decomposition instead of returning a bad approximation. Alternatively the maximal bond can be limited, guaranteeing fast results.

The combination of quantics representation and TCI (shortly called QTCI) [6] allows the integration of multivariate functions with tensor tools. Applications are conceivable everywhere where heavy numeric computation is required. The field of possible applications contains physics related topics as denoising in quantum simulations [15] or computational plasma physics [22], but also includes topics of other sciences, such as financial mathematics [16]. This bachelor thesis aims to explore a small part of this huge space of possible applications. We choose two integrals that currently represent bottlenecks in quantum chemistry and investigate whether QTCI could speed up the existing code. This thesis consists of three main chapters:

- First, we want to introduce the quantics representation and the TCI algorithms. Readers already familiar with QTCI may skip this part and directly proceed with chapters 3 and 4. In this introductory chapter, we explain the core concepts of the method and focus on an understanding the algorithm's input and output.

- We investigate the applicability of the QTCI algorithm to density functional theory (DFT) and discuss different approaches to accelerate bottlenecks in the calculation. Further, we will perform a detailed analysis of the integration error made by QTCI and present a method to determine suitable input parameters.
- Last but not least, we transfer the achieved insights to a similar problem: the evaluation of the xcoulomb matrix in transcorrelated methods. We perform the computation with our QTCI approach and investigate if it is possible to save function evaluations.

Chapter 2

The QTCI algorithm

The QTCI algorithm [6] combines the tensor representation of functions through the quantics method with the usage of tensor compression algorithms as TCI. Tensor train algorithms and tools were originally developed in the context of quantum many-body theory. The transfer of these methods to functions, which can be regarded as a data set corresponding to a specific grid, opens the door to new compression techniques that can reduce memory requirements and deliver a new approach to integration. This chapter is meant as an introductory text to the basics of tensor networks for readers yet unfamiliar with this topic. Readers already experienced with the concept of tensor trains and TCI may skip this chapter and go directly to the applications in chapters 3 and 4.

2.1 Input and output of the algorithm

2.1.1 Tensors

TCI algorithms take a tensor as an input and return a tensor train (TT) as an output. A tensor F is a mathematical object, that has $\mathcal{L}$ discrete indices $\sigma_1, \sigma_2, \ldots, \sigma_{\mathcal{L}}$. $\mathcal{L}$ is called the degree of the tensor. Each index σ_ℓ can take values from $1 \leq \sigma_\ell \leq d_\ell$. In general, d_ℓ can vary for different indices, but for this work in most cases, all indices have the same value range, i.e. $d_1 = \cdots = d_{\mathcal{L}} = d$. We use the more compact notation for the tensor by summarising all indices into an indices vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\mathcal{L}})$. The tensor can now be written as:

$$F_{\sigma_1,\ldots,\sigma_{\mathcal{L}}} = F_{\boldsymbol{\sigma}} \ .$$

For each combination of index values, the Tensor returns a value in a given target space (in our case $\mathbb{R}$ or $\mathbb{C}$). A Tensor can therefore be understood as a function

$$\begin{aligned} F : \ & \mathbb{D}_1 \times \cdots \times \mathbb{D}_{\mathcal{L}} \to \mathbb{C} \\ & \sigma \mapsto F_{\boldsymbol{\sigma}} \end{aligned}$$

where $\mathbb{D}_\ell = \{1, \ldots, d_\ell\}$ is the value space of the index σ_ℓ with $\ell \in \{1, \ldots, \mathcal{L}\}$.

A well-known example of a tensor is the Levi-Civita symbol $\epsilon_{\sigma_1\sigma_2\ldots\sigma_n}$ in n dimensions. It is a tensor of degree $\mathcal{L} = n$ and each index can take values $\sigma_\ell \in \{1, \ldots, n\}$ (in this case we have

$d_1 = \ldots = d_{\mathcal{L}} = n$). The tensor elements are defined by:

$$\epsilon_{\sigma_1 \ldots \sigma_n} = \begin{cases} 1 & \text{if } (\sigma_1, \ldots, \sigma_n) \text{ is a even permutation of } (1, 2, \ldots, n) \\ -1 & \text{if } (\sigma_1, \ldots, \sigma_n) \text{ is a uneven permutation of } (1, 2, \ldots, n) \\ 0 & \text{if at least two indices are identical} \end{cases} .$$

Also one can keep in mind that an $n \times m$ matrix is a tensor with degree $\mathcal{L} = 2$ and $d_1 = n$, $d_2 = m$ and vectors in an n-dimensional space are tensors of degree $\mathcal{L} = 1$ and $d_1 = n$.

Furthermore, a tensor can be notated graphically. Figure 2.1 illustrates this concept. We write the tensor as a circle (or rectangle) and add a leg for every index of the tensor. We can also *contract* a tensor by summing over the same indices

$$A_{\sigma_2} = \sum_{\sigma_1} B_{\sigma_1 \sigma_2} C_{\sigma_1} \overset{\text{EC}}{=} B_{\sigma_1 \sigma_2} C_{\sigma_1} \quad . \tag{2.1}$$

We use the Einstein convention (EC) in the rest of this thesis to shorten the notation. Also, it is worth noting that unlike in other fields of physics, covariant and contravariant notation are equivalent and possess no further meaning. Graphically, one can notate the contraction by connecting two legs representing the same index.

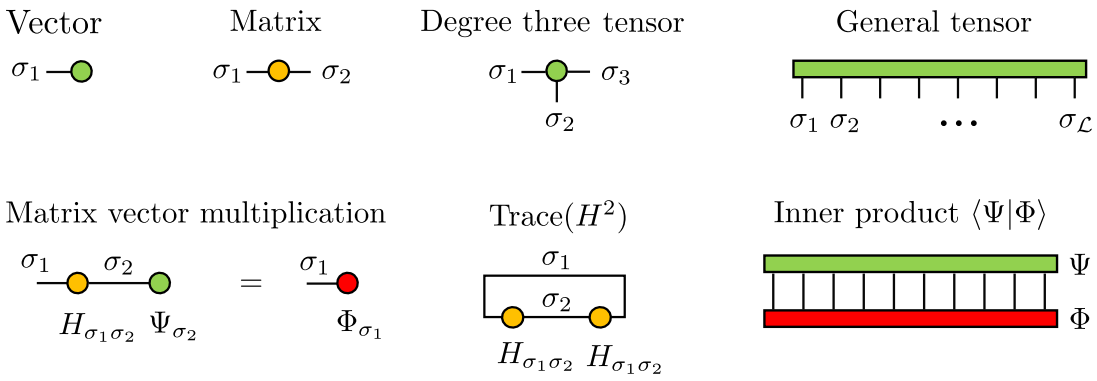


Figure 2.1: Graphical tensor notation. Graphic inspired by [6] figure 1.

2.1.2 Tensor Trains

This subsection is based on [6] pp. 12-13 and delivers a further more pedagogical approach, by adding a concrete example. Any arbitrary tensor $F_{\boldsymbol{\sigma}}$ can be approximated by a tensor train (TT). A TT is a decomposition of the original tensor that consists of $\mathcal{L}$ three-leg tensors (i.e. tensors of degree three), which are connected by contraction of their link bonds. To emphasize that a TT is only an approximation, we notate it with a tilde

$$F_\sigma \approx \widetilde{F}_\sigma = \prod_{\ell=1}^{\mathcal{L}} [M_\ell]^{\sigma_\ell}_{\alpha_{\ell-1}\alpha_\ell} = [M_1]^{\sigma_1}_{1\alpha_1} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \ldots [M_{\mathcal{L}}]^{\sigma_{\mathcal{L}}}_{\alpha_{\mathcal{L}-1}1} \quad . \tag{2.2}$$

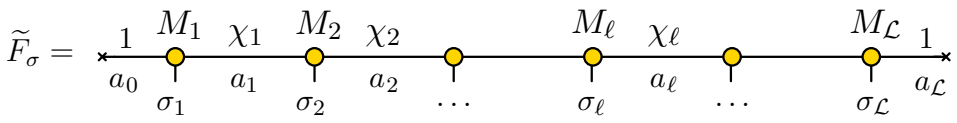


Figure 2.2: Tensor train approximation of a general tensor (from [6] pp. 6).

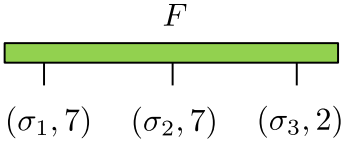
A graphical notation can be seen in figure 2.2. Each three-leg tensor corresponds to one of the original indices σ_ℓ. This external bond keeps its corresponding dimension d_ℓ. The internal bonds α_ℓ are used to connect the tensor train components and have dimension χ_ℓ, which is determined by the algorithm. Since the first and the last link bonds are not contracted, they are just represented by a dummy index of length $\chi_0 = \chi_\mathcal{L} = 1$. We define the rank χ of a TT by the largest bond dimension

$$\chi = \max_{\ell=1\ldots\mathcal{L}-1} \chi_\ell \ . \tag{2.3}$$

We obtain only until $\mathcal{L} - 1$ bond indices, since they connect the $\mathcal{L}$ three leg tensors.

For any input indices vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_\mathcal{L})$, the tensor train can be understood as a chain of matrices that are multiplied. For a fixed σ_ℓ, the three-leg tensor becomes a two-leg tensor, which is a matrix. We want to illustrate this concept with an example:

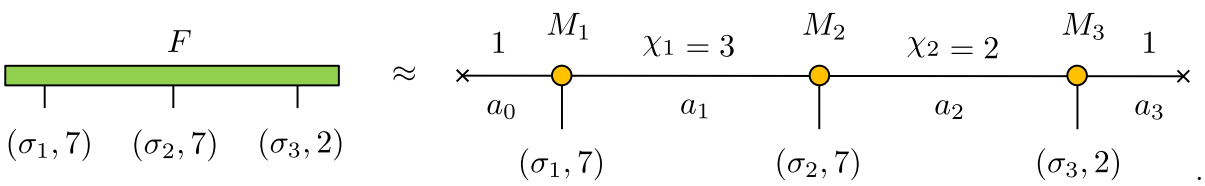
Let us regard the following tensor $F_{\sigma_1\sigma_2\sigma_3}$ of degree three, where $\sigma_1, \sigma_2 \in \{1, \ldots, 7\}$ and $\sigma_3 \in \{1, 2\}$:



$$=$$

$$F[:,:,1] = \begin{bmatrix} 0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0 \\ 1&1&1&1&1&1&1 \\ 0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0 \end{bmatrix} \qquad F[:,:,2] = \begin{bmatrix} 1&1&1&1&1&1&1 \\ 1&0&0&0&0&0&1 \\ 1&0&0&0&0&0&1 \\ 1&0&0&0&0&0&1 \\ 1&0&0&0&0&0&1 \\ 1&0&0&0&0&0&1 \\ 1&1&1&1&1&1&1 \end{bmatrix} .$$

We used the notation (σ_ℓ, d_ℓ) with the explicit numerical value for the dimension of this example. Note that we have $d_1 = d_2 \neq d_3$. Further, the notation ':' means that all rows/columns are shown in the graphic. We represent the tensor here by showing its slices. The tensor has a simple structure, which can be exploited by TCI. The TCI algorithm decomposes the tensor into a TT:



This tensor train would in conclusion, have a rank $\chi = 3$. The three leg tensors are for this example:

$$M_1[1,:,:] = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M_2[:,:,1] = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$M_2[:,:,2] = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$M_3[:,:,1] = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Note that the first leg of M_1 and the last leg of M_3 only have dimension one, since these legs are not contracted. From this tensor decomposition, we can reconstruct the original tensor. We choose an element we want to construct, for example $F[4,4,1] = 1$. By fixing the original indices σ_ℓ but regarding all elements of the bonds a_i, we obtain vectors for the end tensors and a matrix for the center tensor.

$$M_1[:,4,:] = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

$$M_2[:,4,:] = \begin{pmatrix} 1 & -1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix}$$

$$M_3[:,1,:] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The element $F[4,4,1]$ now can be reconstructed by matrix-vector multiplication

$$F[4,4,1] \approx \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1 \tag{2.4}$$

All other elements can be reconstructed in the same way. Note that in equation (2.4) F was only approximated. In this special case, the approximation by TCI is exact (see section 2.2) since the TT and F both have rank three (the maximum of the link dimensions of the TT is three). If the rank of the TT is lower than the tensor rank, then the TT is only an approximation of the original tensor. The error made by the approximation is controlled by the tolerance τ of the compression.

In this small example, the compression by TCI can already be recognised. While the original tensor A needs memory to store $7 \cdot 7 \cdot 2 = 98$ elements, our TT representation only requires $1 \cdot 7 \cdot 3 + 3 \cdot 7 \cdot 2 + 2 \cdot 2 \cdot 1 = 67$ elements. This compression improves dramatically for tensors with a similar rank but index sizes of a few hundred or thousand. Storing the information of such tensors in a compressed TT leads to huge savings in memory and further allows fast tensor operations such as integration or Fourier transformation.

2.2 Tensor Cross Interpolation

Since we have discussed the advantages of learning a tensor train from a given tensor, the question is how to construct it efficiently. This step is achieved by "tensor cross interpolation"

(TCI) algorithms. We want to sketch only the algorithm's key idea since the details and implementation are beyond the scope of this bachelor thesis. It can be regarded as a black box that creates a TT from an input tensor. For further details, we refer to [6].

2.2.1 Matrix cross interpolation

A TCI algorithm achieves the factorisation by generalizing the concept of matrix cross interpolation (MCI) to tensors. The decomposition of a matrix can be done in multiple ways and is the aspect that distinguishes TCI implementations. Originally the decomposition was achieved with cross interpolation (CI)[17]:

$$A \approx CP^{-1}R = \widetilde{A} \ . \tag{2.5}$$

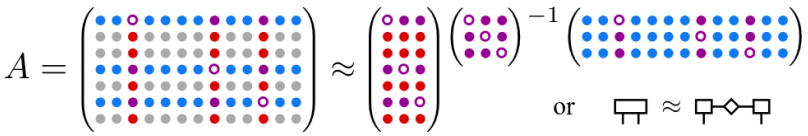


Figure 2.3: Matrix cross interpolation (taken from [14], pp. 2).

C and R consist of columns/rows of A while P is the pivot matrix of the intersecting elements as can be seen in figure 2.3. All colored columns/rows are reproduced by the decomposition, while the grey elements are interpolated. The accuracy of the interpolation can be improved by adding more pivots.

The interpolation can even be made exact $A = \widetilde{A}$, if the matrix A has rank D and D pivots are chosen. A matrix with a low rank thus can be fully reconstructed by just storing a few rows and columns.

The problem with cross interpolation is that it requires the inversion of the pivot matrix, which is for some configurations numerically unstable. To improve this, one can achieve the same results with partial rank revealing LU decomposition (prrLU). The matrix is in this case decomposed into a lower triangular matrix L, a diagonal matrix D and an upper triangular matrix U

$$A \approx LDU = \widetilde{A} \ . \tag{2.6}$$

This method is mathematically equivalent to the CI approximation but circumvents the inversion of the pivot matrix. This results in a more numerically stable algorithm. A proof of the equivalence can be found in [6] section 3.3. For this bachelor thesis, we use the QTCI package, which implements the prrLU decomposition. One should keep in mind that whenever we speak of tensor cross interpolation (TCI), the interpolation itself is achieved by LU decomposition and not cross interpolation as one would assume from the name.

The concept of MCI is generalized to TCI, by regarding a Tensor $F_{\sigma_1 \dots \sigma_{\mathcal{L}}}$ as a matrix F_{ij}, with the multi-index $i = (\sigma_1, \dots, \sigma_{\mathcal{L}-1})$ and $j = \sigma_{\mathcal{L}}$ and iterative application of the shown matrix decomposition. The details of the generalization can be found in [6] chapter 4.

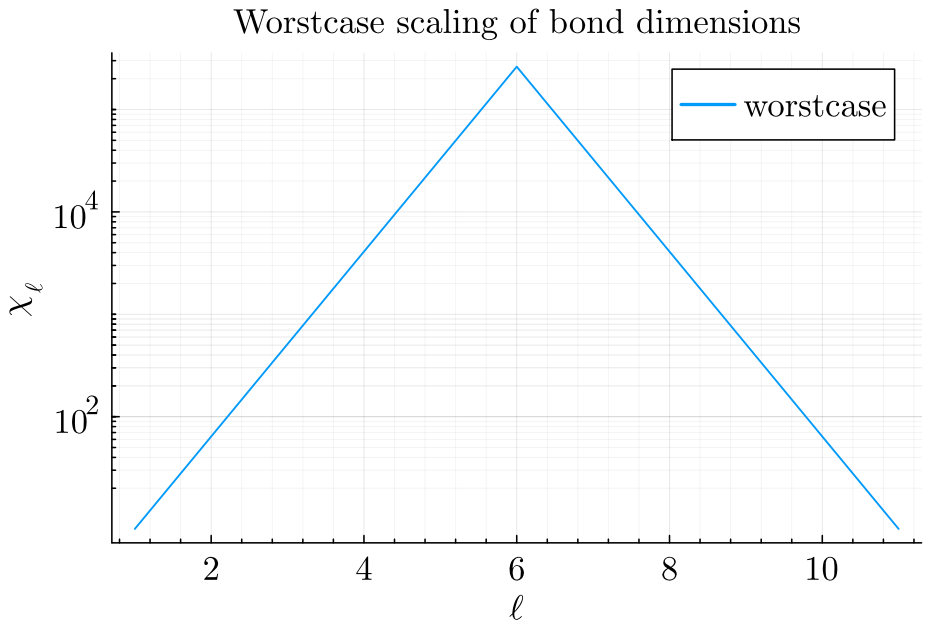


Figure 2.4: Logarithmic plot of worstcase for $\mathcal{R} = 11$

2.2.2 Worstcase scaling of TT rank

The potential of compression for a matrix depends on its rank. For a matrix with high rank, many pivots are needed to approximate the matrix. This statement also holds for the more general tensors. If a tensor has full rank (i.e. a tensor with random elements), many pivots are needed and the bond dimensions χ_ℓ grow exponentially towards the middle. In the worst case of a full rank tensor, this exponential growth does not stop and the dimension of the link bonds is described by :

$$\chi_\ell = \min\left(\prod_{i=1}^{\ell} d_i, \prod_{i=\ell+1}^{\mathcal{L}} d_i\right). \tag{2.7}$$

If all indices have the same length d, this formula reduces to [6] pp. 6

$$\chi_\ell = \min(d^\ell, d^{\mathcal{L}-\ell}) \ . \tag{2.8}$$

In this case, exponential memory cost for storing the TT elements is needed. However, if the tensor has only low rank, which is the case for functions with a non-random structure, then the rank saturates at a certain value. If this value is a lot smaller than the expected exponential worstcase $\chi << 2^{\mathcal{L}/2}$, then we call the tensor strongly compressible since only few elements must be stored to capture its structure.

The exponential growth forms an upper limit to the TT rank. Plotted on a logarithmic scale, the worstcase is a triangle that forms a roof for the TT ranks, as can be seen in figure 2.4.

The comparison of the rank of a function with this worstcase is a parameter of the compressible. A low rank implies that only a few pivots are necessary to capture enough information about a function so that a good interpolation for the rest of the tensor is possible.

2.3 Quantics representation

Every function can be represented as a tensor. Functions can be regarded as a data set, that stores $f(x)$ for every input value x a value. In most practical cases x is part of a set with infinitely many elements. If we limit ourselves to a finite amount of input elements by choosing a suitable grid, we can store the function values in a tensor. To this end, finding a tensor representation of an arbitrary function only means choosing a grid and storing the function values in a clever way. We will demonstrate this by starting with a one-dimensional function.

Assume we have an arbitrary function $f : \mathbb{R} \to \mathbb{C}$ in a continuous variable x. We choose $\mathcal{L}$ arbitrary grid points, and label them with the index $\sigma = \{1, \ldots, \mathcal{L}\}$. Now we have a set that contains a grid point for every index value $\{x(\sigma)\} = \{x(1), x(2), \ldots, x(\mathcal{L})\}$. This leads to f in its *natural representation*, where for each grid index σ we get a value

$$f_\sigma = f(x(\sigma)) \ . \tag{2.9}$$

Since we only chose one index, this process is equivalent to storing function values in a vector (i.e. a degree one tensor).

For a high grid resolution $\mathcal{L}$, a bitwise representation of the grid with multiple indices becomes handy. This method is called the *quantics representation*. We set $\mathcal{L} = 2^{\mathcal{R}}$ where $\mathcal{R}$ is the number of bits we want to use. Each bit has a corresponding index $\sigma_i \in \{0, 1\}$. Instead of using only one index σ as we did in the *natural representation*, we now use $\mathcal{R}$ indices $\sigma_1, \ldots, \sigma_{\mathcal{R}}$.

Let $f : [x_{\min}, x_{\max}] \to \mathbb{C}, x \to f(x)$. We discretize the interval $[x_{\min}, x_{\max}]$ on a cartesian grid with $\mathcal{L} = 2^{\mathcal{R}}$ points. The position of the grid points are defined by

$$x(\sigma_1, \cdots, \sigma_{\mathcal{R}}) = x_{\min} + (x_{\max} - x_{\min}) \cdot \delta(\sigma_1, \cdots, \sigma_{\mathcal{R}}) \ , \tag{2.10}$$

where $\delta(\sigma_1, \cdots, \sigma_{\mathcal{R}}) \in [0, 1)$ can be expressed by $\mathcal{R}$ bits $(\sigma_1, \sigma_2, \ldots, \sigma_{\mathcal{R}})$:

$$\delta(\sigma_1, \cdots, \sigma_{\mathcal{R}}) = \frac{\sigma_1}{2} + \frac{\sigma_2}{2^2} + \cdots + \frac{\sigma_{\mathcal{R}}}{2^{\mathcal{R}}} = \sum_{j=1}^{\mathcal{R}} \frac{\sigma_j}{2^j} \ . \tag{2.11}$$

Every grid point now has a bit representation, and we can use the bit indices $\{\sigma_1, \sigma_2, \ldots, \sigma_{\mathcal{R}}\}$ as legs for our tensor. This leads to the quantics representation of f:

$$f_{\boldsymbol{\sigma}} = f_{\sigma_1 \cdots \sigma_{\mathcal{R}}} = f(x(\sigma_1, \ldots, \sigma_{\mathcal{R}})) \ . \tag{2.12}$$

This concept can easily be generalised on multidimensional functions $f(x_1, x_2, \ldots, x_{\mathcal{N}})$ of $\mathcal{N}$ variables. Every variable x_i is discretized on a grid with $2^{\mathcal{R}}$ points and represented by $\mathcal{R}$ bits $x_i(\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{i\mathcal{R}})$ as in the 1-dimensional case. This leads to $\mathcal{N}\mathcal{R}$ indices in total.

We now start to order the bits by grouping all indices that represent the same grid scale. We start with the bits of all x_i representing the scale 2^{-1}, then the bits of the scale 2^{-2} and so on. The multi-index then can be renamed, resulting in a single index σ_i with $i \in \{1, \ldots, \mathcal{N}\mathcal{R}\}$.

This is called the interleaved representation.

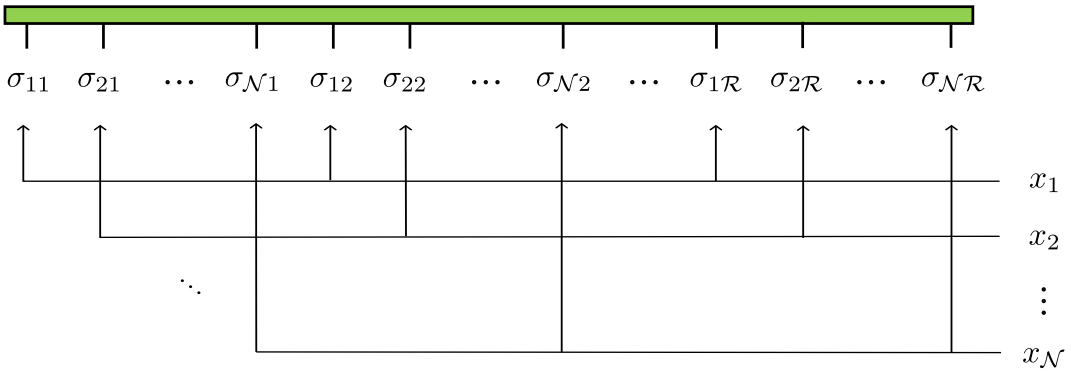


Figure 2.5: Interleaved representation

A more compact approach is given by the fused representation. Starting from the interleaved bit ordering, one can reduce the number of indices by combining all bits of the same scale into one single bit $\widetilde{\sigma}_\ell$

$$\tilde{\sigma}_\ell = \sigma_{\ell 1} + 2 \cdot \sigma_{\ell 2} + \cdots + 2^{\mathcal{N}-1} \cdot \sigma_{\ell \mathcal{N}} = \sum_{n=1}^{\mathcal{N}} 2^{n-1} \sigma_{\ell n} \ . \tag{2.13}$$

This single index can be defined by reversing the bit representation, i.e. $(\sigma_{1\ell}, \ldots, \sigma_{\mathcal{N}\ell})$ are considered as the bit representation of the integer $\widetilde{\sigma}_\ell$. This leads to $\widetilde{\sigma}_\ell \in \{1, \ldots, 2^{\mathcal{N}}\}$ and reduces the number of indices from $\mathcal{R}\mathcal{N}$ to $\mathcal{R}$ but increases the length of the indices from 2 to $2^{\mathcal{N}}$.

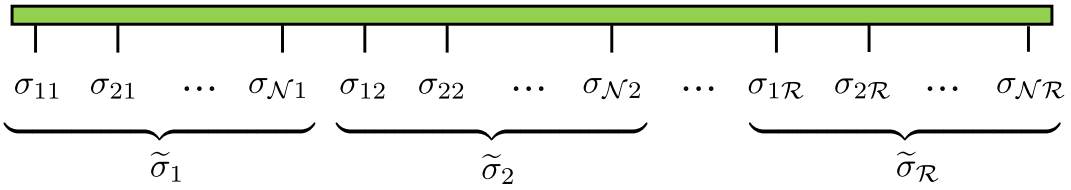


Figure 2.6: Fused representation

As an example for interleaved and fused representation, we regard a function in three dimensions $f : [-1, 1]^3 \to \mathbb{C}$ on a quantics grid with the resolution $\mathcal{R} = 4$. Figure 2.7 shows f in its tensor forms. The grey text comments on the variable and scale each bit corresponds to.

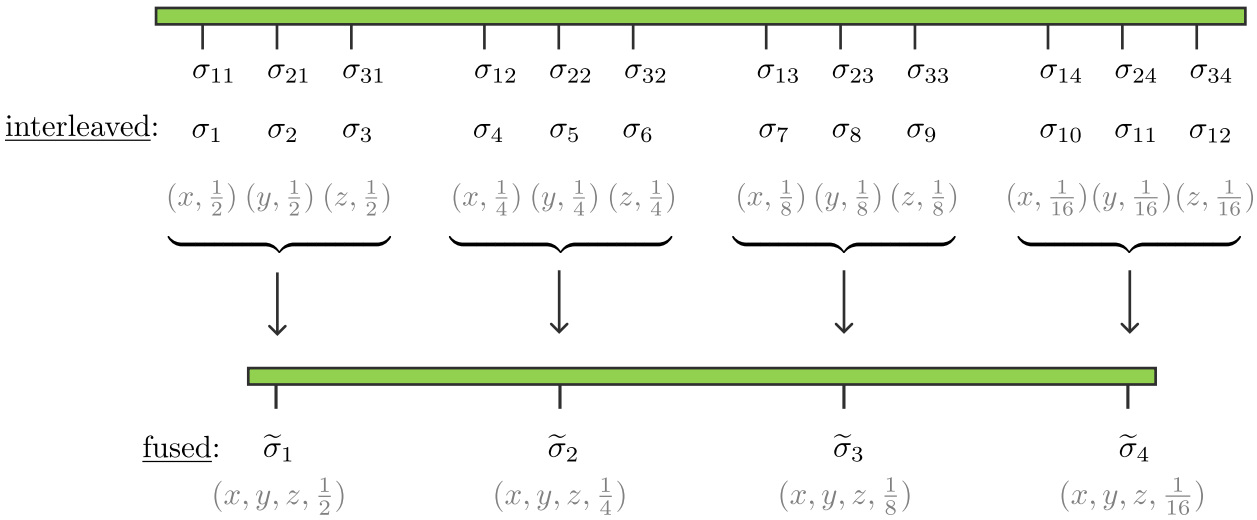


Figure 2.7: Example of fused and interleaved representation of a three dimensional function. All σ_{ij} and σ_i can take values from $\{0, 1\}$, while the $\widetilde{\sigma}_i$ have values in $\{1, \ldots, 8\}$

2.4 Integration in TT representation

Last but not least, we want to show that the integration of a function can simply be achieved by a summation over the constructed tensor train.

Let $f(x_1, \ldots, x_\mathcal{N})$ be a function in $\mathcal{N}$ variables. Assume we already have a tensor train approximation

$$f_{\boldsymbol{\sigma}} \approx \widetilde{f}_{\boldsymbol{\sigma}} = [M_1]^{\sigma_1}_{1\alpha_1} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \ldots [M_\mathcal{R}]^{\sigma_\mathcal{L}}_{\alpha_{\mathcal{L}-1}1} \tag{2.14}$$

of f on a specific grid with $\mathcal{L}$ indices. If we now wish to compute the $\mathcal{N}$-dimensional integral, we can achieve this by quadrature

$$\int d^\mathcal{N}x \;\; f(x_1, \ldots, x_\mathcal{N}) \approx \sum_{\boldsymbol{\sigma}} \omega(\boldsymbol{\sigma}) \widetilde{f}_{\boldsymbol{\sigma}} \;\; , \tag{2.15}$$

where $\sum_{\boldsymbol{\sigma}}$ describes the sum over all index combinations, i.e. all grid points and $\omega(\boldsymbol{\sigma})$ the integration weight at this grid point. The sum is only an approximation of the solution since we sum the function values only over finitely many grid points, and the integrand itself is interpolated by the TT. The errors of these approximations are discussed in more detail in Chapters 3 and 4 at the example of the tested functions.

For a cartesian grid, the integration weights reduce to the cartesian volume element

$$\sum_{\boldsymbol{\sigma}} \omega(\boldsymbol{\sigma}) \widetilde{f}_{\boldsymbol{\sigma}} = \frac{V}{N_\text{grid}} \sum_{\boldsymbol{\sigma}} \widetilde{f}_{\boldsymbol{\sigma}} \;\; , \tag{2.16}$$

where $V = \prod_{n=1}^{\mathcal{N}} (x_{\text{n,max}} - x_{\text{n,min}})$ is the volume of the $\mathcal{N}$-dimensional cube we integrate over. Since we made no constraints on $f_{\boldsymbol{\sigma}}$, we could also compress the grid weights into the function and proceed with the combined $g_{\boldsymbol{\sigma}} = \omega(\boldsymbol{\sigma}) \widetilde{f}_{\boldsymbol{\sigma}}$. To this end, the condition of a cartesian grid

is not necessary but simplifies understanding this derivation.

So far we haven't made use of the tensor structure, since summing over all grid elements is also the concept of the standard approach. But because of the TT structure of $\widetilde{f}_\sigma$, the sum can be factorized:

$$\begin{aligned}\sum_{\boldsymbol{\sigma}} \widetilde{f}_\sigma &= \sum_{\sigma_1 \dots \sigma_{\mathcal{L}}} [M_1]^{\sigma_1}_{1\alpha_1} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \dots [M_{\mathcal{L}}]^{\sigma_{\mathcal{L}}}_{\alpha_{\mathcal{L}-1}1} \\ &= \sum_{\sigma_1} [M_1]^{\sigma_1}_{1\alpha_1} \sum_{\sigma_2} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \cdots \sum_{\sigma_{\mathcal{L}}} [M_{\mathcal{L}}]^{\sigma_{\mathcal{L}}}_{\alpha_{\mathcal{L}-1}1} \\ &= \prod_{\ell=1}^{\mathcal{L}} \left[\sum_{\sigma_\ell=1}^{d} [M_\ell]^{\sigma_\ell}_{\alpha_{\ell-1}\alpha_\ell} \right] .\end{aligned}$$

In conclusion, integration of a tensor train requires only $\mathcal{O}(\mathcal{L}d\chi^2)$ summations and matrix multiplication of the cost $\mathcal{O}(\mathcal{L}\chi^2)$. This method is for TT of low rank χ exponentially cheaper than the integration by quadrature, which requires the summation over the exponentially large grid at a cost $\mathcal{O}(d^{\mathcal{L}})$. For functions with a low rank TT structure, QTCI could bring therefore an exponential speedup.

We also want to mention that we did not make any assumptions about the form of the TT approximation and the number $\mathcal{L}$ of indices. We could have used the presented natural, interleaved, or fused representation, which differ drastically in the parameters $\mathcal{L}$ and d. Even though the integral approximation holds for all imaginable tensor representations, the natural form has a feature that gives the integration approach a new significance. In the natural representation TCI can be understood as an factorization of the integrals. To this end, we will again derive the result, but highlight this time the new aspect.

We represent a function $f(x_1, \dots, x_{\mathcal{N}})$ in its natural tensor form by discretizing each variable x_ℓ on a grid $x(\sigma_\ell) = \{x_\ell(1), \dots, x_\ell(d)\}$, where $\sigma_\ell = 1, \dots d$. For every input vector $\boldsymbol{\sigma}$, a tensor element is now defined by

$$f_{\boldsymbol{\sigma}} = f(x_1(\sigma_1), x_2(\sigma_2), \dots, x_{\mathcal{N}}(\sigma_{\mathcal{N}})) \ . \tag{2.17}$$

In this form, f can be given as input to the TCI algorithm, which returns a TT approximation of f

$$f_{\boldsymbol{\sigma}} \approx [M_1]^{\sigma_1}_{1\alpha_1} [M_2]^{\sigma_2}_{\alpha_1\alpha_2} \dots [M_{\mathcal{N}}]^{\sigma_{\mathcal{N}}}_{\alpha_{\mathcal{N}-1}1} \ . \tag{2.18}$$

One should keep in mind that in the natural representation, every index σ_ℓ is a discrete version of a continuous variable x_ℓ. The discretization can in theory be extended to the continuum by increasing the number of grid points d. To emphasise this one to one correspondence between σ_ℓ and x_ℓ we notate the tensor train leg σ_ℓ as if it were continuous:

$$f(x_1, \dots, x_{\mathcal{N}}) = [M_1]_{1\alpha_1}(x_1) [M_2]_{\alpha_1\alpha_2}(x_2) \dots [M_L]_{\alpha_{\mathcal{N}-1}1}(x_{\mathcal{N}}) \ . \tag{2.19}$$

To this end, TCI can also be regarded as a factorization of the function in its variables. For $\chi = 1$, the function is indeed factorizable, and for a small rank of for example $\chi \approx 30$, we call

the function almost factorizable.

Because of this separation, the $\mathcal{N}$-dimensional integral can be reduced to $\mathcal{N}$ 1-dimensional integrals by inserting the TT approximation for f. The single integrals are multiplied together by tensor contraction (i.e. matrix multiplication, since all M_l are matrices for a fixed x_i or σ_l)

$$\int dx^{\mathcal{N}} f(x_1, \ldots, x_{\mathcal{N}}) = \int dx_1 \left[M_1\right]_{1\alpha_1}(x_1) \cdots \int dx_{\mathcal{N}} \left[M_{\mathcal{L}}\right]_{\alpha_{\mathcal{L}-1}1}(x_{\mathcal{N}}) \ . \tag{2.20}$$

The integrals in each variable are numerically evaluated by a Riemann sum

$$\int dx_i \left[M_i\right]_{\alpha_{i-1}\alpha_i}(x_i) \approx \delta_i \sum_{\sigma_\ell=1}^{d} [M_l]^{\sigma_\ell}_{\alpha_i \alpha_{i+1}} \ , \tag{2.21}$$

with the 1-dimensional cartesian volume element $\delta_i = (x_{\mathrm{i,max}} - x_{\mathrm{i,min}})/d$. Inserting (2.21) in (2.20) yields the already known end result

$$\int dx^{\mathcal{N}} f(x_1, \ldots, x_{\mathcal{N}}) \approx \frac{V}{d^{\mathcal{N}}} \prod_{\ell=1}^{\mathcal{L}} \left[\sum_{\sigma_\ell=1}^{d} [M_\ell]^{\sigma_\ell}_{\alpha_{\ell-1}\alpha_\ell} \right] \ . \tag{2.22}$$

In conclusion, TCI tries to factorize a given input as well as possible and exploits the uncovered structure in the integration step. The reduction of necessary summation steps, and therefore the speed of the integral evaluation, depends heavily on the rank of the TT. The condition for a high acceleration of the computation, compared to standard integration by quadrature, is a low rank structure of the integrand function f. If for specific given functions this low rank tensor structure is available will be investigated in the following chapters.

Chapter 3

Application of TCI in DFT

QTCI is a general tool that can find application wherever heavy numerical (multidimensional) integration is required. One such application is the field of 'Density Functional Theory' (DFT) in quantum chemistry, where the evaluation of lots of overlap integrals is required.

3.1 Density Functional Theory

The goal of density functional theory (DFT) [11] is, to determine the electron density $\rho(\mathbf{r})$ of a given molecule in the ground state. The knowledge of $\rho(\mathbf{r})$ then allows, to calculate observables, like the energy $E[\rho]$, which is a functional of the $\rho(\mathbf{r})$.

The electron density is found in an iterative self consistent field (SCF) calculation. For a multi electron system ρ can be written as a linear combination of N_e occupied molecular orbitals [3] pp. 13-16. These orbitals can be described by single electron wave functions Ψ_ℓ

$$\rho(\mathbf{r}) = \sum_{\ell}^{N_e} |\phi_\ell(\mathbf{r})|^2 \ . \tag{3.1}$$

The orbitals can be expanded over the whole molecule and are themselves described by a linear combination of atomic basis functions

$$\phi_\ell(\mathbf{r}) = \sum_{\mu} C_{\ell\mu}\, \varphi_\mu(\mathbf{r}) \tag{3.2}$$

These basis functions can arbitrarily be chosen. The task of the basis is to approximate the molecule orbitals with as few basis elements as possible. Popular choices for the basis φ_i are discussed later. Finding ρ therefore reduces to finding the the molecular orbital coefficient matrix $\mathbf{C} = \mathbf{C}_{\ell\mu}$.

This can be achieved by solving the Kohn-Sham equation [12] pp. 616-617

$$\mathbf{F}\,\mathbf{C} = \mathbf{S}\,\mathbf{C}\,\epsilon \ , \tag{3.3}$$

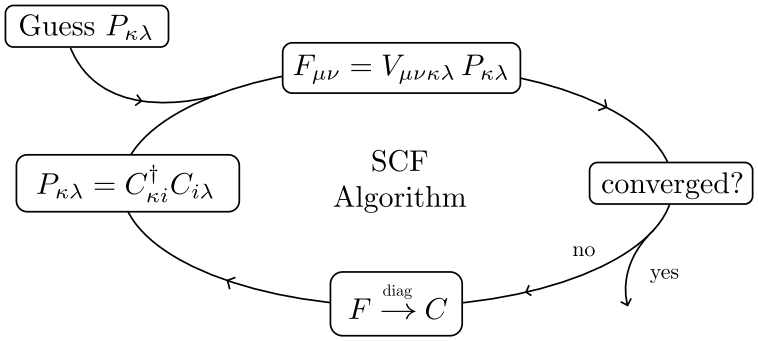


Figure 3.1: Self consistent field algorithm in DFT. Graphic inspired from [12] pp. 616.

with the Kohn-Sham matrix $\mathbf{F} = F_{\mu\nu}$ and the overlap matrix $\mathbf{S} = S_{\mu\nu}$. For a known $\mathbf{F}$ and $\mathbf{S}$ this equation is an eigenvalue problem and can be solved by diagonalisation of $\mathbf{F}$. While $\mathbf{S}$ can be directly computed from the chosen basis

$$S_{\mu\nu} = \int d\mathbf{r}\ \ \varphi_\mu(\mathbf{r}) \cdot \varphi_\nu(\mathbf{r}) \tag{3.4}$$

the Kohn-Sham matrix contains multiple factors, requiring the knowledge of $\mathbf{C}$, among which only one is relevant for us

$$F_{\mu\nu} = V_{\mu\nu\kappa\lambda}\, P_{\kappa\lambda} + \dots \tag{3.5}$$

$$V_{\mu\nu\kappa\lambda} = \iint \frac{\varphi_\mu(\mathbf{r}') \cdot \varphi_\nu(\mathbf{r}') \cdot \varphi_\kappa(\mathbf{r}) \cdot \varphi_\lambda(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|}\ d\mathbf{r}\, d\mathbf{r}' \tag{3.6}$$

$$P_{\kappa\lambda} = \sum_{i \in occ} C^{\dagger}_{\kappa i} C_{i\lambda} \tag{3.7}$$

Summarizing this, to obtain the coefficient matrix $\mathbf{C}$ from the Kohn-Sham matrix, we already need $\mathbf{C}$ to calculate $\mathbf{F}$ in the Kohn-Sham equation. This requires an iterative process, to solve the equations. We start with an inital guess for $\mathbf{C}$ and compute $\mathbf{F}$ with it. We then solve the Kohn-Sham equation and obtain an new coefficient matrix. Then repeat this process until $\mathbf{C}$ and hence ρ does not change anymore. It has reached a the self consistent point. After we found ρ the energy and other observables can be determined for the ground state.

The term we presented in the calculation of $F_{\mu\nu}$ between the is called the coulomb matrix $J_{\mu\nu} = V_{\mu\nu\kappa\lambda}\, P_{\kappa\lambda}$. While all other terms that contribute to $\mathbf{F}$ are easy to compute, the evaluation of the coulomb matrix forms the bottleneck of the algorithm. The problem behind the bottleneck has for distinct bases a different form. Before we explain the nature of the bottleneck, we therefore must have a look on the most common basis.

3.2 The Gaussian basis

For the basis function $\phi_\ell(\mathbf{r})$ one can choose an arbitrary basis. Since one can only implement finitely many basis functions, a basis should be able to approximate the true atomic orbital

function by combining just a few basis elements. The most popular choice are the Gaussian basis sets. They are available in cartesian and spherical form. The spherical sets approximate the radial term of the atomic wave functions by a linear combination of Gauss functions and are then multiplied with the spherical harmonics. In this thesis we use the cartesian ansatz, which has the form:

$$\phi_\ell(\mathbf{r}) = (r_x - R_x)^{l_x}(r_y - R_y)^{l_y}(r_z - R_z)^{l_z} \cdot \sum_{n=0}^{N_{\text{coef}}} c_n e^{-\zeta_n(\mathbf{r}-\mathbf{R})^2} \quad . \tag{3.8}$$

With a basis of this specific form the six dimensional integral of the V-tensor can be evaluated analytically. This reduces the cost of calculating the tensor elements dramatically, since expensive numerical integration can be avoided. For this reason, the gaussian basis is used in almost all molecular DFT programs.

The disadvantage of the gaussian basis is that it approximates the true atomic orbitals rather badly. Cusps at the position of the kernel require a large linear combination of smooth Gauss functions to get well approximated. Higher precision of the calculation requires therefore larger basis sets.

To get more intuition about the gaussian basis, we consider the H_2O basis as an example. This molecule is not a trivial single atom, but has an easy enough structure to present all basis functions in this thesis. Basis sets of different size, and therefore of different resolution are tabulated online, and can easily be implemented. We use the Julia package *GaussianBasis.jl* [1], since it is perfectly compatible for the later use with the QTCI algorithm, which is also based on Julia.

The number of basis functions N_{bas} for the H_2O-molecule depends on the chosen basis set as can be seen in table 3.2. Also the number of expansion coefficients N_{coef} increases for larger bases and allows a better approximation of the atomic orbitals. Each basis function corresponds to a specific atom of the molecule and an angular momentum value (ℓ_x, ℓ_y, ℓ_z). For small sets, most orbitals are only represented by one basis function. In larger sets, orbitals with higher quantum numbers get included, and orbitals may possess multiple functions that approximates them. Hydrogen for example has in the small *sto-3g* basis no p-orbitals and only one s-orbital function, but in the larger *def2-svp* basis also an p-orbital and two different s-orbitals. The basis functions that are implemented for H_2O in the *def2-svp* basis are listed in table 3.1.

Atom	N_{coeff}	ℓ	ℓ_x	ℓ_y	ℓ_z	i
O	5	s	0	0	0	1
	1	s	0	0	0	2
	1	s	0	0	0	3
	3	p	1	0	0	4
			0	1	0	5
			0	0	1	6
	1	p	1	0	0	7
			0	1	0	8
			0	0	1	9
	1	d	2	0	0	10
			1	1	0	11
			1	0	1	12
			0	2	0	13
			0	1	1	14
			0	0	2	15
H	3	s	0	0	0	16
	1	s	0	0	0	17
	1	p	1	0	0	18
			0	1	0	19
			0	0	1	20
H	3	s	0	0	0	21
	1	s	0	0	0	22
	1	p	1	0	0	23
			0	1	0	24
			0	0	1	25

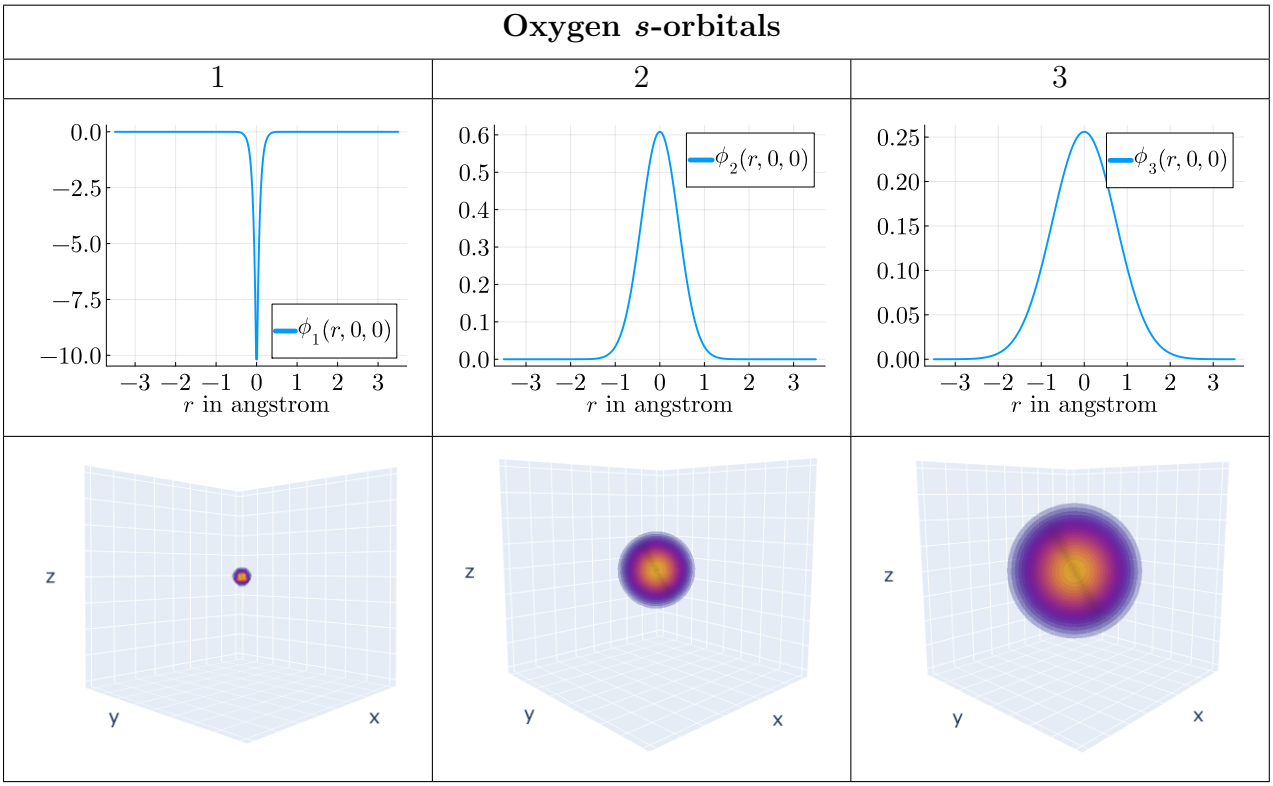
Table 3.1: Basis functions with their corresponding angular momentum of H_2O and number of coefficients N_{coeff} in the *def2-svp* basis. The later use of a function label ϕ_i refers to the index i of this table

Basis Set	N_{bas}
sto-3g	7
def2-svp	25
def2-tzvp	48
def2-qzvp	142

Table 3.2: Number of functions for H_2O in different gaussian basis sets. The used basis sets were developed by [21], [20], [5].

A plot of the *def2-svp* basis can be seen on the following pages. The upper plot shows the function values on a axis. This axis is determined by the angular momentum. We chose the plot axis to point in the direction of $\ell = (\ell_x, \ell_y, \ell_z)$ and normalized it to maintain the correct length units. Functions with $|\ell| = 0$ are spherical symmetric, so we chose for the axis in this case the x-axis. The variable r represents the covered distance on the axis in angstrom. The zero point of this scale is located in the atom kernel. We plotted all function for the same r distance to emphasize the different expansions of the orbitals.

The lower graphic shows the absolute value of the basis function as a density plot in 3D space. Again all functions are plotted on the same volume to emphasize the expansion. We chose a cubic volume $[-2, 2]^3$, centered around the corresponding atom. The density shows all points $\mathbf{r}$, whose function value lie in in the interval $0.2 \cdot f_{\max} \leq f(\mathbf{r}) \leq f_{\max}$, which leads to a representation of the main support of each function.



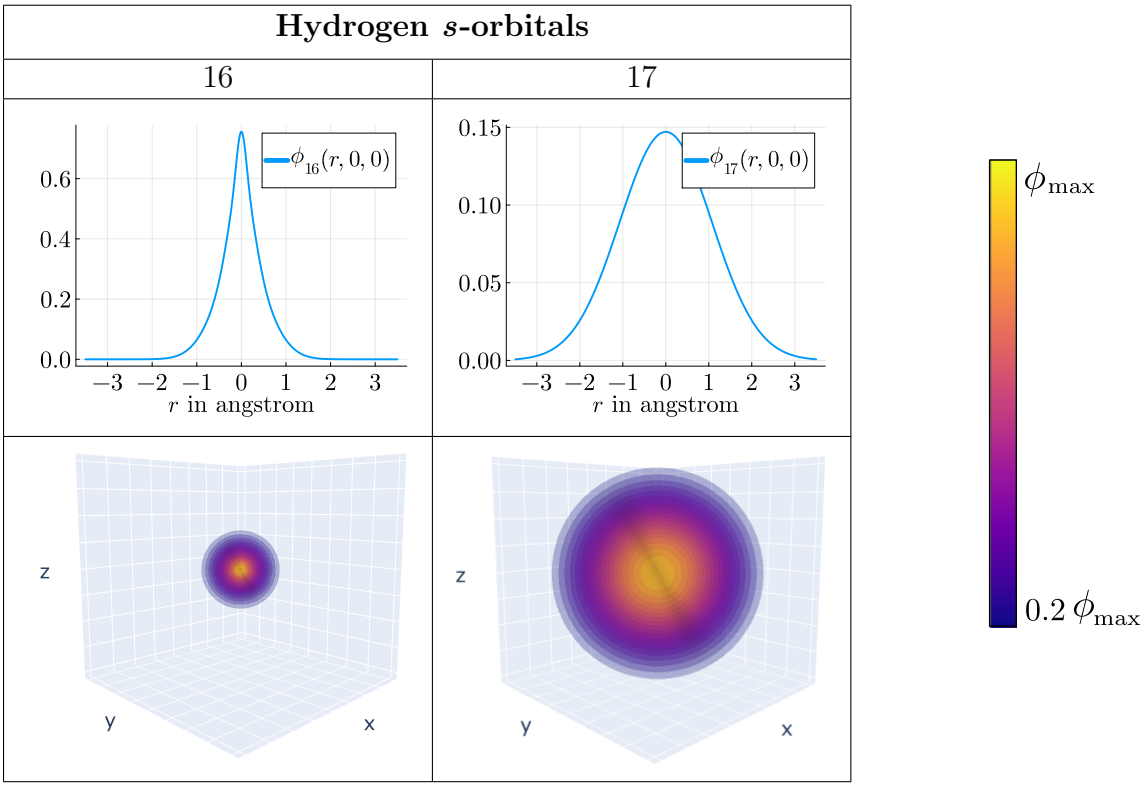


Figure 3.2: Atom centred density and x-axis plots of s-orbitals. The density is plotted on the volume $[-2,2]^3$. The heatscale is used for all 3D plots.

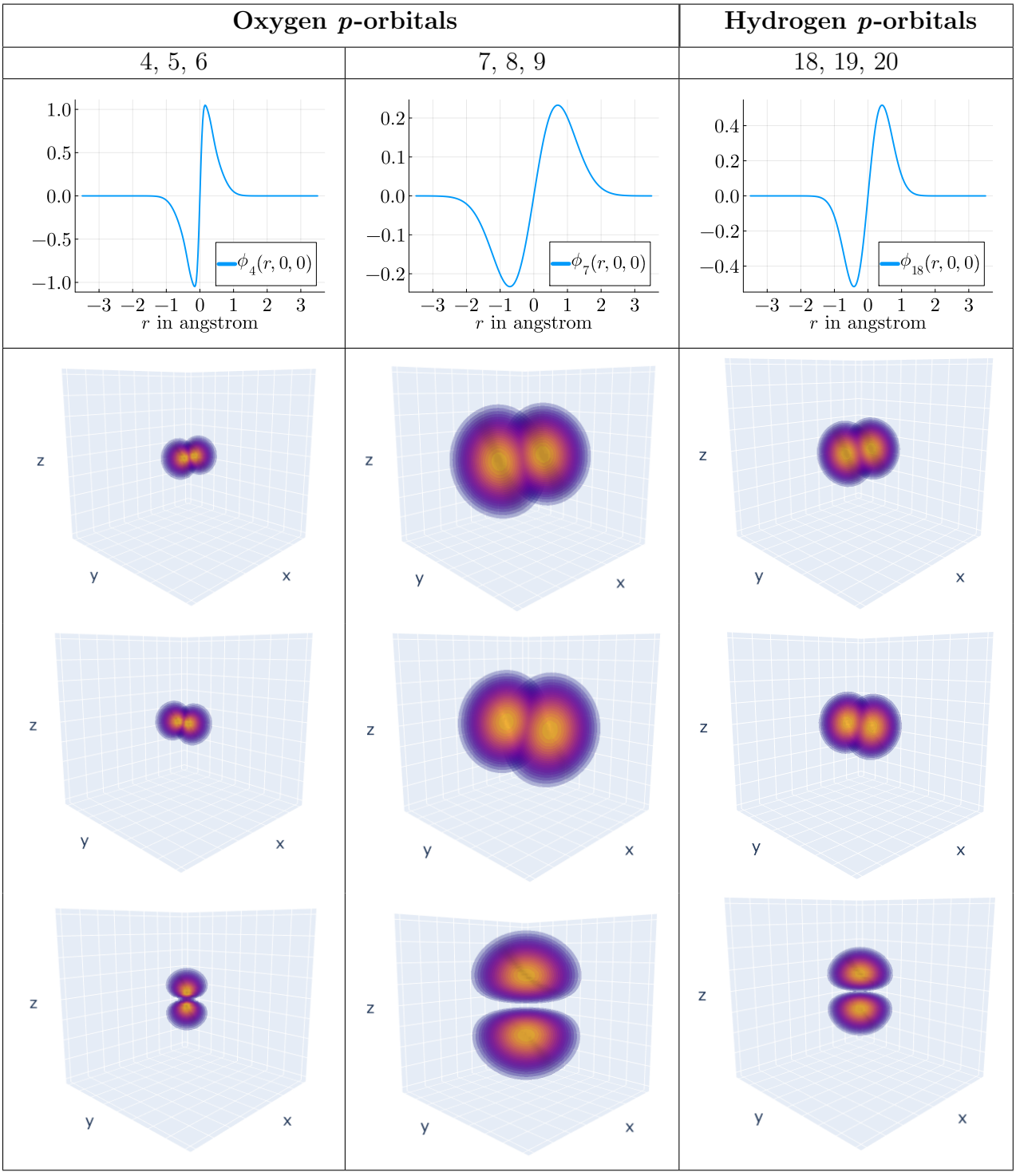


Figure 3.3: Atom centred density and x-axis plots of p-orbitals. The density is plotted on the volume $[-2,2]^3$.

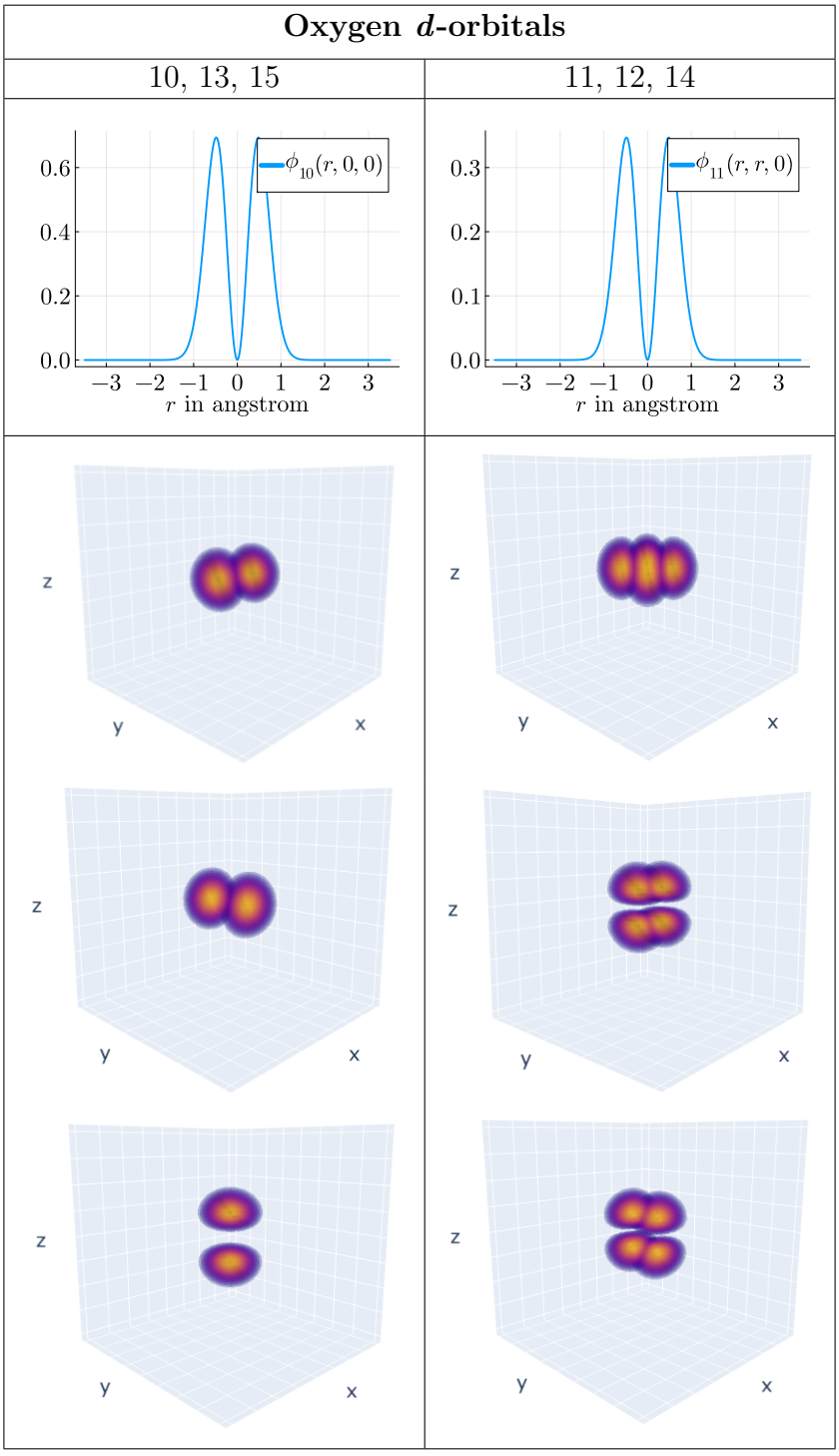


Figure 3.4: Atom centred density and x-axis plots of d-orbitals. The density is plotted on the volume $[-2, 2]^3$.

3.3 Approaches to the bottleneck

In DFT calculations the bottleneck is formed by the rapidly increasing number of basis functions N_{bas}. More basis functions make the computation of the coulomb matrix

$$J_{\mu\nu} = V_{\mu\nu\kappa\lambda}\, P_{\kappa\lambda} \tag{3.9}$$

$$V_{\mu\nu\kappa\lambda} = \iint \frac{\varphi_\kappa(\mathbf{r}) \cdot \varphi_\lambda(\mathbf{r}) \cdot \varphi_\mu(\mathbf{r}') \cdot \varphi_\nu(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}\, d\mathbf{r}\, d\mathbf{r}' \tag{3.10}$$

more expensive. Although one would expect a cost of $\mathcal{O}(N_{\text{bas}}^4)$, the computation of J can for large molecules be obtained for $\mathcal{O}(N_{\text{bas}}^2)$ due to a property of the atomic orbitals [8]. DFT is a general tool and is used on a big range starting from single atoms and reaching up to molecules with more than a thousand atoms.

Atoms that are spatially separated over a large distance have a negligible overlap and therefore do not contribute to the coulomb matrix. To this end, each atom only has an approximately constant number of relevant second atoms in his proximity. Each double index combination therefore has only $\mathcal{O}(N_{\text{bas}})$ non zero combinations.

In the concepts of our approaches we assume that we only regard molecules with such a screening. All statement also hold for smaller molecules, but without the screening all combinations have to be considered. In this case we have to replace $N_{\text{bas}} \to N_{\text{bas}}^2$ in the formulas.

Although the computational cost for the coulomb matrix can for large molecules be reduced, it still scales worse than other terms. We will now present three different approaches that may accelerate this algorithm step.

3.4 Approach 1: compression of orbital indices

If one chooses a gaussian basis the evaluation of the six dimensional integral in $V_{\mu\nu\kappa\lambda}$ can be done analytically for each fixed index combination $\mu\nu\kappa\lambda$. The tensor evaluation is in this case very quick and achieved in a constant time. The bottleneck is here given by the rapidly growing length of the orbital indices $\mu, \nu, \lambda, \rho = 1, \dots N_{\text{bas}}$, that lead to a costly contraction with the density matrix $P_{\kappa\lambda}$ of $\mathcal{O}(N_{\text{bas}}^2)$.

A possible solution to this bottleneck is a tensor decomposition in the orbital indices. We can summarize the indices $\kappa\lambda$ and $\mu\nu$ into big multi indices I and J. This is only a change in notation, since the sum runs over the pair. Filtering only the contributing elements of atoms close to each other, both indices have $\mathcal{O}(N_{\text{bas}})$ elements. With these multi indices V can be seen as a $I \times J$ matrix. The more general tensor decomposition reduces in this case to a matrix decomposition. We will further use terms as TCI and tensor trains, to remember that I and J describe multi indices, but want to raise attention to the fact that the approach is mathematical equivalent to a matrix decomposition.

If V has enough structure to interpolate the whole tensor from a few elements (i.e V has a

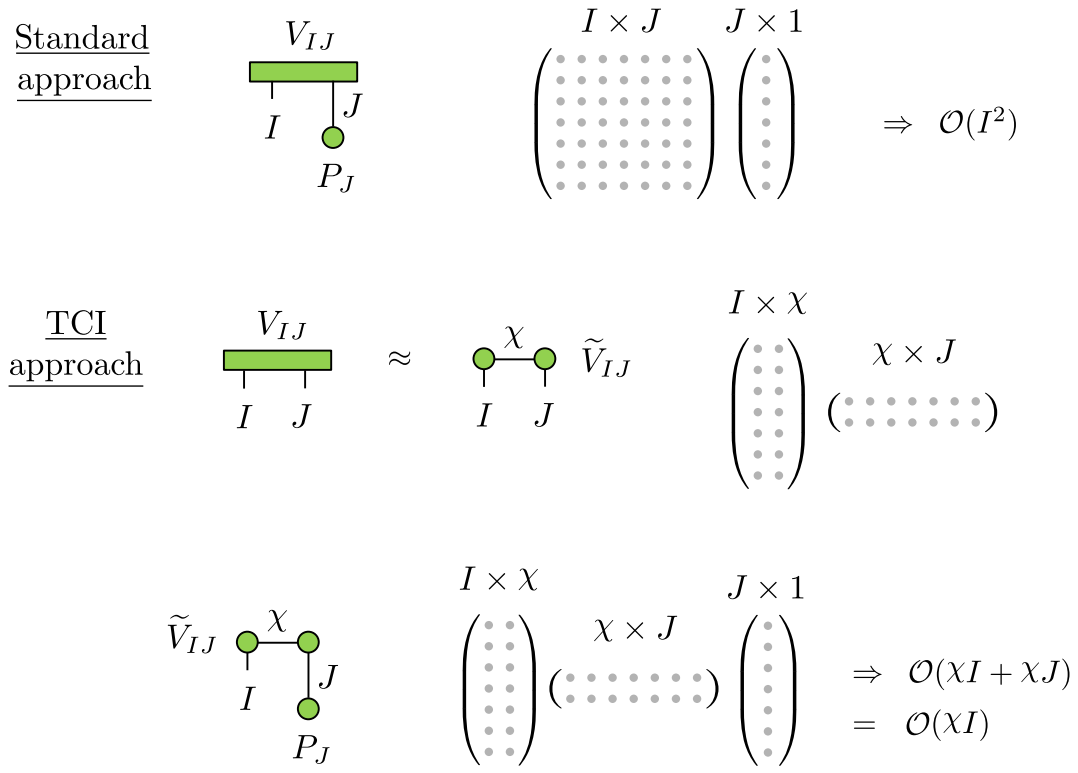


Figure 3.5: Accelerated contraction with density matrix by tensor decomposition

low rank), a TT could be build with TCI by just evaluating the analytic formula for these elements. The TT structure would then allow perform the contraction with the density matrix at a reduced cost. While the standard approach requires to add up for all I ($\kappa\lambda$) all values of J ($\mu\nu$) at a cost of $\mathcal{O}(IJ) = \mathcal{O}(I^2) = \mathcal{O}(N_{\text{bas}}^2)$, TCI can reduce this process to the order $\mathcal{O}(\chi I) = \mathcal{O}(\chi N_{\text{bas}})$, where χ denotes the rank of the TT. In the case $\chi << I$ this would speed up the calculation by a factor $\mathcal{O}(N_{\text{bas}})$, which is for large basis sets very desirable.

Since the density matrix P is in the multi-index representation already a degree one tensor, it is already fully decomposed and no further work is required.

Finding a low rank tensor train representation of V would allow to use bigger and more precise basis sets, due to the improved scaling behavior. But if this high compression really is achievable is an open question, since the orbitals have very different shapes and locations. To this end, the structure of J may be random and consequently not compressible. The crucial question of this approach will therefore be, if a low rank representation of V can be found or not.

3.4.1 Compression of full V-tensor

The most simple approach to obtain a tensor train from $V_{\mu\nu\kappa\lambda}$ is, to directly use it as an input to TCI. The data of this first test was provided by Markus Frankenbach.

basis	atom	N_{atom}	N_{bas}	χ	χ_{worst}
sto-3g	water	3	7	24	28
	ethane	8	16	71	136
	propanol	12	28	124	406
def2-svp	water	3	25	313	325
	ethane	8	60	1324	1830

Table 3.3: Compression in all four indices in *def2-svp* basis with tol=10^{-5}

The worstcase rank of this compression includes the symmetry of the V_{IJ} tensor in the multi indices I and J. Since the matrix is symmetric, we get a free compression factor of approximately 2 and the worstcase is only

$$\chi_{\text{worst}} = \frac{N_{\text{bas}} \cdot (N_{\text{bas}} + 1)}{2} \quad . \tag{3.11}$$

Table 3.3 shows the tensor train ranks of this approach for molecules of various sizes and different bases. We can see, that for larger molecules the compression improves. Nevertheless, the rank of the tensor increases with the number of the basis functions N_{bas}. Maybe the size of the selected molecules was below the saturation point and for larger molecules the rank reaches a converging point.

A simple compression in multi-indices is not sufficient to solve the bottleneck. To really speed up the calculation by a reduction of the evaluated orbitals, we would require $\chi << \chi_{\text{worst}}$, which is for our examined molecules not the case.

3.4.2 Concept of compression by parts

We have seen, that the V tensor can barely be compressed by an tensor decomposition. The rank of the tensor train increases with the the number of basis functions. If the compression remains at the fixed factor of approximately 1.2, for large basis set of molecules with more than thousand atoms this would lead to huge bond dimensions. We expected that the simple tensor decomposition will not suffice, since the problem exists for a long time and tensor decomposition is no new feature, so such a simple solution would have been found already. We proceed with decomposing V in a seminumerial way. The idea is, to separate the the integral into two parts. The coulomb kernel is then integrated with two of the gaussian basis functions analytical. The second integral is then evaluated numerical by quadrature

$$\begin{aligned} V_{\kappa\lambda\mu\nu} &= \iint \frac{\phi_\kappa(\mathbf{r}) \cdot \phi_\lambda(\mathbf{r}) \cdot \phi_\mu(\mathbf{r}') \cdot \phi_\nu(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r} \, d\mathbf{r}' \\ &= \int d\mathbf{r} \;\; \phi_\kappa(\mathbf{r}') \cdot \phi_\lambda(\mathbf{r}') \cdot \int d\mathbf{r}' \;\; \frac{\phi_\mu(\mathbf{r}) \cdot \phi_\nu(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} \\ &= \int d\mathbf{r} \;\; B_{\kappa\lambda}(\mathbf{r}) \cdot A_{\mu\nu}(\mathbf{r}) \quad . \end{aligned}$$

The A and P tensors are defined as:

$$A_{\mu\nu}(\mathbf{r}) = \int \frac{\phi_\nu(\mathbf{r}') \cdot \phi_\mu(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r}' \tag{3.12}$$

$$B_{\lambda\rho}(\mathbf{r}) = \phi_\lambda(\mathbf{r}) \cdot \phi_\rho(\mathbf{r}) \ . \tag{3.13}$$

The concept of the seminumerical evaluation is not new, see for example [8]. Our new contribution is to replace the numerical integration with QTCI.

First we construct tensor trains of the analytic coulomb integral $A_{\mu\nu}(\mathbf{r})$ and the overlap of the basis functions $B_{\mu\nu}(\mathbf{r})$. The compression includes for both tensors the orbital index $I = \mu\nu$ and the grid indices of $\mathbf{r}$, obtained by the quantics representation.

If both tensor trains $\widetilde{A}$ and $\widetilde{B}$ have low rank structure, the integral over $\mathbf{r}$ can be performed by tensor contraction and we obtain a low rank TT representation of V as an end result. Although we assumed it while working on this section we recently recognised that this tensor train can not have a smaller rank than the simple tensor decomposition of section 1.4.1. This means that the seminumerical approach will also not allow a higher compression. Solving the bottleneck by compressing the orbital indices is therefore not possible.

The results of this section are nevertheless relevant for our other approaches. We will proceed even though we can already say that this approach will not bring any improvements.

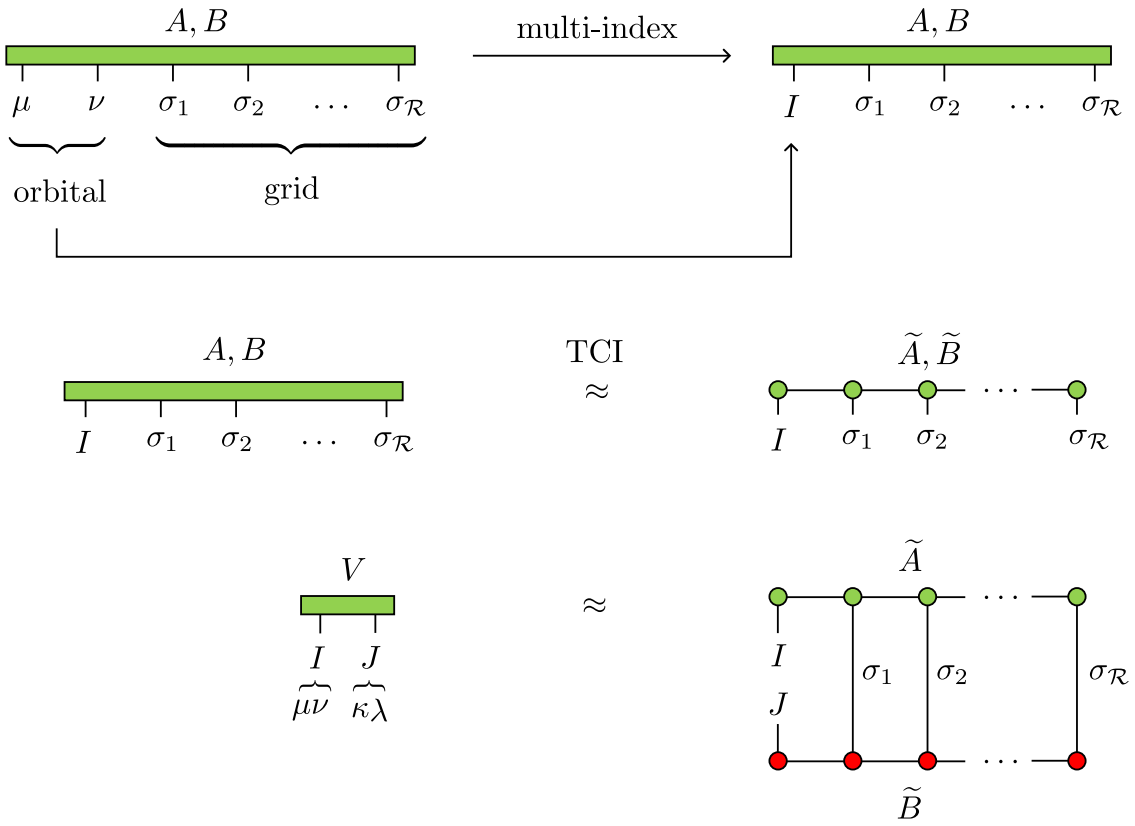


Figure 3.6: Evaluation of the integral by using tensor networks

3.4.3 Visualisation of A and B tensor

Before we start with the compression of the tensors, we want to have a look on the functional dependency. Because we already saw the single basis functions plotted for H_2O in the *def2-svp* basis, we also look at the A and B tensors for this molecule.

The B tensor simply is the overlap of two basis functions. The first plot of the following figure shows various combinations of atomic orbitals with the corresponding overlap. The functions are plotted along the connecting line between the kernels for distinct atoms, or respectively on the x-axis for functions of the same atom. The variable $r - R1$ denotes the distance on the plotted axis with respect to the position $R1$, of the first kernel (i.e. the first atom has its center at $r - R1 = 0$).

Looking at the definition of A we notice that it is a convolution of B with the coulomb kernel. The convolution smoothens the function and extends its support to a larger domain as can be seen in the second plot. To allow a comparison between A and B we plotted both again on the axis, connecting the kernels of the respective atoms.

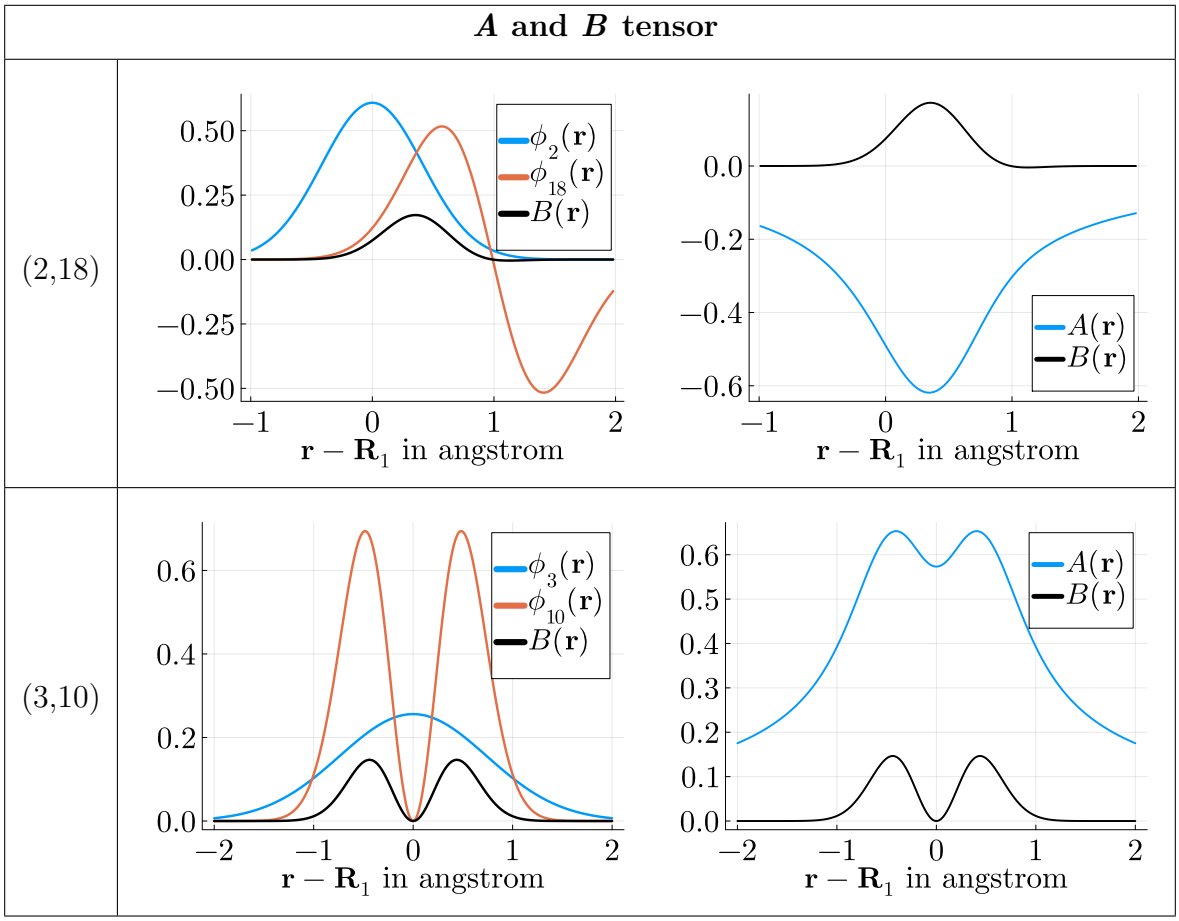


Figure 3.7: A and B tensor plotted on the connecting axis of the respective atoms.

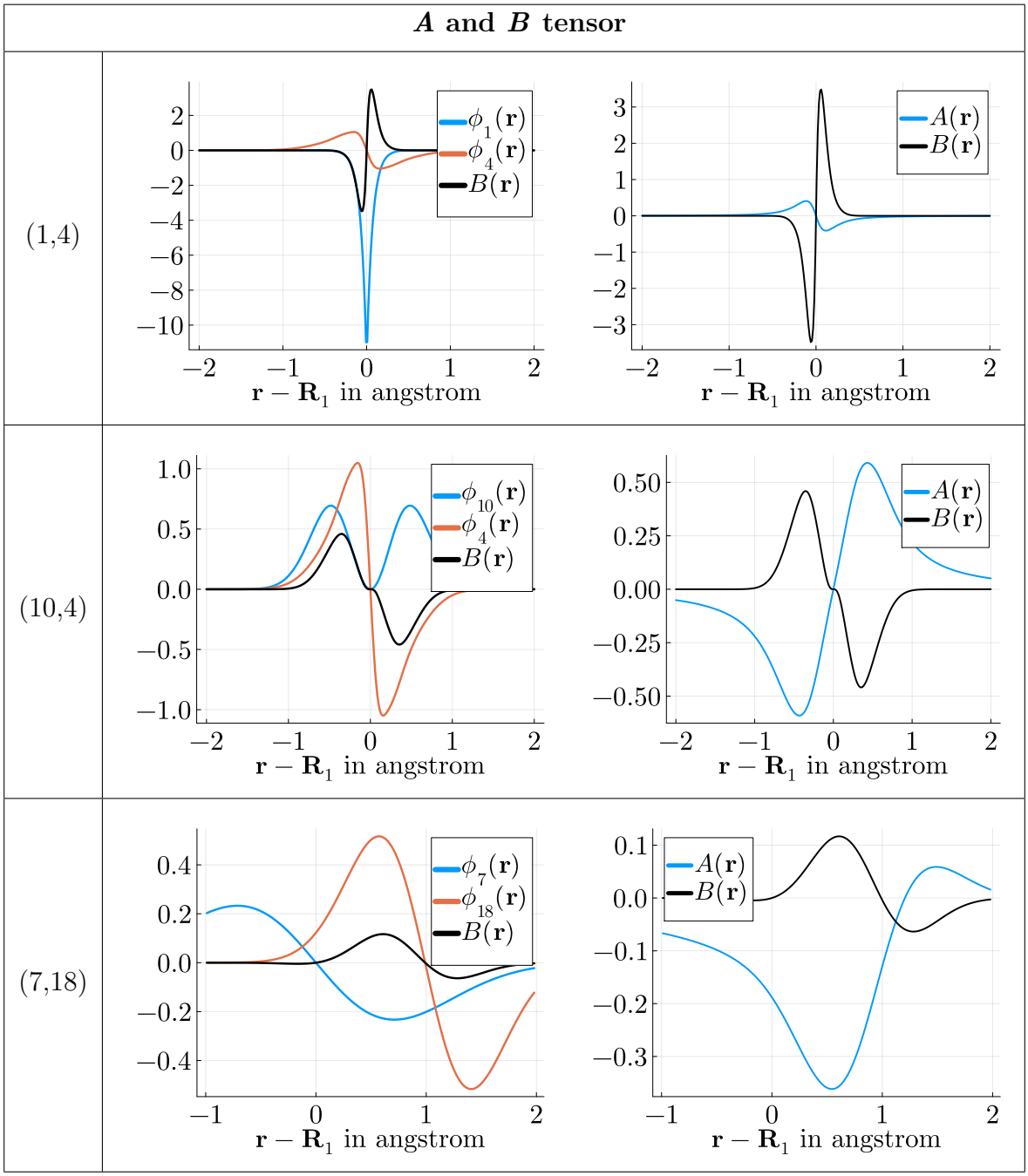


Figure 3.8: A and B tensor plotted on the connecting axis of the respective atoms.

3.4.4 Implementation and grid

We describe briefly the implementation in julia. We refer to all important functions and packages that were used.

The two functions A and B are implemented in julia with the use of the earlier named *GaussianBasis.jl* [1] package. The package contains the tabulated coefficients $\zeta_n, d_n, \ell_x, \ell_y, \ell_z$ of the Gauss functions (3.8), as well as an implemented function that returns the integral with the coulomb kernel. The B tensor is implemented directly by evaluating the gaussian basis formula (3.8) with the correct coefficients, while the A tensor is realised with the use of the pre implemented *overlap()* function of the *GaussianBasis.jl* package.

Before we can construct a TT from A and B , we have to choose a suitable grid. We use a cartesian grid that captures the whole volume of the molecule. On the one hand the grid has to be large enough to contain all relevant structures. By restricting the function to a finite volume, we neglect the contribution of the function tail to the integral. Increasing the grid volume minimizes this error. On the other hand, a large grid requires a fine grid resolution, to resolve small peaks and tiny structures. A smaller grid does therefore already resolve all features at a low resolution. Having these aspects in mind, we determined the limits of our grid, by evaluating each basis function at the closest point of each boundary wall. If one function value is larger than a limit value of 10^{-6} the boundary is increased. This process is repeated iterative until all functions fulfill the condition. By the monotone decay of the basis, this construction ensures that all basis functions are smaller then 10^{-6} outside the grid and therefore, $B_{\mu\nu}(\mathbf{r}) = \phi_\mu(\mathbf{r})\phi_\nu(\mathbf{r})$ is smaller then 10^{-12}. Since we cover with this method the support of any orbital function, the whole molecule is contained in this grid, which is why we will call this grid the 'molecular' grid.

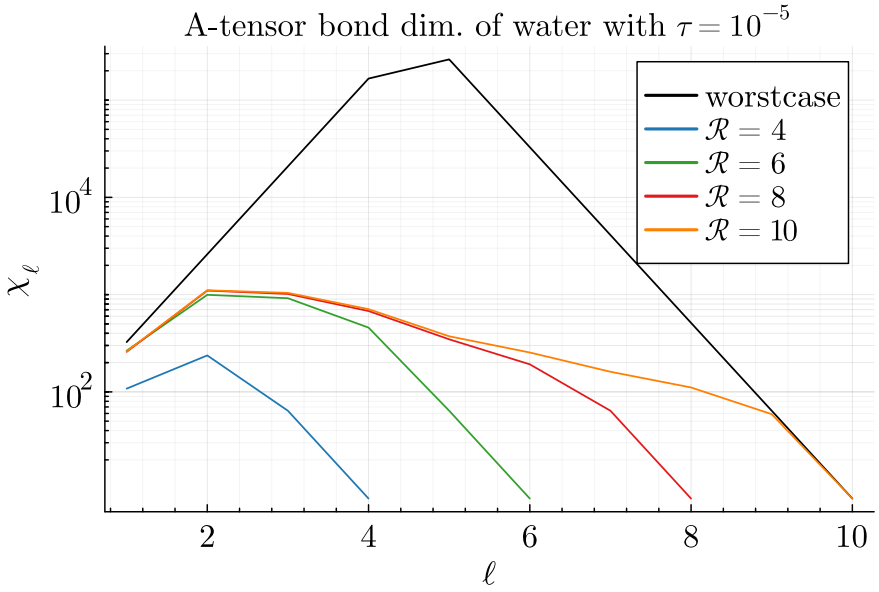
As we have seen in the last chapter $A_{\mu\nu}$ expands further than the $P_{\lambda\rho}$ tensor and does not fulfill the condition $A_{\mu\nu} < 10^{-12}$ at the boundary of the molecular grid. But since $A_{\mu\nu}$ is later contracted with $B_{\lambda\rho}$, only the the support of $B_{\lambda\rho}$ is relevant for the summation. This allows us to compress $A_{\mu\nu}$ also on the molecular grid.

3.4.5 Results for A

Figure 3.9 shows the bond dimensions of $\widetilde{A}$ at the example of the H_2O molecule in the *def2-svp* basis. The maximum bond dimension, i.e the rank of the TT, does not increase with $\mathcal{R}$, which implies that all relevant structures are already revealed at a medium grid resolution. The rank saturates at $\chi \approx 1100$, which is a rather high rank for a tensor train. The reason for this high rank has its origin in the orbital index. TCI only achieves a reduction from the worst case scenario with $\chi_{\text{worst}} = 325$ to $\chi \approx 260$ in the first index, which is a compression by the factor 1.2.

The bond dimension does even increase in the second index even though this index is a grid index of a smooth function and is therefore expected to be highly compressible. This increase is due to the bad compression of the orbital grid as we will see later, after performing a compression in only the grid indices.

As we explained in section 3.4 to really deliver a speedup against the standard approach $\chi << I$ is required. This condition is here with $\chi \approx I$ not fulfilled. Further does the small gain in the compression not outweigh the costs of constructing a tensor train for a large grid for several hours. The overhead of having to build the TT at first is simply too large for this molecule.



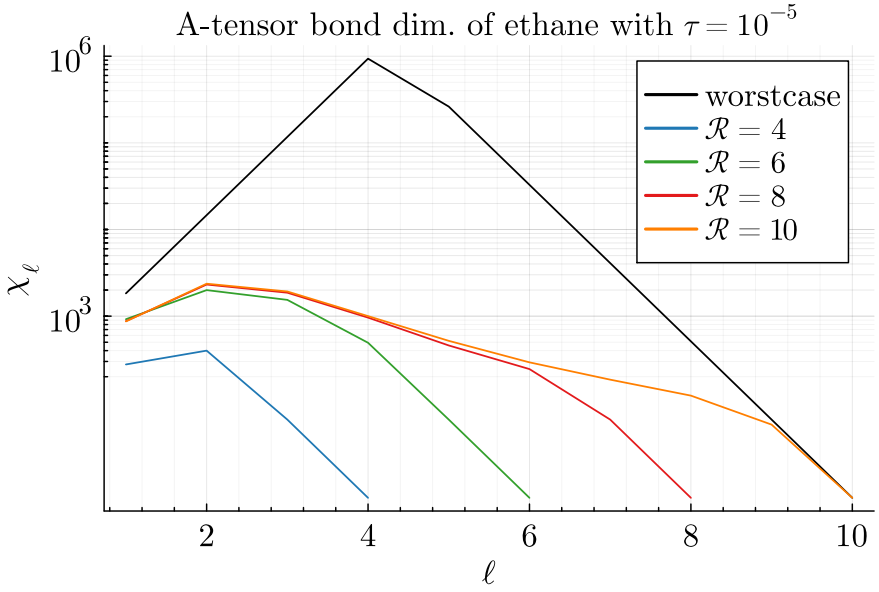
Grid bounds

$x \in [-4.31, 7.43)$
$y \in [-3.43, 7.69)$
$z \in [-5.22, 5.86)$

$N_{\text{bas}} = 25$
I = 325

$\mathcal{R}$	χ	t [h]
4	237	0.03
6	995	1.51
8	1100	3.58
10	1108	4.06

Figure 3.9: Bond dimension of the TT at every chain position ℓ for multiple grid resolutions. The table to the right shows the boundary of the used grid, the number N_{bas} of basis functions for the molecule, as well as the length of the multi-index $I = N_{\text{bas}}^2$. Also the rank of all presented tensor trains χ with the runtime t of the construction in hours is noted.



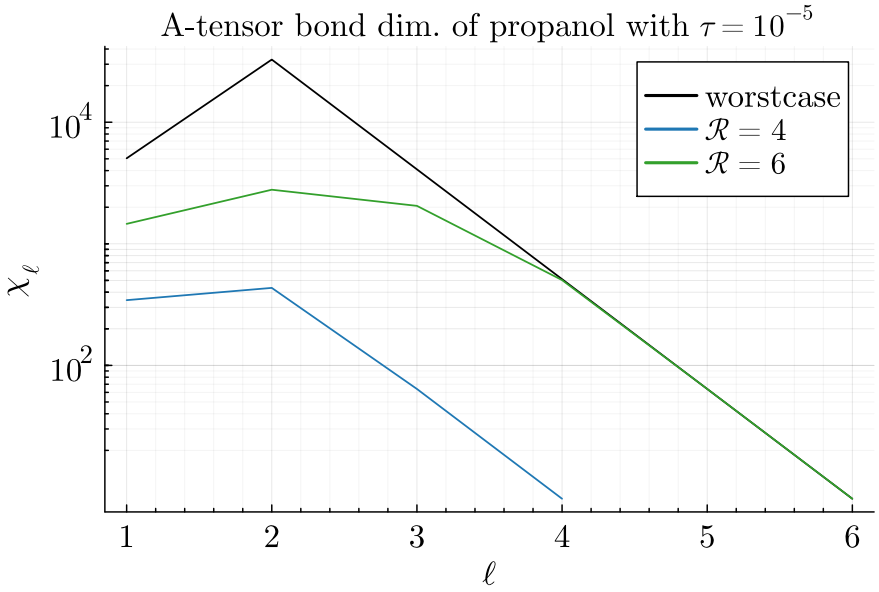
Grid bounds

$x \in [-6.41, 6.41)$
$y \in [-6.17, 6.17)$
$z \in [-6.22, 6.22)$

$N_{\text{bas}} = 60$
I = 1830

$\mathcal{R}$	χ	t [h]
4	400	0.18
6	1998	4.18
8	2317	7.18
10	2369	9.87

Figure 3.10: Bond dimension of the TT at every chain position l for multiple grid resolutions



Grid bounds

$x \in [-9.21, 5.42)$
$y \in [-5.00, 8.58)$
$z \in [-6.25, 6.41)$

$N_{\text{bas}} = 100$
$\text{I} = 5050$

$\mathcal{R}$	χ	t [h]
4	434	0.27
6	2786	29.57

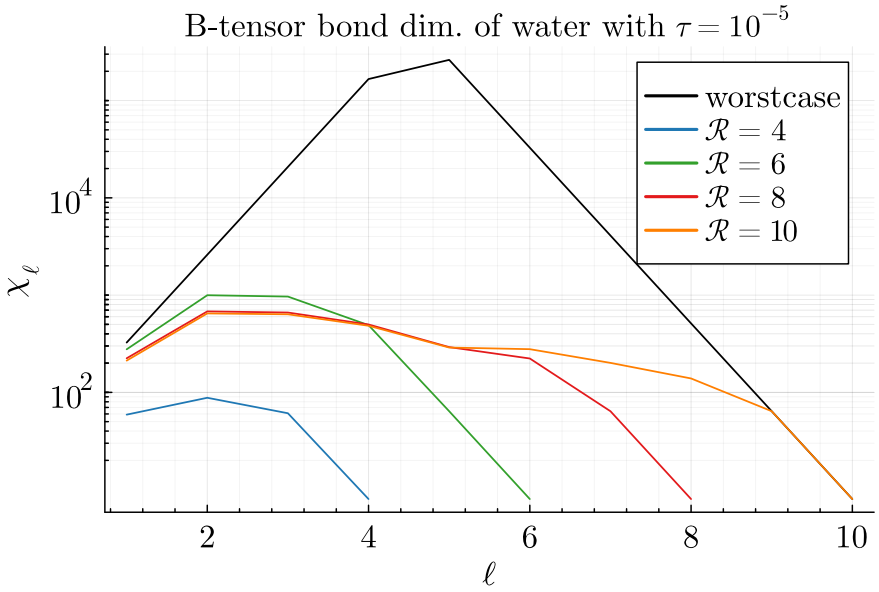
Figure 3.11: Bond dimension of the TT at every chain position l for multiple grid resolutions

The hope now was that the compression in the grid index could be better for larger molecules, and χ_1 saturates at a certain value when N_{bas} is increased. But the compression factor does not significantly improve for larger molecules. Figure 3.10 and 3.14 show the bond dimensions for the molecules ethane and propanol in the *def2-svp* basis.

Again the first index is compressed from the potential worst case $\chi_{\text{worst}} = 1830$ to $\chi = 900$ (i.e a factor 2) for ethane and from $\chi_{\text{worst}} = 5050$ to $\chi = 1450$ (i.e a factor 3.5) for propanol. The compression does indeed improve but the rank nevertheless rises, which leads to a enormous runtime for the TT construction. For larger molecules even worse ranks are to be expected. This is a big problem, since ethane and propanol are with six and twelve atoms not large molecules for DFT calculations, which can run up to more than thousand atoms.

3.4.6 Results for B

The computation of the integral in $\mathbf{r}$ in the V tensor requires tensor trains of A and B. If we compress the B tensor, we obtain very similar bond dimensions. In this case the ranks are around 15% lower, but still too large to achieve a contraction that could challenge the conventional schemes. Since B has a similar but yet different functional dependency compared to A this result supports our hypothesis that the compression over different orbital indices is responsible for the high ranks and not the shape of the function itself.



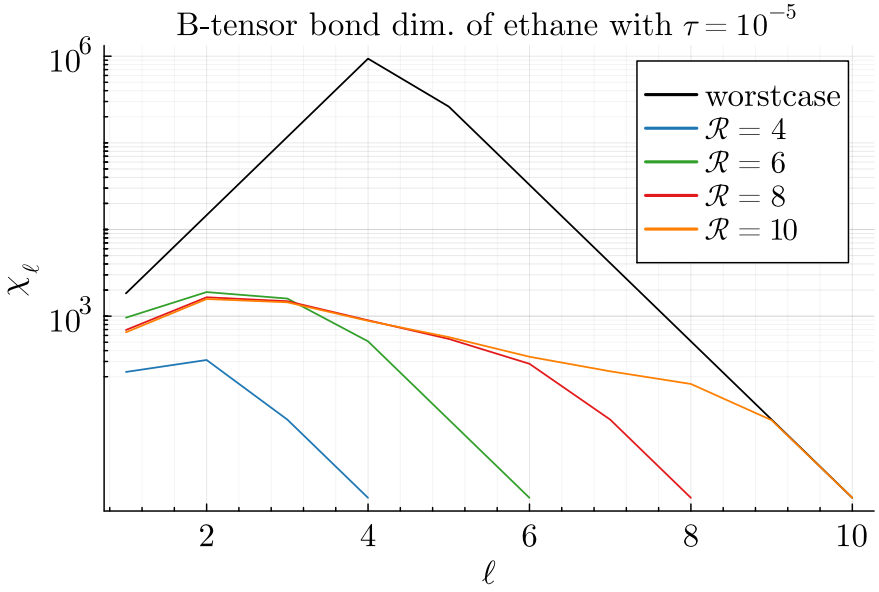
Grid bounds

$x \in [-4.31, 7.43)$
$y \in [-3.43, 7.69)$
$z \in [-5.22, 5.86)$

$N_{\text{bas}} = 25$
I = 325

$\mathcal{R}$	χ	t [h]
4	88	0.005
6	997	1.16
8	680	1.17
10	648	1.66

Figure 3.12: Bond dimension of the TT at every chain position l for multiple grid resolutions



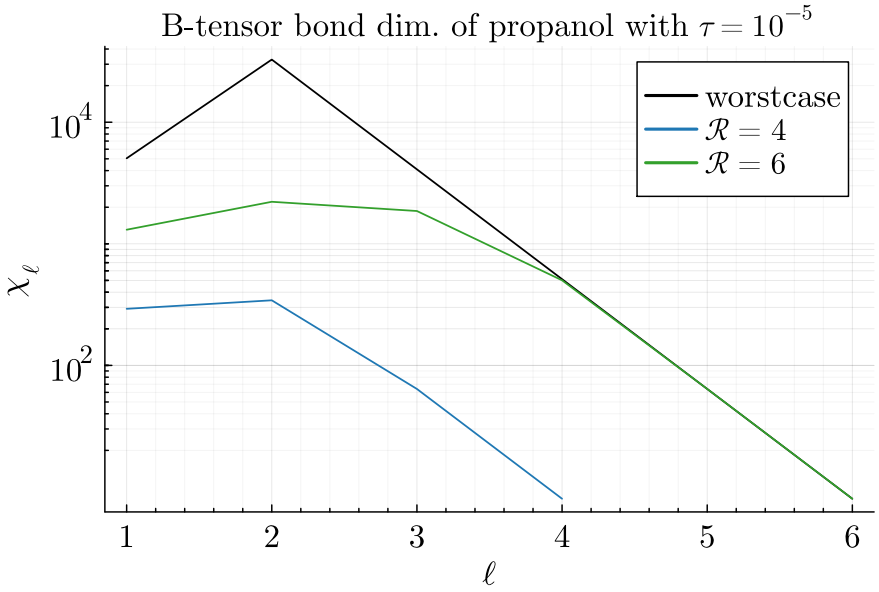
Grid bounds

$x \in [-6.41, 6.41)$
$y \in [-6.17, 6.17)$
$z \in [-6.22, 6.22)$

$N_{\text{bas}} = 60$
I = 1830

$\mathcal{R}$	χ	t [h]
4	312	0.09
6	1897	6.94
8	1652	5.99
10	1576	6.54

Figure 3.13: Bond dimension of the TT at every chain position l for multiple grid resolutions



Grid bounds

$x \in [-9.21, 5.42)$
$y \in [-5.00, 8.58)$
$z \in [-6.25, 6.41)$

$N_{\text{bas}} = 100$
$\text{I} = 5050$

$\mathcal{R}$	χ	t [h]
4	343	0.23
6	2217	12.42

Figure 3.14: Bond dimension of the TT at every chain position l for multiple grid resolutions

3.5 Approach 2: Contraction with electron density

Our first approach aimed to speed up the summation over the orbital indices with TCI

$$J_{\mu\nu} = \sum_{\kappa\lambda} V_{\mu\nu\kappa\lambda} P_{\kappa\lambda} \ . \tag{3.14}$$

By making use of definition of the A tensor we can rearange this equation and include the contraction with $P_{\kappa\lambda}$ directly into the integrals:

$$J_{\mu\nu} = \sum_{\kappa\lambda} P_{\kappa\lambda} V_{\mu\nu\kappa\lambda} \tag{3.15}$$

$$= \sum_{\kappa\lambda} P_{\kappa\lambda} \int d\mathbf{r} \ \ \phi_\kappa(\mathbf{r}')\phi_\lambda(\mathbf{r}') \cdot \int d\mathbf{r}' \ \ \frac{\phi_\mu(\mathbf{r}) \cdot \phi_\nu(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} \tag{3.16}$$

$$= \int d\mathbf{r} \ \ \left(\sum_{\kappa\lambda} P_{\kappa\lambda} \ \ \phi_\kappa(\mathbf{r}')\phi_\lambda(\mathbf{r}') \right) \cdot \int d\mathbf{r}' \ \ \frac{\phi_\mu(\mathbf{r}) \cdot \phi_\nu(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} \tag{3.17}$$

$$= \int d\mathbf{r} \ \ \rho(\mathbf{r}) \cdot A_{\mu\nu}(\mathbf{r}) \ . \tag{3.18}$$

In the last step we used the defintions

$$\rho(\mathbf{r}) = \sum_{\kappa\lambda} P_{\kappa\lambda} \ \ \phi_\kappa(\mathbf{r}')\phi_\lambda(\mathbf{r}') \tag{3.19}$$

$$A_{\mu\nu} = \int d\mathbf{r}' \ \frac{\phi_\mu(\mathbf{r}) \cdot \phi_\nu(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} \ . \tag{3.20}$$

We can compute a fixed matrix element $J_{\mu\nu}$ with QTCI by constructing tensor trains of ρ and A and performing the integration with tensor contraction. The full coulomb matrix can then be computed at the cost of $\mathcal{O}(\chi^3 N_{\text{bas}} \log(N_{\text{grid}}))$, where N_{grid} is the number of grid points that are needed to compute the numerical integral with QTCI.

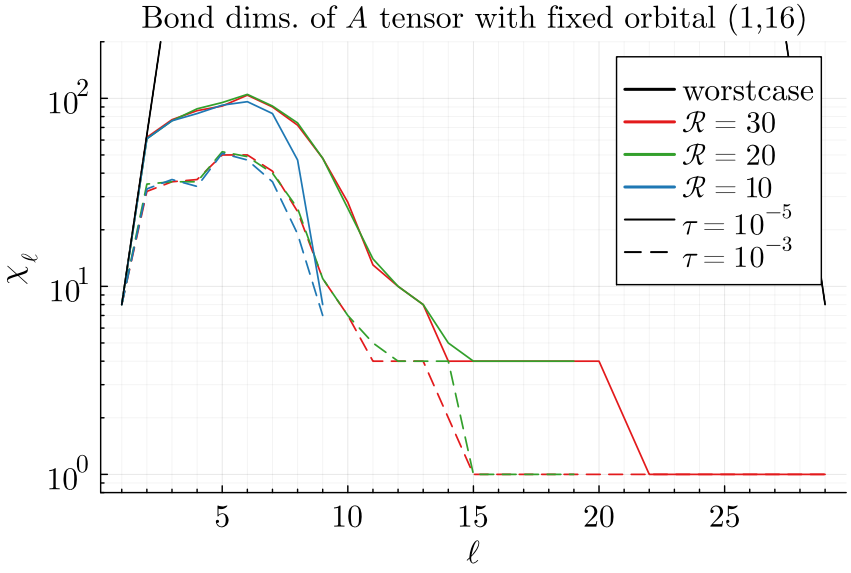
For a low rank of the two tensors, this approach delivers a speedup compared to the standard approach, which scales with $\mathcal{O}(N_{\text{bas}}^2)$.

3.5.1 Compression of A with fixed indices

We investigated the rank of the already implemented A tensor. The rank of the density matrix ρ could due to the limited time scale of this bachelor thesis not be investigated.

At first we have to fix the orbital pair $\kappa\lambda$. Figure 3.15 shows the bond dimensions of the compression of $A_{\mu\nu}$ with $\mu\nu = (1, 16)$ of H_2O in the *def2-svp* basis. To save space the ranks of further index combinations are listed in table 3.4. Since the expansion of ρ is limited to the volume of the molecule, we can again use our molecular grid and do not have to search a larger grid that captures the whole support of A. To this end, all compression used the grid in the table of figure 3.15 regardless of the index combination.

The rank of the TT depends on the tolerance of the compression. For a smaller tolerance, more pivots have to be added, to fulfil the error condition of the approximation. For both tolerances the bond dimension saturates very quickly and reveals the low rank structure of the basis functions with $\chi \approx 105$ for $\tau = 10^{-5}$ and $\chi \approx 51$ for $\tau = 10^{-3}$. Furthermore, for all examined index pair combinations, the rank does not exceed $\chi = 120$ (for tolerance $= 10^{-5}$, so we conclude that the low rank structure is not a feature of some orbital index



Grid bounds

$x \in [-4.31, 7.43)$
$y \in [-3.43, 7.69)$
$z \in [-5.22, 5.86)$

tol	$\mathcal{R}$	χ	t [s]
10^{-3}	10	51	5.1
	20	52	7.7
	30	50	6.8
10^{-5}	10	96	24.3
	20	105	58.7
	30	104	57.1

Figure 3.15: Bond dimension of the TT at every chain position ℓ for multiple grid resolutions

combinations, but a general characteristic of the used basis.

To be able to compute a coulomb matrix element, we now would have to find a tensor representation of the electron density ρ. This will be the topic of future research. Since ρ a smooth function in $\mathbf{r}$, it is very likely to have a low rank structure. Because of the already revealed low rank structure of the A tensor at fixed orbital combinations, it seems worthwhile to investigate this approach in the future.

μ	ν	tol=3	tol=4	tol=5
2	4	66	98	134
2	9	65	87	124
4	17	62	92	131
4	25	68	102	138
6	12	86	117	157
6	17	62	96	135
7	9	62	101	140
8	12	77	111	147
9	1	63	100	139
9	12	70	104	144
10	18	56	100	128
11	6	99	120	170
11	8	68	103	143
11	17	67	96	137

μ	ν	tol=3	tol=4	tol=5
15	2	72	105	147
13	7	69	103	146
17	1	50	77	113
17	6	63	93	138
17	10	58	91	126
17	25	65	84	113
17	19	63	88	120
18	25	60	88	130
19	1	60	93	135
22	9	64	99	133
23	22	69	88	113
24	16	61	93	133
24	17	57	84	116
25	1	51	78	115

	tol=3	tol=4	tol=5
min	50	77	113
max	99	120	170

Table 3.4: Tensor train ranks for further index combinations. The rank was determined in a compression with $\mathcal{R} = 30$ and is not expected to increase for arbitrary large $\mathcal{R}$ due to the saturation of the rank.

3.6 Approach 3: Using a non gaussian basis

In our first two approaches, we chose a gaussian basis to compute the integrals analytically. Alternatively we could solve the bottleneck by reduce N_{bas} with the selection of a better basis. This idea was originally proposed in [10]. Non-gaussian bases are not limited to a specific functional dependency and can approximates the orbital structure better than the Gauss functions.

In this case fewer basis elements are required to get precise results, but the integral in the V tensor cannot be done analytically anymore. Integration now has to be done for every fixed index pair numerically. This new bottleneck might be resolved by the QTCI-algorithm, which constructs low rank tensor trains from the integrand and evaluates the integral by tensor contraction numerically. This method requires a TT construction for every index pair,

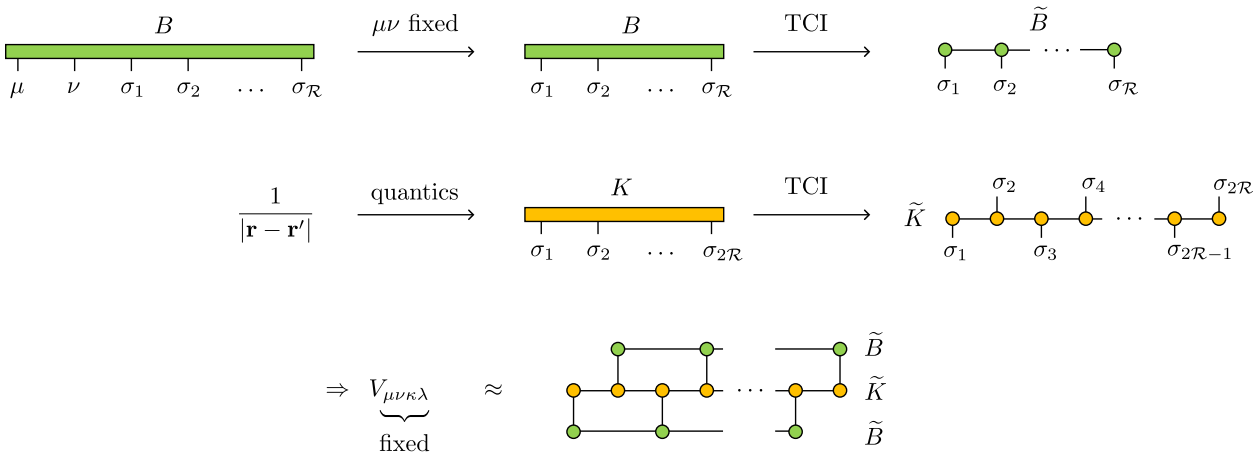


Figure 3.16: Evaluation of the integral by using tensor networks

but since the basis is chosen to approximate the orbitals well, N_{bas} is rather low.

Our task has now shifted. Instead of finding a way to compress the orbital indices, we now search a way to numerically evaluate the V integral fast for every fixed index combination. The advantage of this method is, that now only grid indices are contracted. The basis functions have a smooth form in $\mathbf{r}$ and therefore are expected to possess a low rank structure.

The main task of this method is to find a suitable basis, that approximates the true atomic structure very well and features a low rank structure. Since the use of the gaussian basis is so common in the DFT community, finding a non gaussian one is no easy task. Some programs, such as as FHI-AIMS [2], feature a suitable set, but are not available open source.

The goal of this section is to demonstrate the potential of this method and show that further research might be fruitful. For a working algorithm a suitable basis has to be chosen and examined. Since we have already investigated the ranks of the A tensor, we will also present the ranks of the B tensor. The gaussian basis is not suitable as an basis, since the whole reason for this approach is to be able to use other functions. The Gauss functions are nevertheless smooth basis functions and thus will share some properties with an arbitrary non gaussian basis. Studying the ranks of the Gauss functions will hence give an impression about the potential of this approach.

3.6.1 Grid

For a fixed index pairs $\mu\nu$ and $\kappa\lambda$ not the whole molecular grid is required for the compression. Because of the contraction with $B_{\kappa\lambda}$, only its support contributes to the end result. An optimal algorithm would therefore find for every $\kappa\lambda$ a fitting grid. For a better comparison between the results with free and fixed orbital indices we nevertheless use again the whole molecular grid for the contraction, as described in 3.4.4.

3.6.2 Compression results of B with fixed indices

We again have to fix the orbital pair $\kappa\lambda$. Figure 3.17 shows the bond dimensions of the compression of $B_{\mu\nu}$ with $\mu\nu = (1, 16)$ of H_2O in the *def2-svp* basis. The plot is very similar to the already performed compression of the A tensor.

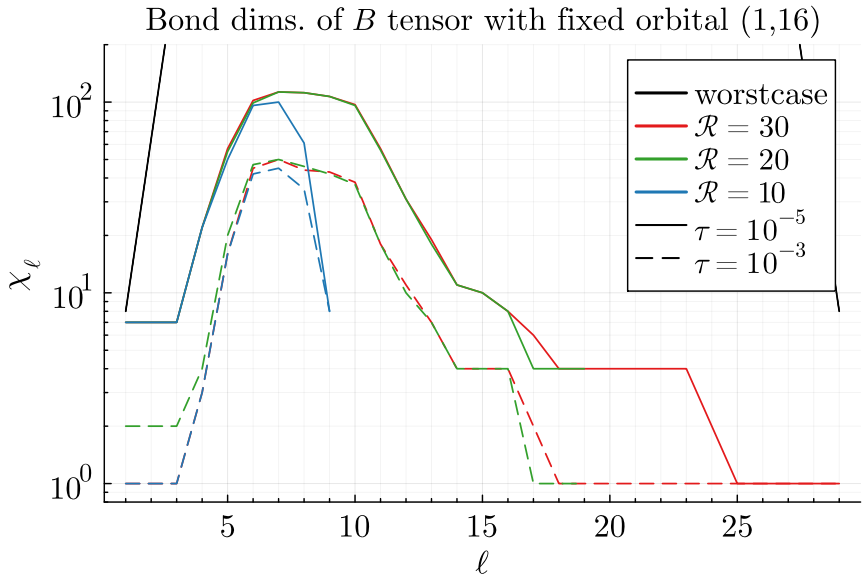


Figure 3.17: Bond dimension of the TT at every chain position ℓ for multiple grid resolutions

To compare the two different functions, we also included the data of the A tensor compression in table 3.5

A tensor			
tol	$\mathcal{R}$	χ	t [s]
10^{-3}	10	51	5.1
	20	52	7.7
	30	50	6.8
10^{-5}	10	96	24.3
	20	105	58.7
	30	104	57.1

B tensor			
tol	$\mathcal{R}$	χ	t [s]
10^{-3}	10	45	1.3
	20	5	6.1
	30	50	4.6
10^{-5}	10	100	9.5
	20	113	87.9
	30	113	83.8

Table 3.5: Comparison of A and B tensor for at fixed orbital (1,16)

Although the two tensors have different functional dependencies, both tensors achieve an enormous compression and saturate at a rank of $\chi \approx 110$ for $\tau = 10^{-5}$. This encourages our hypothesis that any arbitrary basis with a smooth dependency in $\mathbf{r}$ will have a low rank structure. Our approach of choosing a non gaussian basis has therefore much potential to accelerate the bottleneck.

3.7 Error Analysis

3.7.1 Error types and dependence in N_{grid}

As we explained in the introduction to TCI, three approximations are made for the integration:

1. Limiting the integral on a finite volume
2. Discretizing the integral into a Riemann sum
3. Approximation of the function by interpolation

These three approximations lead to errors for the calculated integral value. The significance of these errors and their dependence in N_{grid} shall now be investigated. At first we limit the integral to the finite volume:

$$\begin{aligned} I &= \int_{\mathbb{R}^{\mathcal{N}}} f(x_a, \ldots, x_{\mathcal{N}}) d^{\mathcal{N}}\mathbf{x} \\ &= \int_V f(\mathbf{x}) d^{\mathcal{N}}\mathbf{x} + \int_{\overline{V}} f(\mathbf{x}) d^{\mathcal{N}}\mathbf{x} \ . \end{aligned}$$

V represents here the finite integration volume and $\overline{V}$ its complement, i.e the rest of the space. The error by neglecting the function tail ϵ_{tail} is independent of the number of grid points, since we haven't yet chosen a resolution. It is a constant offset between the convergence limit of our TCI result and the true analytic integral value. If the decay behavior of the integrated function is known, the value of ϵ_{tail} can be estimated.

Another error comes from the approximation of the integral by the Riemann sum:

$$\begin{aligned} \int_V f(\mathbf{x}) d^{\mathcal{N}}\mathbf{x} &= \frac{V}{N_{\text{grid}}} \sum_{n=1}^{N_{\text{grid}}} f(\mathbf{x}_n) + \epsilon_{\text{grid}} \\ &= \frac{V}{N_{\text{grid}}} \sum_{\sigma} f_\sigma + \epsilon_{\text{grid}} \ . \end{aligned}$$

The error of the Riemann sum converges as $\mathcal{O}(1/N_{\text{grid}})$.

Last but not least we do get an error by approximating the function in form of a tensor train

$$\begin{aligned} \frac{V}{N_{\text{grid}}} \sum_{\sigma} f_\sigma &= \frac{V}{N_{\text{grid}}} \sum_{\sigma} (\widetilde{f}_\sigma + \Delta f_\sigma) \\ &= \frac{V}{N_{\text{grid}}} \sum_{\sigma} \widetilde{f}_\sigma + \frac{V}{N_{\text{grid}}} \sum_{\sigma} \Delta f_\sigma \\ &= I_{TCI} + \epsilon_{\text{approx}} \ . \end{aligned}$$

The total error resulting from the approximation of the function depends on the error distribution Δf_σ of the grid points. The tolerance condition of the tensor trains construction

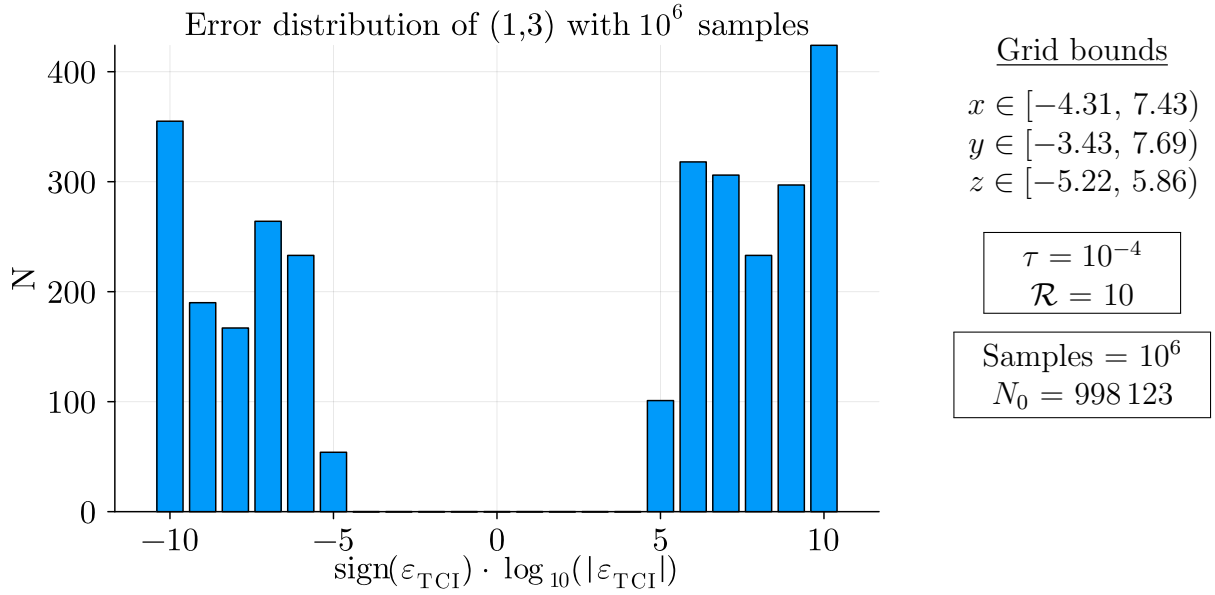


Figure 3.18: Error distribution of ϵ_{TCI}. N_0 denotes all points that deviate less than 10^{-10} from the original function.

delivers an upper bound for the error

$$\frac{|\Delta f_{\boldsymbol{\sigma}}|}{f_{\max}} \leq \tau \ , \tag{3.21}$$

where τ denotes the given tolerance. For a further analytic calculation the exact distribution would be required but is not available to us. Also the distribution varies for distinct integrated functions and also contains a random component as we shall see later. Nevertheless, we can derive some general statements by investigating the error distribution of a gaussian basis function.

We constructed a tensor train of the $B_{\mu\nu}$ tensor at a fixed orbital value $\mu\nu$. Then we subtracted the exact tensor value from the TCI approximation and divided it by the function maximum for 10^6 random grid point samples

$$\epsilon_{\mathrm{TCI}} = \frac{\Delta B_{\boldsymbol{\sigma}}}{B_{\max}} = \frac{\widetilde{B}(\boldsymbol{\sigma}) - B(\boldsymbol{\sigma})}{B_{\max}} \ . \tag{3.22}$$

From equation (3.21) we get that $\epsilon \leq \tau$. Figure 3.18 shows the distribution of the errors for the orbital (1,3) at tolerance $\tau = 10^{-4}$ and grid resolution $\mathcal{R} = 10$. The distribution is plotted in the order of magnitude of the error times the sign. A value of 54 at the x position -5 means for example that a negative deviation between $10^{-5} < 10^{-4}$ appeared 54 times during the 10^6 samples. We only decided to plot deviations until the order of 10^{-10}, since smaller values do not contribute to the overall error.

While for most grid points the error of the approximation is negligible, some grid points deviate with an error $0.1 \cdot \tau < \epsilon_{\mathrm{TCI}} < \tau$. The upper limit of the tolerance can clearly be seen

in the distribution. In rare cases we also get errors above the maximal allowed error. TCI checks condition (3.21) only for the evaluated points. Since only a fraction of the whole grid is evaluated, the approximation may be for some points worse than the given tolerance. The number of such points is very small as can be seen in the distribution plot, but when they appear, they significantly contribute to the error of the approximation.

One can also note that the distribution is not perfectly symmetric. We notate the number of points with positive/negative errors with N_+ and N_-. These parameters contain the number of points that contribute with an error of τ, so for our upper example of $\tau = 10^{-4}$ the points with error 10^{-5} contribute with a factor of 0.1

$$N_- = N_{-3} \cdot 10 + N_{-4} + N_{-5} \cdot 0.1 + N_{-6} \cdot 0.01 \tag{3.23}$$

$$N_+ = N_{+3} \cdot 10 + N_{+4} + N_{+5} \cdot 0.1 + N_{+6} \cdot 0.01 \ , \tag{3.24}$$

where $N_{\pm x}$ notates the number of points with an error of the magnitude $\pm 10^{-x}$.

The number of points with maximal error is proportional to the total number of grid points. This can be easily seen in plot 3.19, which shows a plot of the number of grid points with maximum error from a sample of 10^6 points. The number does not decrease in $\mathcal{R}$, so we can conclude that a constant fraction of the grid points has a maximal deviation

$$N_\pm \approx \frac{12}{10^6} N_{\text{grid}} = 1.2 \cdot 10^{-5} \cdot N_{\text{grid}} \ . \tag{3.25}$$

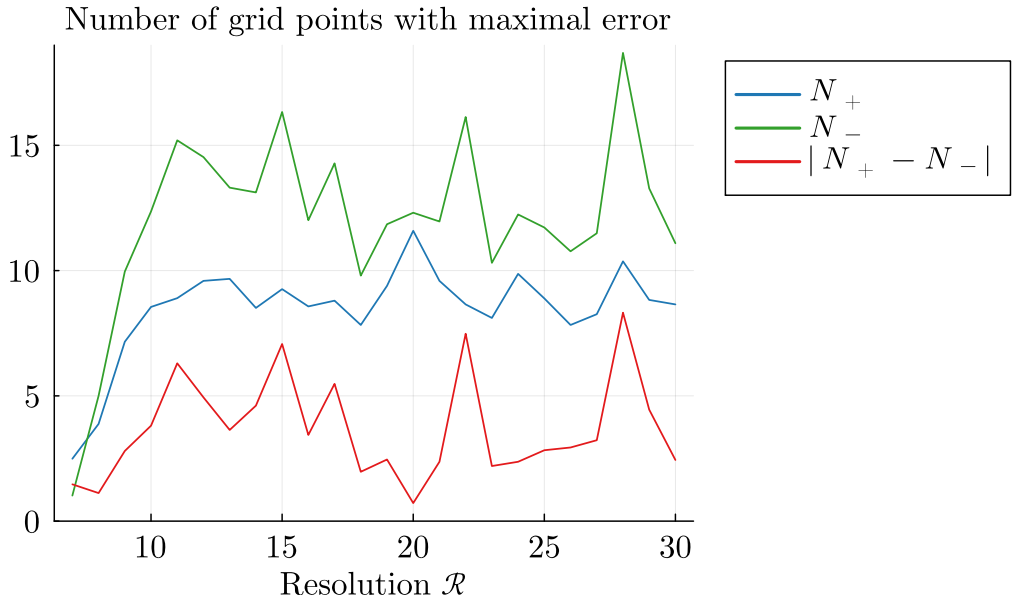


Figure 3.19: Asymmetry in the error distribution in 10^6 samples, that leads to an error of τ.

Also the asymmetry $|N_+ - N_-|$ does not decrease for higher $\mathcal{R}$, so we have to expect a remaining error that is proportional to N_{grid}:

$$|N_+ - N_-| \approx a \cdot N_{\text{grid}} \tag{3.26}$$

$$\sum_\sigma \Delta B_\sigma \approx a \cdot N_{\text{grid}} \cdot B_{\max} \cdot \tau \ , \tag{3.27}$$

where we can fit $a \approx 8 \cdot 10^{-6}$ from our data. Finally we can estimate

$$\epsilon_{\text{approx}} = \frac{V}{N_{\text{grid}}} \sum_\sigma \Delta f_\sigma \approx \frac{V}{N_{\text{grid}}} \cdot a \cdot N_{\text{grid}} \cdot f_{\max} \cdot \tau \tag{3.28}$$

$$= V \cdot a \cdot f_{\max} \cdot \tau \ . \tag{3.29}$$

Summarizing the upper derivation for the errors, we can approximate the analytic integral value with TCI by

$$I = I_{\text{TCI}} + \epsilon_{\text{tail}} + \epsilon_{\text{grid}} + \epsilon_{\text{approx}} \ , \tag{3.30}$$

where the errors have the following dependencies

$$\epsilon_{\text{tail}} \propto const \tag{3.31}$$

$$\epsilon_{\text{grid}} \propto \frac{1}{N_{\text{grid}}} \tag{3.32}$$

$$\epsilon_{\text{approx}} \propto \tau \ . \tag{3.33}$$

3.7.2 Error discussion of V tensor

As we have seen in the last sections, the runtime and required memory depend heavily on the input parameters i.e the tolerance τ and grid resolution $\mathcal{R}$. In the case of a compressible functions like the A and B tensors for fixed orbital indices, the runtime increases at a cheap cost $\mathcal{O}(\mathcal{R})$ linear in $\mathcal{R}$, but nevertheless it is advisable to find parameters that match the required precision of the end result, to save time and resources.

After have we found the convergence behavior of the different error sources in the last section, we now continue with a quantitative example. The integral values of the V tensor are analytically known and therefore perfectly suited for an error analysis.

We chose to compare the results for the orbital combination $(\mu\nu|\kappa\lambda) = (1, 16|3, 17)$. Figure 3.20 shows the relative error $|(I - I_{\text{TCI}})/I|$ of the analytic integral value I and our TCI approximation I_{TCI} for increasing $\mathcal{R}$ and different tolerances τ. For each $\mathcal{R}$ value the contraction was performed five times and the minimal error was plotted. This avoids the statistical errors of the algorithm (more details in the next section)

We can clearly see our derived dependencies of the different errors. For small resolutions $\mathcal{R}$ the grid error ϵ_{grid} dominates and converges with $1/N_{\text{grid}}$. After reaching a certain precision the constant error of ϵ_{approx} becomes relevant and the convergences in the accuracy terminates.

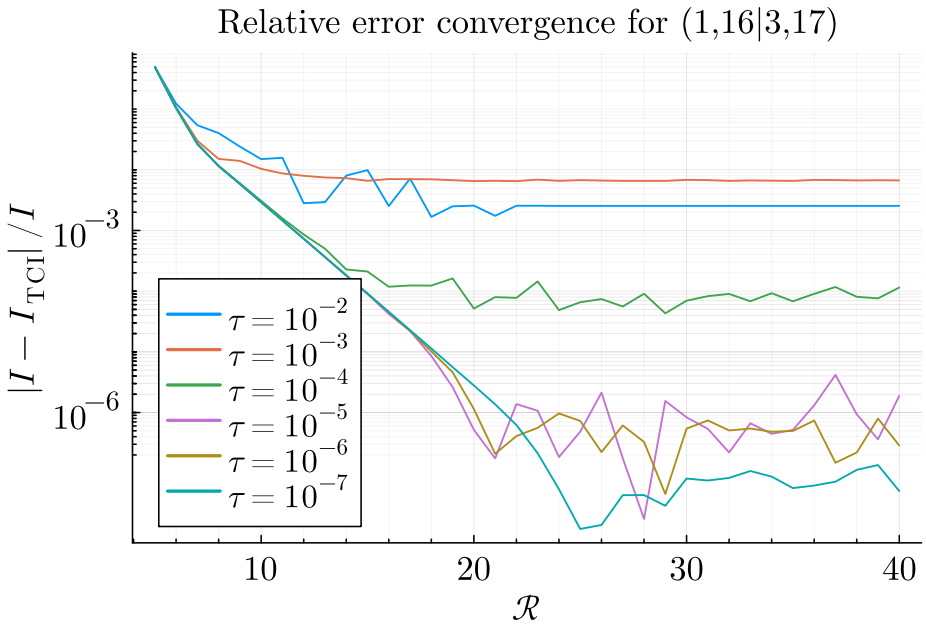


Figure 3.20: Absolute error $|(I - I_{\mathrm{TCI}})/I|$ for different $\mathcal{R}$ and τ

Since $\epsilon_{\mathrm{approx}}$ is proportional to τ, an increase in $\mathcal{R}$ does not change this error. A higher grid resolution will thus not always result in better integral approximations. The dependency of the constant convergence limit can clearly be seen in the graphic. We also notice two anomalies. Tolerance $\tau = 10^{-6}$ and $\tau = 10^{-7}$ converge at the same level and $\tau = 10^{-2}$ even has an higher accuracy then $\tau = 10^{-3}$. The reason for this strange behavior could lie in the specific form of the function. Maybe the unprecise approximation with $\tau = 10^{-2}$ by chance converges against a tensor train that eliminates approximation errors of the same magnitude.

Apart from that, the error convergence has the expected dependency in $\mathcal{R}$ and τ. By setting a limit on the required precision, we can determine from this graphic the first tolerance that achieves the limit and also find the minimum $\mathcal{R}$ that is required. This gives us a method to determine appropriate input parameters for the TCI algorithm.

3.7.3 Statistical error

The TCI algorithm searches a given tensor for structure it can exploit, to construct a low rank TT. The search for suitable pivots contains a random component, which makes the TCI algorithm not completely deterministic. The randomness is obtained during the selection of the global pivots.

The search is optimized to find the best pivots and deliver a good approximation. However in rare cases, the random component of the global pivot search leads to an approximation other than the best possible one. For these cases one obtains a higher error. This effect makes the TCI results statistical. We investigated the variation over different compression runs again for the orbital $(\mu\nu|\kappa\lambda) = (1, 16|3, 17)$ at tolerance $\tau = 10^{-6}$ and $\mathcal{R} = 11$. We used a sample of 1000 runs. The distribution of the errors can be seen in figure 3.21.

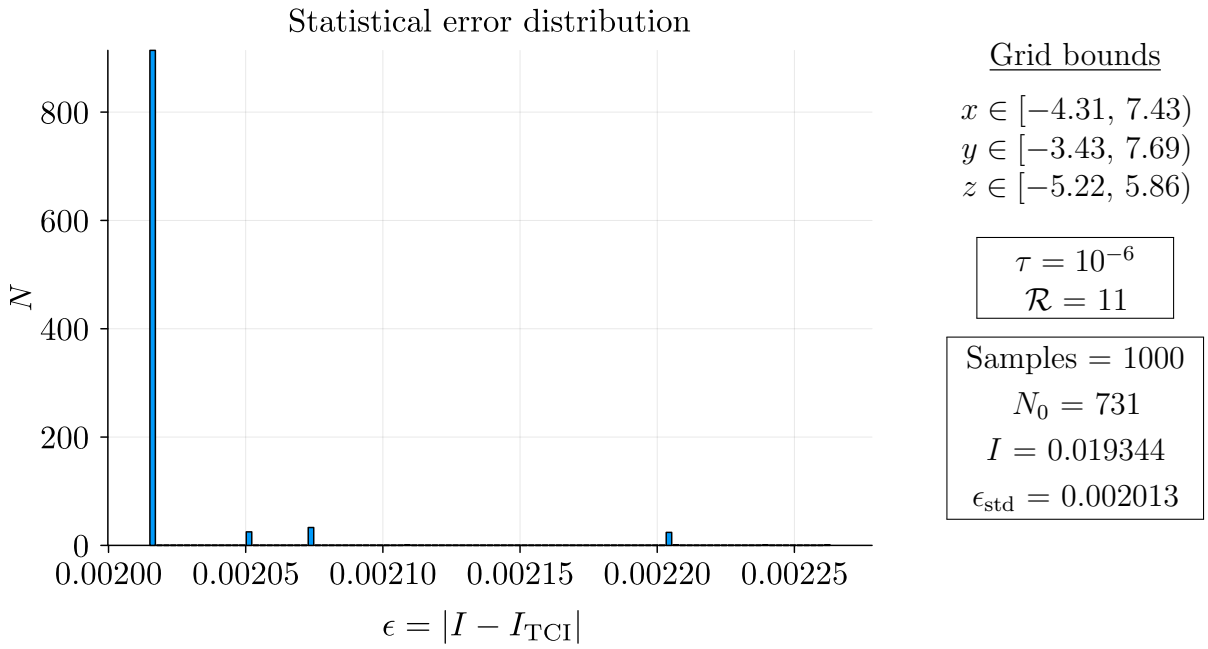


Figure 3.21: Statistical deviation of the TCI algorithm at orbital (1,16,3,17). I is the analytical integral value that should be approximated. ϵ_{std} is the standard error that is found in most cases by the algorithm. N_0 notates the number of runs that returned this value. The first peak summarizes smaller deviations is therefore larger than N_0.

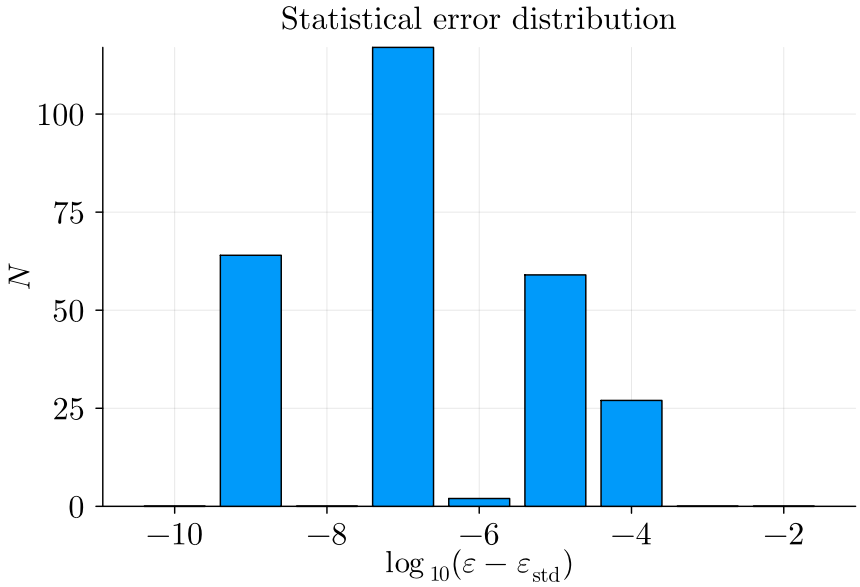


Figure 3.22: Logarithmic plot of deviation $|\epsilon - \epsilon_{\mathrm{std}}|$ of the statistical errors ϵ to the standard error ϵ_{std}. The graph does only include errors that are distinct from ϵ_{std}.

We can clearly see that there is one value of the error that is found in most compressions. We will refer to this value as the standard error. We also see that there are some runs that deviate a lot from the standard error. But Figure 3.21 does not give a good overview over

small differences. We decided to plot the exponents again on a logarithmic scale. For this example all deviating results have a higher absolute value than the most probable one, so the deviations will only be positive.

Figure 3.22 shows the distribution of the statistical error again on logarithmic x scale. We can see that many integration results differ only slightly from the standard error. The random global pivot search has determined only alternative pivots that do not have much influence on the integral value. But in rare cases the algorithm finds bad alternatives for important pivots. These are the cases with an untypical large error.

Concluding form the set of errors as a whole, TCI returns in approximately 30% of the runs a alternative result. For a precise error analysis multiple runs are hence required to find the standard value.

Chapter 4

Transcorelated methods with TCI

As explained in the last chapter, the compression in the orbital indices is not large enough to give the numerical TCI approach an advantage over the analytic computations. One should focus on integrals that require numerical integration and hence, are more likely to be improved by TCI.

During this project, we came in touch with the group of Prof. Dr. Ali Alavi, whose group is working on transcorrelated methods [4]. The bottleneck of the calculation consists of a three-dimensional integration, which is a potential use case of TCI.

4.1 Problem formulation

4.1.1 Theory

The details of the Jastrow factorization are out of focus of this bachelor thesis and can be found in [4]. The following section is a summary of aspects of the paper that are relevant to this thesis. The key idea of the method is to perform a similarity transformation of the second quantised Hamiltonian with the Jastrow factors. Afterwards, high-accuracy solutions of the electronic Schrödinger equation can be computed. The Hamiltonian contains, among others, the term:

$$\widetilde{H} = \frac{1}{2}\sum_{pqrs}(V_{rs}^{pq} - K_{rs}^{pq})\sum_{\sigma\tau} a_{p\sigma}^{\dagger}a_{q\tau}^{\dagger}a_{s\tau}a_{r\sigma} + \dots \quad , \tag{4.1}$$

where $a_{p\sigma}^{\dagger}(ar_{p\sigma})$ are the spin-1/2 creation (annihilation) operators and $V_{rs}^{pq} = \langle pq|r_{12}^{-1}|rs\rangle$ is the two electron term of the Schrödinger equation. $|rs\rangle$ is the combination of two molecule orbitals $\phi_i(\mathbf{r})$ that are again represented by a linear combination of gaussian basis functions $\varphi_i(\mathbf{r})$

$$|rs\rangle = \phi_r(\mathbf{r}_1)\phi_s(\mathbf{r}_2) \tag{4.2}$$

$$\phi_s(\mathbf{r}) = \sum_n \varphi_n(\mathbf{r}) \quad . \tag{4.3}$$

The K tensor contains jastrow factor $u(\mathbf{r}_1, \mathbf{r}_2)$, and consists itself out of three terms:

$$K_{rs}^{pq(1)} = \langle pq| \nabla_1 u(\mathbf{r}_1, \mathbf{r}_2) \cdot \nabla_1 |rs\rangle \tag{4.4}$$

$$K_{rs}^{pq(2)} = \langle pq| \nabla_1^2 u(\mathbf{r}_1, \mathbf{r}_2) |rs\rangle \tag{4.5}$$

$$K_{rs}^{pq(3)} = \langle pq| (\nabla_1 u(\mathbf{r}_1, \mathbf{r}_2))^2 |rs\rangle \ . \tag{4.6}$$

The bottleneck of the Hamiltonian construction consists of the K tensor computation. After integrating $K_{rs}^{pq(2)}$ by parts and adding up all three terms, the tensor evaluates to:

$$K_{rs}^{pq} = \int d\mathbf{r}_2 \ \phi_q(\mathbf{r}_2)\, \phi_s(\mathbf{r}_2) \int d\mathbf{r}_1 \ g_{pr}(\mathbf{r}_1, \mathbf{r}_2) \tag{4.7}$$

$$g_{pr}(\mathbf{r}_1, \mathbf{r}_2) = \nabla_1 u(\mathbf{r}_1, \mathbf{r}_2) \cdot \phi_p(\mathbf{r}_1) \nabla_1 \phi_r(\mathbf{r}_1) + (\nabla_1 u(\mathbf{r}_1, \mathbf{r}_2))^2 \cdot \phi_p(\mathbf{r}_1) \phi_r(\mathbf{r}_1) \ . \tag{4.8}$$

Both integrals can in this case not be evaluated analytically. One could compute the six-dimensional integral by naive quadrature with a repeated three-dimensional grid:

$$K_{rs}^{pq} \approx \sum_{m_1}^{N_{\text{grid}}} \sum_{m_2}^{N_{\text{grid}}} \phi_q(\mathbf{r}_{m2})\, \phi_s(\mathbf{r}_{m2})\, g_{pr}(\mathbf{r}_{m1}, \mathbf{r}_{m2})\, \omega(\mathbf{r}_{m_1})\, \omega(\mathbf{r}_{m_2}) \ . \tag{4.9}$$

where $\omega(\mathbf{r}_{m_i})$ are the integration weights of the grid points. With N_{bas} basis functions, each index p, q, r, s takes on values $(1, \ldots, N_{\text{bas}})$ and the computation of the whole K tensor would come at the cost of $\mathcal{O}(N_{\text{grid}}^2 N_{\text{bas}}^4)$.

However, this cost can be reduced by dividing the summation into two steps. To achieve this, an intermediate object, the xcoulomb matrix, is created for every $\mathbf{r}_2$ and p, r

$$XC_{pr}(\mathbf{r}_2) = \int d\mathbf{r}_1 \ g_{pr}(\mathbf{r}_1, \mathbf{r}_2) \ . \tag{4.10}$$

This requires $\mathcal{O}(N_{\text{grid}}^2 N_{\text{bas}}^2)$ steps. In a second step the summation over $\mathbf{r}_2$ is performed in $\mathcal{O}(N_{\text{grid}} N_{\text{bas}}^4)$ steps. The number of basis functions is with $N_{\text{bas}} \approx \mathcal{O}(10^1)$ rather small, compared to the required grid $N_{\text{grid}} \approx \mathcal{O}(10^4)$. Higher precision of the integral approximation requires even more grid points. Therefore the bottleneck of K tensor evaluation lies in the first step ($\propto N_{\text{grid}}^2$).

4.1.2 Approaches to the bottleneck

With the QTCI algorithm, this bottleneck could possibly be improved. If one achieves to construct for every combination of fixed $\mathbf{r}_2, p, r$ a TT with low bond dimension χ from the the integrand

$$g_{pr}(\mathbf{r}_1, \mathbf{r}_2) \xrightarrow{p,r,\mathbf{r}_2 \text{ fixed}} f(\mathbf{r}_1) \ , \tag{4.11}$$

then the first sum could be performed at a cost of $\mathcal{O}(\chi^2 N_{\text{grid}} \log(N_{\text{grid}}) N_{\text{bas}}^2)$ by simple tensor contraction, which is exponentially cheaper than the integration by quadrature. One should

nevertheless keep in mind that the cheap integration requires the preliminary construction of the TT at the cost of $\mathcal{O}(\chi^3 \log(N_{\text{grid}}))$. Whether TCI can bring an advantage thus depends strongly on the rank of the integrated function.

Note that we will further refer with f to the integrand with fixed orbitals and $\mathbf{r}_2$ and with g to the integrand in both variables. Both f and g refer to the same function but with a different number of free variables. If we do not explicitly write down indices for f or g, this implies they are fixed.

Our task is to determine the rank of $f(\mathbf{r}_1)$ for a significant amount of index combinations $\mathbf{r}_2, q, s$ and compare the number of function evaluations that are required to construct the tensor train, with the number of grid points the quadrature requires to achieve the same result.

4.2 Implementation

Before determining the necessary precision of the TCI compression, we want to comment on a few implementation details and get a feeling for the integrated functions $f(\mathbf{r}_1)$ by looking at their plots.

The complete energy calculation for a molecule with a given basis is usually performed in Fortran with the package *TCHint*. Recently, a Python wrapper was developed, which allows the important Fortran subroutines to be called directly with Python.

The integration by quadrature utilises an atom-centered grid built from Treutler-Ahlrichs radial grids and Lebedev angular grids. Such types of grids are commonly used in the field of DFT and are optimized for integration. They contain comparably few grid points since the knowledge about the basis functions is used to directly choose appropriate coordinates. The grids are available via the python package *PySCF* [19], which then redirects the grid as an input to the Fortran code. Further, the basis functions ϕ_i and their gradients are also evaluated by the package and passed to *TCInt*.

The second property requires *PySCF*, and hence Python, inevitable for this project, even though we do not use the DFT grids. For this research, it is necessary to pass data between three different programming languages: Fortran, Python and Julia, in which our QTCI library is encoded. We make the function $f_{pr}(\mathbf{r}_1, \mathbf{r}_2)$ accessible to Julia by calling the Python wrapper with the Julia package *PyCall*. The Python wrapper then generates the input data with *PySCF* and passes it to Fortran, which then returns the function value of the integrand. This routine currently leads to a costly function evaluation of approx. 1 ms per function call and could be optimized a lot, but since we are only interested in determining the number of function calls that are necessary to construct a TT, we worked with the non-optimal code.

QTCI does not rely on an evenly spaced cartesian grid and can be applied to any curvilinear coordinate system as long as the number of grid points equals a multiple of two. We only need a bijection between a bit vector of arbitrary length and a grid in 3D space. For a cartesian grid, this bijection is given by the quantics method, but for the DFT grids, it is non-trivial. The bijection should further be smooth, i.e. the bits should contain information about the 3D location of the grid point. Just mapping the bits to a random sequence of the

grid points would eliminate the structure of a function and therefore lead to full rank tensors.

Because of this aspect we conducted our analysis with the most simple approach, the cartesian grid. As a result, the TCI algorithm receives no information about the structure of the integrand even though this information is a priori available.

4.2.1 Visualisation of integrand

The transcorrelated method is specialised in computing high-accuracy results. This expensive algorithm is mainly applied to small molecules with only a few basis elements. For our analysis, we used a single beryllium atom with five basis functions.

The integrand f_{pr} has no intuitive shape since it consists of two terms, each containing the derivative of the Jastrow factor $\nabla_1 u(\mathbf{r}_1, \mathbf{r}_2)$. The Jastrow is only available numerically to us, by the program implementation. This makes it hard to get a feeling for the integrand function.

Also, plotting the integrand is not trivial since it is a six-dimensional function. To get at least a small impression of the function, we plotted all orbital combinations at the point $r_2 = [0., 0., 0.]$. Even though the K tensor is not symmetric, a few symmetries can be recognized, as can be seen in table 4.1.

All of the functions resemble the atomic orbitals we have seen earlier. The letters s, p, d denote the atomic shells the functions correspond to. All functions of a given block in the xcoulomb matrix represent the same atomic orbital but with different rotations.

Further we noticed a symmetry in the p rows and columns. The p row at the upper right and the p row at the upper left seem to have the same functional dependency. This holds for both p rows/columns. This means that for these elements we indeed have the symmetry (a,b) = (b,a). This symmetry also holds for the d block. Although a lot of matrix elements seem to be symmetric, we see at the example of (1,2) and (2,1) that the matrix as a whole is asymmetric. By looking at the functions definition (4.8), we expected this asymmetry since the gradiant is only applied one basis function.

	1	2	3	4	5
1	S	S	P		
2	S	S	P		
3	P	P	D		
4					
5					

Table 4.1: Symmetries in the Xcoulomb matrix at the point $\mathbf{r}_2 = [0, 0, 0]$. The letter notates the atomic orbital the function does resemble. Matrix elements with the same label (for example P1) have (almost) the same shape

We plotted all functions of the same shell on the x-axis (where we picked for functions with multiple rotations the one lying on the x-axis). The x range varies for the different shells so all features can be seen. Note the different expansions of the shells. Also, a 3D plot of the orbitals can be seen below the axis plot. The 3D plot shows (as for the Gauss orbitals) all function values in the range $0.2 \cdot f_{\max} \leq f(\mathbf{r}_1) \leq f_{\max}$. The orbitals are plotted on a cubic volume, centred around the kernel.

We present the plots in the order p, s, d since s and d require a whole page.

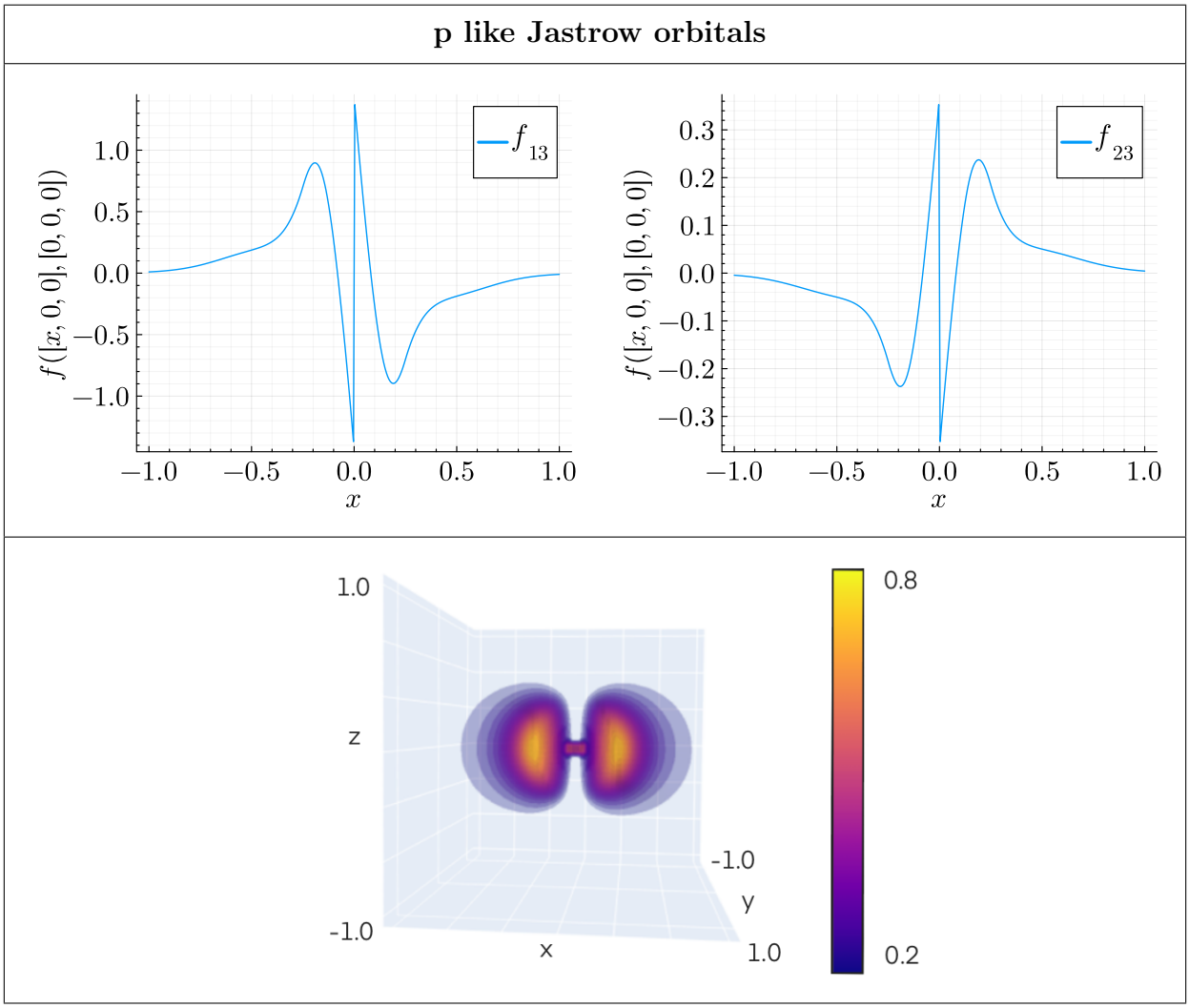


Figure 4.1: Plots of integrand functions $g_{pr}(\mathbf{r}_1, \mathbf{r}_2)$ with p orbital resemblance. The 3D Plot shows orbital (1,3). The maximum is not reached, because for 3D plots, the resolution had to be chosen small.

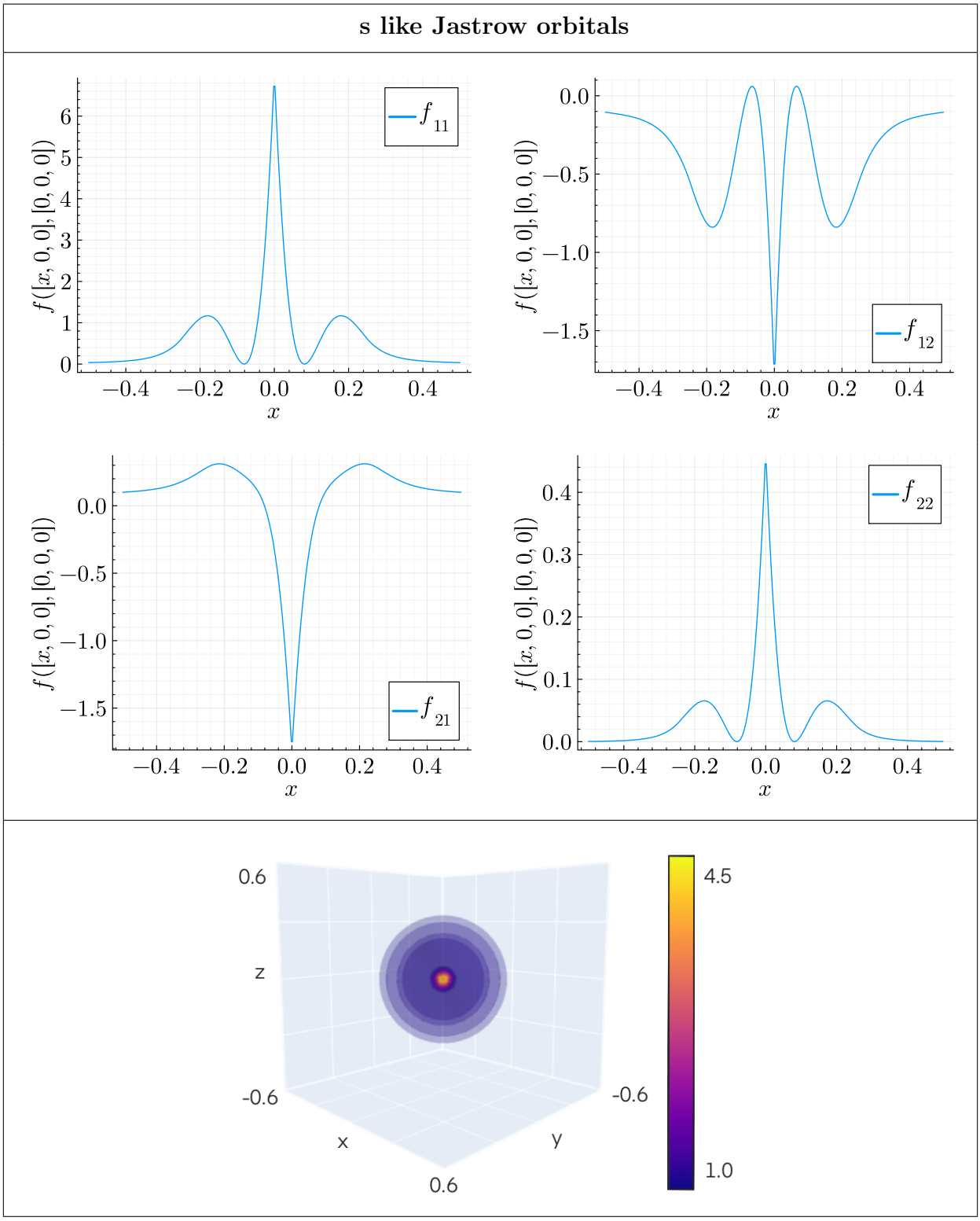


Figure 4.2: Plots of integrand functions $f_{pr}(\mathbf{r}_1)$ with s orbital resemblance at the position $\mathbf{r}_2 = [0,0,0]$. The 3D Plot shows f_{11}. The maximum is not reached, because for 3D plots, the resolution had to be chosen small. One should note the anti-symmetry in (1,2) and (2,1). All shown orbitals are spherical symmetric

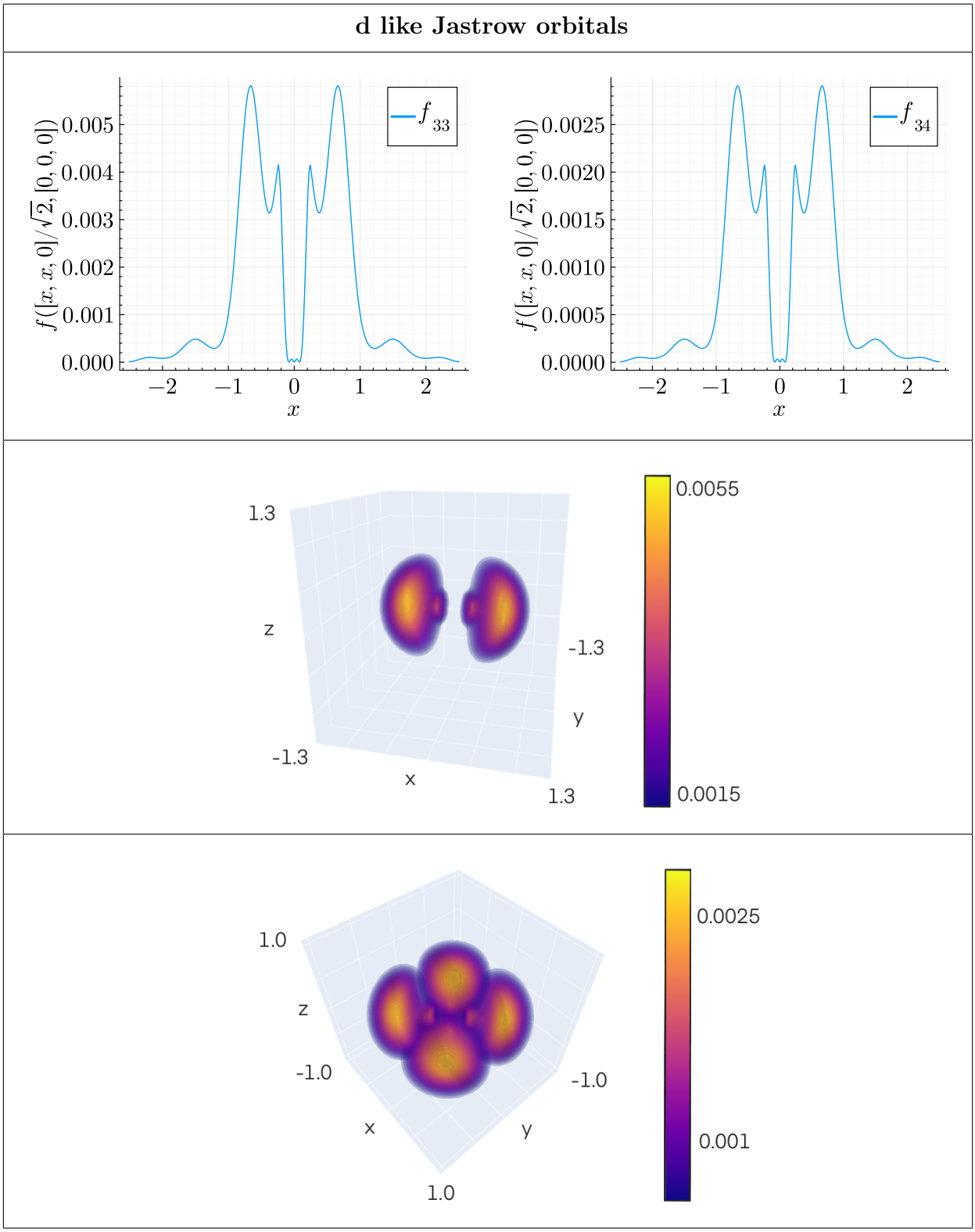


Figure 4.3: Plots of integrand functions $g_{pr}(\mathbf{r}_1, \mathbf{r}_2)$ with d orbital resemblance. The 3D Plots show orbital (3,3) (top) and orbital (3,4) (bottom).

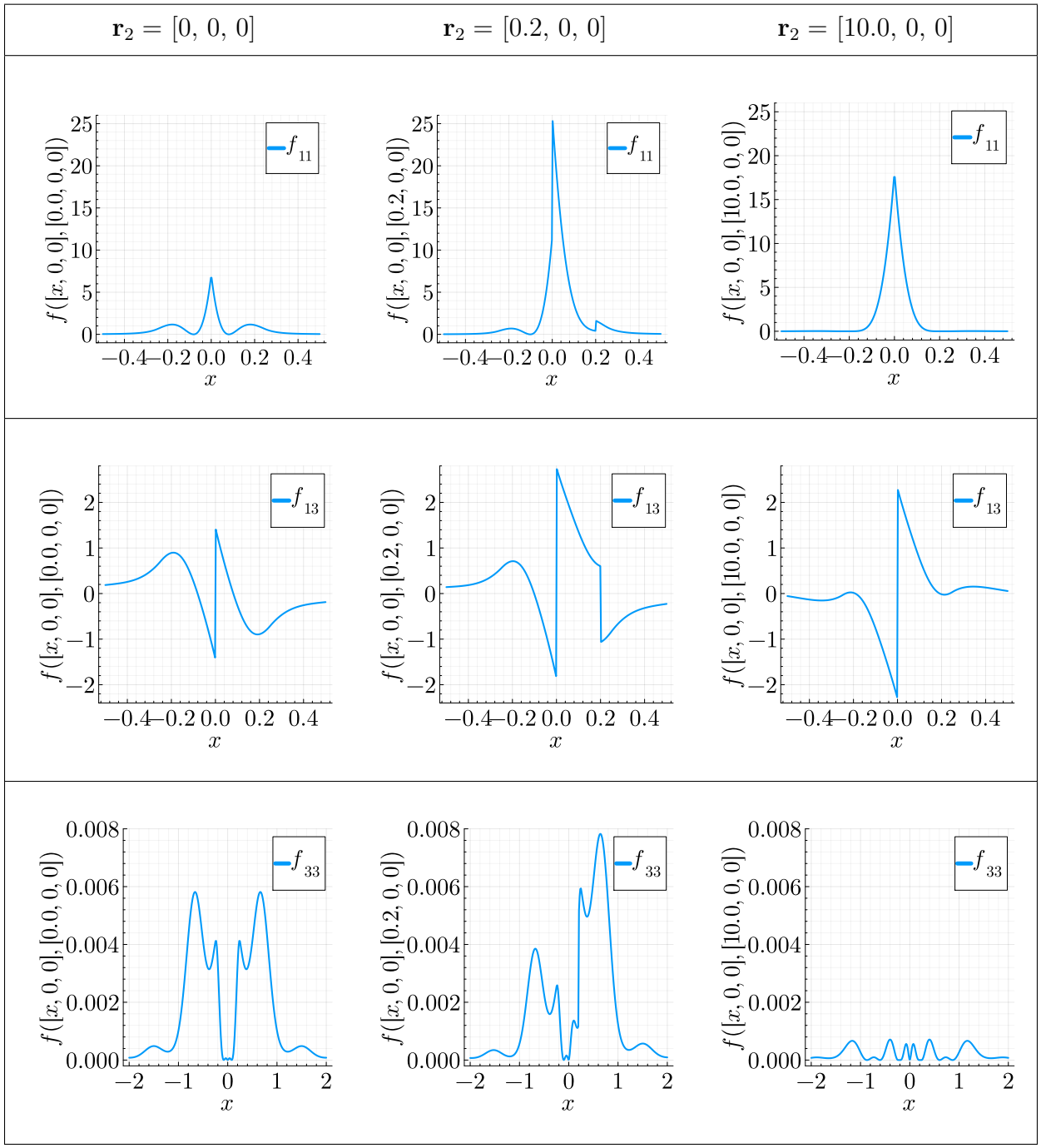


Figure 4.4: X-axis plot of integrand at different positions of $\mathbf{r}_2$

One should keep in mind that all these plots only show the six-dimensional function $g(\mathbf{r}_1, \mathbf{r}_2)$ at the specific point $\mathbf{r}_2 = [0, 0, 0]$. For a different value of $\mathbf{r}_2$ the functions might have a totally different shape. Table 4.4 shows three orbitals during a transition of $\mathbf{r}_2$ on the x-axis. The shape and the integral over the function changes noticeably during the transition. Also, a cusp appears at the position $\mathbf{r}_1 = \mathbf{r}_2$.

For large values of $\mathbf{r}_2$ the graph does not vanish. The function reaches a static form and does not change anymore. We therefore assume that the integrand consists of the static function, we see for large distances between $\mathbf{r}_1$ and $\mathbf{r}_2$, added on an additional function that vanishes for large values of $|\mathbf{r}_1 - \mathbf{r}_2|$. For further statements about the function's properties, we would require an analytic formula, which is currently not available to us.

4.3 Error Analysis

The goal of this chapter is to test if QTCI can be a tool that is able to accelerate the existing Jastrow code. The given code sets a minimum limit on the required precision of the integral in the xcoulomb matrix, which our TCI approach has to fulfil. It is advisable to determine the maximum error before we start with the research so that we will not create tensor trains that cannot reach the required precision or overshoot it. For the error analysis we use the graphical method to search suitable input parameters: first, we find the maximum error that is still acceptable and then determine the necessary grid resolution $\mathcal{R}$ and tolerance τ a TT needs for this accuracy, by plotting relative error for different $\mathcal{R}$ and τ.

4.3.1 Required accuracy

The maximum error can be obtained from the original code. The xcoulomb matrix is here computed by quadrature and its accuracy heavily depends on the number of grid points, which is fixed by a parameter called 'grid level' (simply notated as 'lvl'). The higher the level, the higher the precision of the xcoulomb matrix and hence the end results. The following table shows N_{grid} for our beryllium example at different grid levels.

lvl	N_{grid}
0	1 296
1	7 760
2	18 120
3	22 656
4	53 104
5	80 856

Table 4.2: Grid points for single beryllium atom at different grid levels

Typically, the computations are performed at lvl=2. The exact result can be obtained by setting the grid resolution to an unnecessary high value of lvl=5. Then the absolute and relative errors can be calculated by comparing the lvl=2 and lvl=5 results. Note that since we calculate $XC_{pr}(\mathbf{r}_2)$, all results are only valid for a fixed value of $\mathbf{r}_2$. We can choose an arbitrary point, so we selected $\mathbf{r}_2 = [0, 0, 0]$ for which we already saw the corresponding plots.

Table 4.3 shows the 'literature values' of the computation with lvl=5 and table 4.4 the relative errors $\epsilon = |\text{lvl2} - \text{lvl5}|/\text{lvl5}$. Here, we can see more clearly that no perfect symmetry in the functions is given. Elements where $p = 5$ or $r = 5$ have an absolute value close to zero (compared with the rest of the matrix and the functions maxima). The numerical

p \ r	1	2	3	4	5
1	1.3e-01	-2.7e-01	7.1e-05	7.1e-05	1.4e-17
2	2.3e-01	9.5e-03	1.9e-05	1.9e-05	9.5e-18
3	-4.0e-05	-1.7e-05	8.6e-03	4.1e-09	5.9e-19
4	-4.0e-05	-1.7e-05	4.1e-09	8.6e-03	5.5e-19
5	5.0e-17	-2.6e-17	-3.9e-19	-1.5e-19	8.6e-03

Table 4.3: Literature values (lvl=5) of xcoulomb matrix at $\mathbf{r}_2 = [0,0,0]$

p \ r	1	2	3	4	5
1	6.1e-04	1.6e-04	3.1e-05	3.1e-05	1.8e+00
2	4.0e-05	2.3e-04	1.0e-03	1.0e-03	3.7e-01
3	6.2e-05	5.4e-05	1.5e-04	9.9e-03	8.2e-01
4	6.2e-05	5.4e-05	9.9e-03	1.5e-04	7.1e-01
5	1.5e+00	6.2e-01	1.0e+00	2.1e+00	1.5e-04

Table 4.4: Relative error $\epsilon = |\text{lvl2} - \text{lvl5}|/\text{lvl5}$ of xcoulomb matrix at $\mathbf{r}_2 = [0,0,0]$

p \ r	1	2	3	4	5
1	6.1e-04	1.6e-04	1.9e-04	5.4e+00	7.3e+00
2	3.8e-05	2.1e-04	6.0e-04	9.9e-01	1.3e+00
3	2.1e-04	1.1e-04	2.0e-04	2.0e+00	5.0e+01
4	3.0e-01	1.1e+00	1.5e-01	1.7e-04	1.2e+00
5	1.4e+00	1.1e+00	1.1e+01	3.3e-01	1.7e-04

Table 4.5: Relative error $\epsilon = |\text{lvl2} - \text{lvl5}|/\text{lvl5}$ of xcoulomb matrix at $\mathbf{r}_2 = [0.1,0,0]$

computation requires the elimination of positive and negative values, which is numerically unstable. Although lvl=2 achieves results in the same order of magnitude as lvl=5, this numerical noise leads to very high relative errors of, for example, 210% for the orbitals (5,4). We conclude that for matrix elements that are practically zero, results in the same order of magnitude are sufficient.

All other matrix elements have a relative error in the range $10^{-5} < \epsilon < 10^{-3}$. This limits the required order of magnitude for the relative precision of a xcoulomb matrix element. We recall that all the data only holds for the point $\mathbf{r}_2 = [0, 0, 0]$. To confirm if this result can be generalised, we performed the same process for another point $\mathbf{r}_2 = [0.1, 0, 0]$. Table 4.5 has a remarkable resemblance with 4.4. Two points do not allow us to make a general conclusion but encourage us in our decision to aim at a precision of $\approx 10^{-4}$.

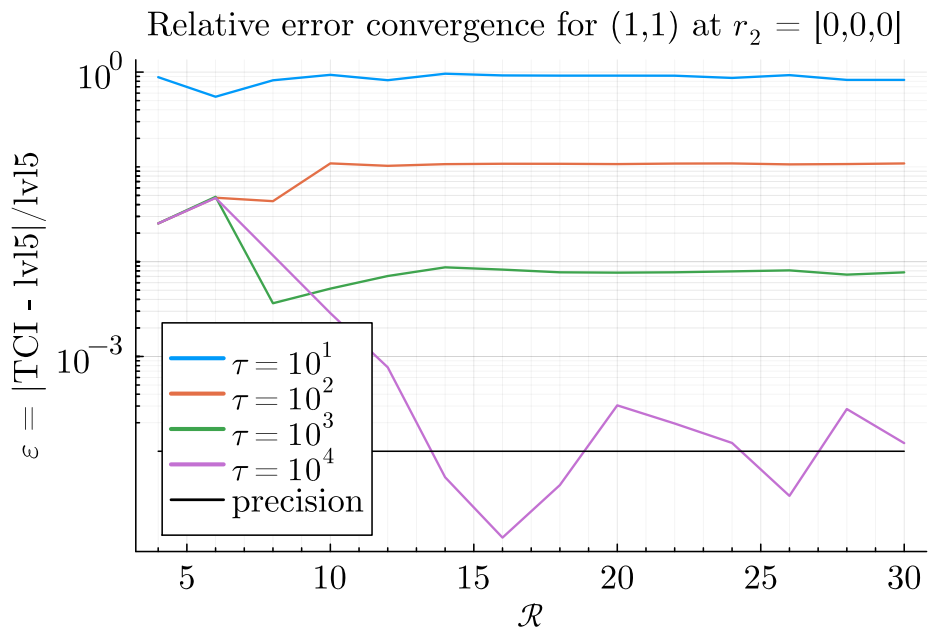


Figure 4.5: Relative error ϵ = (TCI - lvl5)/lvl5 of the TCI approximation of the xcoulomb matrix for the orbital (1,1) at $\mathbf{r}_2 = [0, 0, 0]$. The error is shown in dependency of $\mathcal{R}$ and τ.

4.3.2 TCI error

We now try to reproduce the exact results of lvl=5 as well as possible with TCI by first constructing for fixed $\mathbf{r}_2, p, r$ a tensor train from the integrand $f(\mathbf{r}_1)$ and then summing it over all indices. We chose again $\mathbf{r}_2 = [0, 0, 0]$ and $p, r = (1, 1)$ and plotted the relative error

$$\epsilon = \frac{|\text{TCI} - \text{lvl5}|}{\text{lvl5}} \tag{4.12}$$

As with the Gauss function, one can see that the error converges at a level that depends on the tolerance τ of the approximation. The first tolerance value for which the required precision of $10^{-4} < 10^{-3}$ is achieved is $\tau = 10^{-4}$. For small $\mathcal{R}$, the grid level dominated, while above $\mathcal{R} \approx 13$, the tolerance is the dominating uncertainty.

We also notice that the error varies for $\tau = 10^{-4}$ in a whole order of magnitude, while for all other tolerances the relative error converges against a constant value. This is due to the asymmetry in the error distribution of the tensor train approximation. For some values of $\mathcal{R}$ TCI returns a more symmetric distribution and the errors cancel each other. This leads to a fluctuation of the approximation error and hence the total relative error. This fluctuation becomes more dominant for lower values of τ since more points have a maximal deviation and hence, the asymmetry can be larger.

The variation of the relative error results in relative errors in high $\mathcal{R}$ that are worse than our fixed limit of 10^{-4}. Since this limit was only a guideline this is behavior is not problematic. The variation remains in the acceptable range of $10^{-5} < 10^{-3}$. This is the same range we obtained with the integration by quadrature in table 4.4.

From figure 4.5, we conclude that $\tau = 10^{-4}$ and $\mathcal{R} = 13$ are sufficient to achieve the same precision as integration by quadrature at lvl=2. These parameters could be further optimized since the tolerance can take a continuous value and is not limited to powers of ten. This optimization is too sophisticated at this point and would be the last step before a potential application of the code.

Lower values of τ would theoretically also fulfil the accuracy limit, but these tensor trains do not converge faster in $\mathcal{R}$ than $\tau = 10^{-4}$. In conclusion, one could not reduce $\mathcal{R}$ with more restrictive tolerances but would obtain a higher rank of the TT.

Having determined the input parameters for the QTCI algorithm, we continue to study the properties of the created tensor trains. By comparing the function evaluations of the tensor train creation and the quadrature integration, we can conclude if the use of QTCI may be fruitful for the transcorrelated method.

4.4 Fixed Compression

The question of interest is whether we can reduce the number of function evaluations by using QTCI integration. The key idea of QTCI is to evaluate the function on grid points until the function structure is completely revealed. From that point on, more function evaluations do not contribute new information, and the algorithm stops. This also explains why the tensor ranks saturate at a specific point. If a higher grid resolution would not reveal new structure and make already known peaks or features smoother, no new information is added, and the algorithm does not need more function evaluations to capture the whole structure, which leads to an approximately constant amount of function evaluations for an increasing grid resolution. Figure 4.6 demonstrates this concept

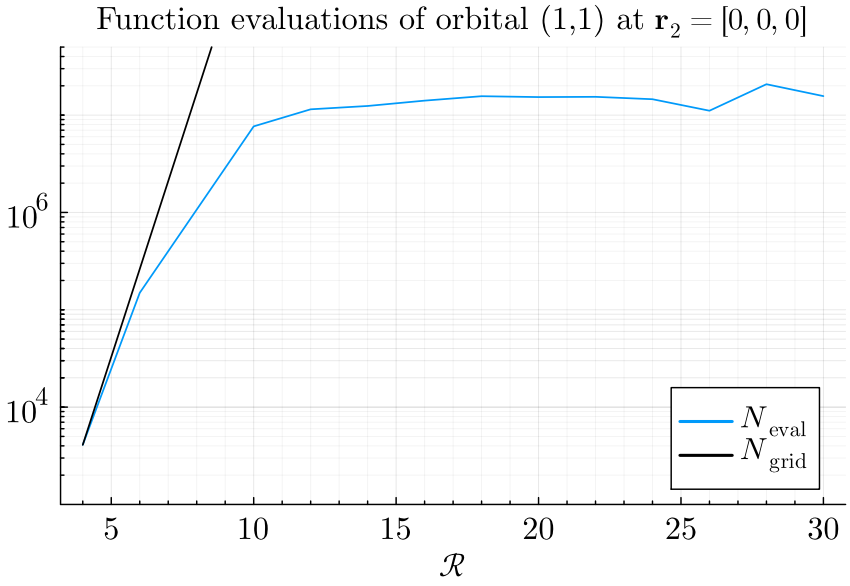
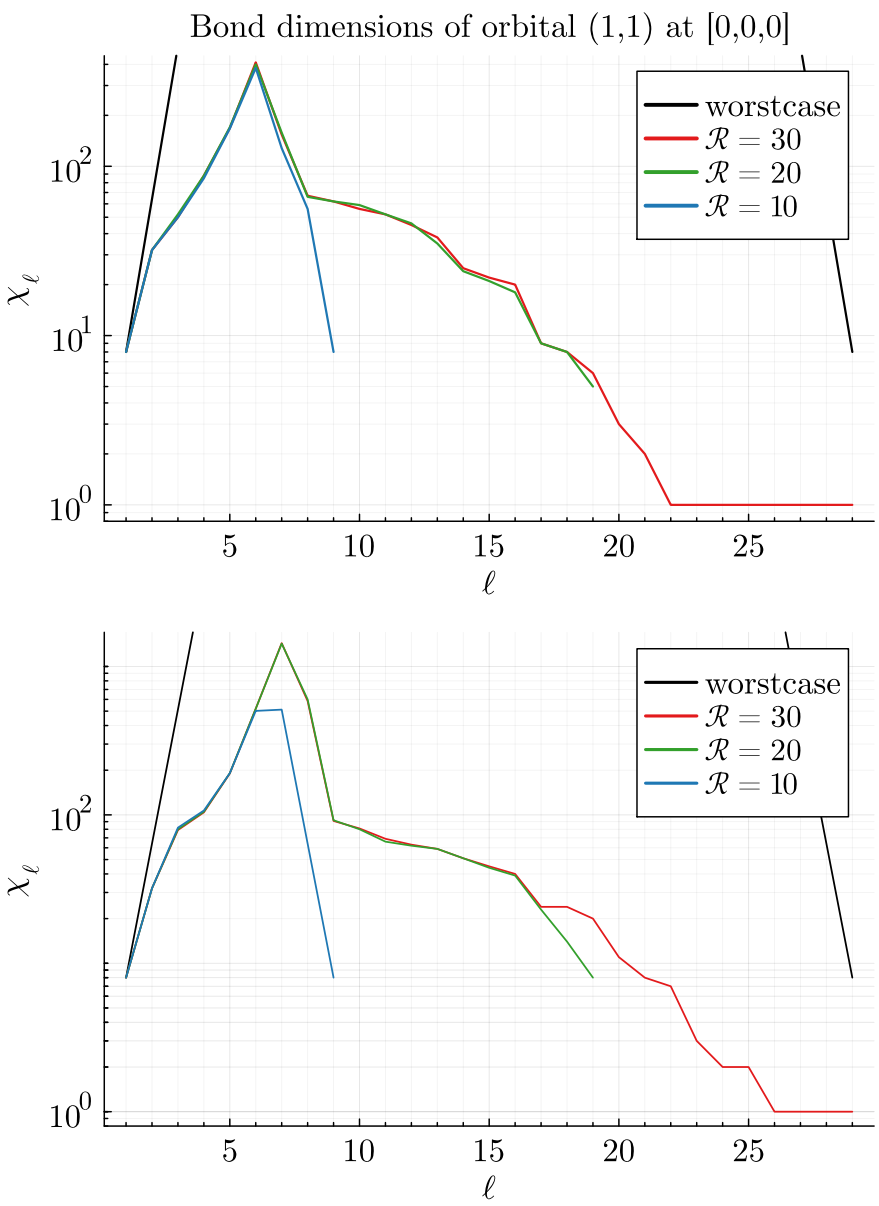


Figure 4.6: Saturation of the function evaluations in $\mathcal{R}$ at the example of (1,1) with $\tau = 10^{-4}$



Tolerance: $\tau = 10^{-4}$

Side length: $a = 3.32$

$\mathcal{R}$	N_{eval}
10	$7.6 \cdot 10^6$
20	$15.3 \cdot 10^6$
30	$15.6 \cdot 10^6$

$\mathcal{R}$	χ	t [h]
10	381	0.39
20	400	0.83
30	411	0.95

Tolerance: $\tau = 10^{-5}$

Side length: $a = 3.32$

$\mathcal{R}$	N_{eval}
10	$11.3 \cdot 10^6$
20	$88.1 \cdot 10^6$
30	$102.8 \cdot 10^6$

$\mathcal{R}$	χ	t [h]
10	512	0.82
20	1426	7.51
30	1436	5.25

Figure 4.7: Bond dimension of the TT at every chain position ℓ for tolerances 10^{-4} and 10^{-3}.

To this end, for functions with a low rank tensor structure, only a fraction of the total grid points must be evaluated. The hope is that the integrand of the xcoulomb matrix has a low enough rank, such that constructing a tensor train requires way fewer function calls than the direct summation in the quadrature.

At first, we consider again our well-known orbital (1,1) at $\mathbf{r}_2 = [0, 0, 0]$. For the required precision of $< 10^{-4}$, we need a tolerance $\tau = 10^{-4}$, as we concluded from our error analysis. We used a cubic grid with a side length of $a = 3.32$ centred around the origin. Figure 4.7 illustrates that the rank of the TT indeed saturates quickly at $\chi \approx 410$ for $\tau = 10^{-4}$, and the information content of the tensor is strongly compressible. Although the $8.1 \cdot 10^6$ function evaluations are few compared to the grid size of $1.0 \cdot 10^9$, the construction of the TT requires

(p, r)	a	χ	$N_{\text{eval}}\,[10^6]$	t [h]	ϵ_{abs}	ϵ_{rel}
(1,2)	2.44	643	21.1	1.6	1.7e-4	6.4e-5
(1,3)	4.96	460	16.8	0.9	7.0e-5	1.0
(2,1)	4.92	210	12.5	0.7	1.3e-5	6.0e-6
(2,2)	1.16	288	12.3	0.9	4.5e-3	0.5
(2,4)	3.8	385	22.5	1.1	2.2e-5	1.3
(3,1)	5.0	503	19.5	1.4	3.8e-5	0.9
(3,3)	5.64	476	50.0	2.6	7.8e-7	9.0e-5
(5,3)	5.64	582	48.7	2.4	2.9e-10	7.6e+8
(5,5)	5.64	479	41.3	2.3	7.7e-7	8.9e-5

Table 4.6: Compression results for further index combinations. The compression was performed on a cubic volume with side length a, centred around the origin.

way more evaluations than a direct summation over an optimized DFT grid, which only requires 18 120 grid points. The hope of our approach was that QTCI would require less function evaluations than the quadrature. We conclude form our data that this is for the investigated example not the case.

We want to check if these results are also valid for other orbitals. Table 4.6 lists the data from the respective compression with $\mathcal{R} = 14$ and $\tau = 10^{-4}$. We want to mention that these compressions were not performed with smoothed functions (see appendix) since the interpolation has to be optimized for every function individually and takes much time. We therefore used the original formula with the zero plateau. The end results are nevertheless not affected by this change since the grid resolution of $\mathcal{R} = 14$ is not high enough to resolve the tiny structures in the centre.

At first we notice that the relative error varies for different orbitals. This originates in the absolute error, which is for all functions approximately of the same order of magnitude. Integral values of almost zero can not be resolved with the given tolerance.

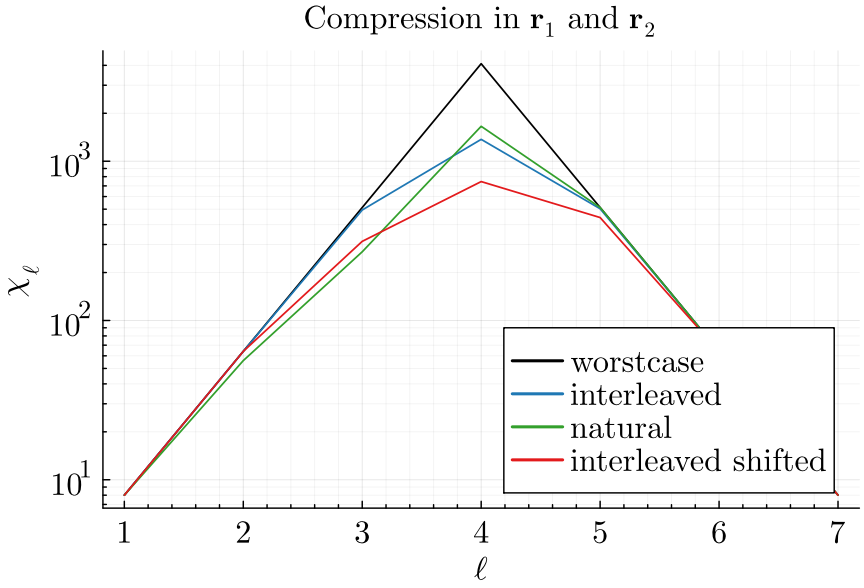
Last but not least, for all examined orbitals, the ranks χ and function evaluations N_{eval} are of the same magnitude. We can conclude from these data that 10 - 40 million function evaluations are required to learn a TT of sufficient accuracy. This confirms our conclusion that TCI cannot outperform the integration by quadrature, which requires only 18 120 function evaluations on an optimized DFT grid.

4.5 Compression in both variables

For completeness, we also tried to compress the integrand in both variables. The orbital variables were again fixed, but instead of a three-dimensional function, we obtain six dimensions

$$f_{\mu\nu}(\mathbf{r}_1, \mathbf{r}_2) \overset{\mu\nu \text{ fixed}}{\longrightarrow} f(\mathbf{r}_1, \mathbf{r}_2) \ . \tag{4.13}$$

Instead of compressing only one grid variable, we search for structure in both of them. Although our main task was to evaluate the xcoulomb matrix fast for every given $\mathbf{r}_2$, it is worth trying a different approach. We couldn't accelerate the computation with constructing a TT for each single $\mathbf{r}_2$. The overhead of N_{grid} repeated TCI compressions is simply too high. By including $\mathbf{r}_2$ into the algorithm, we only need one single TT construction in both variables and can possibly gain an advantage over the repeated overhead. Having a TT in $\mathbf{r}_1$ and $\mathbf{r}_2$, both integrals of the K tensor can be simultaneously evaluated with tensor contraction.



Grid bounds

$x \in [-1.0, 1.0]$
$y \in [-1.0, 1.0]$
$z \in [-1.0, 1.0]$

Tolerance: $\tau = 10^{-3}$

mode	χ	t [h]
worst.	4096	-
sep.	1657	1.3
inter.	1372	2.4
shift.	745	1.8

Figure 4.8: Compression in both variables $\mathbf{r}_1$ and $\mathbf{r}_2$ for the orbital (1,1) with tolerance 10^{-3} and $\mathcal{R} = 4$. Since both variables are resolved with $\mathcal{R} = 4$, we get a tensor of degree $\mathcal{L} = 8$. The labels refer to the used ordering of the indices. The grid is used for $\mathbf{r}_1$ as well as $\mathbf{r}_2$. For 'shifted', the vector $[10^{-6}, 0, 0]$ is added to the grid.

The compression results can be seen in figure 4.8. The three different lines denote different index orderings. The ordering of the indices can have a huge impact on the rank of the resulting tensor trains, so it is worth to try different index combinations. In the first approach with natural ordering, we simply added all bits for $\mathbf{r}_2$ after the bits of $\mathbf{r}_1$, while for the interleaved ordering, we pair all indices that represent the same scale. This concept is depicted graphically in figure 4.9.

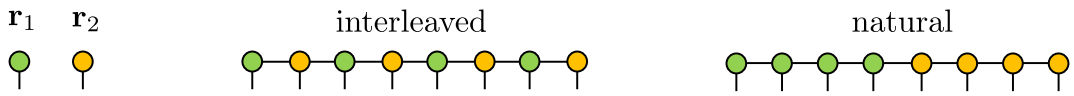


Figure 4.9: Concept of separated and interleaved index ordering

The 'shifted' approach targets the already mentioned problem of the zero plateau (see Appendix). If one uses the same grid for $\mathbf{r}_1$ and $\mathbf{r}_2$, the function will be evaluated multiple times at the problematic point $\mathbf{r}_1 = \mathbf{r}_2$. By shifting the grids relative to each other with the vector $[10^{-6}, 0, 0]$, the variables are never evaluated at the same point, and one obtains the

smooth version of the functions.

All three approaches achieve a small compression. While the natural and the interleaved ordering return approximately the same rank, the shifted approach has the lowest rank. We can conclude that the "zero plateau" problem influences the rank. Although we could achieve a compression, the ranks are far to large to achieve a competitive algorithm. Already at these low parameters $\mathcal{R} = 4$ and tol $= 10^{-3}$, the bond dimensions are in the order of a thousand. We also tried to compress the function with $\mathcal{R} = 5$ but the calculation did not finish during our time limit of 24 hours. Therefore, we expect higher parameters bond dimensions in the order of multiple thousands. For required grid resolution $\mathcal{R} = 14$, this approach will therefore not bring the desired acceleration.

Chapter 5

Conclusion

In this bachelor thesis we worked on two similar but unrelated problems. To this end we will split our conclusion in two sections.

The first problem we investigated was the coulomb matrix in density functional field theory calculations. The main problem of the computation of the matrix elements was the large number of basis functions that are needed for a precise approximation of the molecule orbitals. The usage of gaussian basis functions allows us to compute the integrals analytically, but to not solve the problem of large basis sets.

In our first approach we aimed to compress the tensors in the orbital index and therefore accelerate the summation over all orbitals. Unfortunately, the tensor was not compressible, which implies that many pivots are needed to find a precise approximation. TCI can therefore not result in an acceleration through a simple tensor decomposition.

A further compression of two instead of four orbital indices also returned ranks close to the worstcase. The orbital indices represent functions of various shapes and thus possess a random structure that gives the tensor a high rank. From these results, we can conclude that a compression of orbital indices is not advisable.

We then moved to another approach that makes use of the properties of a gaussian basis. By defining the electron density ρ and the A tensor, we presented another approach to calculating coulomb matrix elements. A compression of A with fixed orbital indices returned tensor trains with a rank of $\mathcal{O}(10^2)$ and thus revealed the low rank structure of the tensor. This is a promising result and shows the potential of the method. Future research may determine the rank of the electron density ρ and decide the applicability of the approach.

Last but not least, we mentioned that the number of basis functions could drastically be reduced by using a non-gaussian basis. The integrals can in this case be computed with QTCI. This requires a basis with low rank elements. By comparing tensor trains of the A tensor and the gaussian basis with fixed orbital indices, we showed that any arbitrary basis with a smooth dependency in $\mathbf{r}$ will likely have such a low rank structure. Finding a suitable basis for this approach will be the task of future research.

In the second part of the thesis, we investigated the xcoulomb matrix in the transcorrelated methods. The QTCI algorithm achieves an enormous compression of the integrand in one

variable. All the integrand functions $f(\mathbf{r}_1)$ indeed have a low rank structure since they saturate at a rank $\chi \approx 400$. The demand for the low rank structure has thus been fulfilled. So why is the method so far away from a practical application?

The problem reduces to mainly one simple point: the integration by quadrature with optimized DFT grids requires very few grid points. Density functional theory is a well established and highly optimized method that has been worked on for multiple decades. While the integral can be computed by quadrature with only 18 120 grid points to a sufficient precision, TCI requires 10-40 million function calls to learn a TT of the same accuracy. To this end, constructing a TT to compute the integral is currently highly inefficient.

TCI tries to choose appropriate grid points (pivots) by learning the structure of an unknown function. In quantum chemistry, the structure of the examined functions is well known and directly used to choose an optimal grid. The grid which TCI tries to find by sweeping through the function tensor is therefore known a priori. By choosing a cartesian grid, we do not use this knowledge, which in the end leads to way more function evaluations than the quadrature would require.

However, there is a way to use the available knowledge in the TCI methods well. As explained earlier, TCI is not limited to the cartesian grid but could use any arbitrary grid as long as N_{grid} equals a power of two to use the bit representation. For a single atom, a DFT grid could easily be implemented since it is in this case equivalent to spherical coordinates with a certain spacing in the radial direction (the density of grid points is higher near the kernel and lower farther outside). With such a grid, TCI could try to find all important pivots among the same points that are used for the quadrature. The use of the available knowledge could be the advantage TCI needs to outperform the quadrature.

But not all problems might be solved with another grid. If one would convert the 18 120 points at lvl=2 into the bit scale, this would be equivalent to a quantics grid with exponent $\mathcal{R} = 4.7$. At such small grid resolutions, the rank of the tensor will probably be close to the worst case of 512. With only so few grid points, almost all of them are important to capture the features of a function, so whether the number of function evaluations can be reduced is questionable.

The overhead of constructing the tensor train makes TCI probably only usable, when enormous grid sizes are required. This would lead us to larger molecules with tens or hundreds of atoms. But the transcorelated methods are only applied for smaller molecules due to their high computation cost. Further, in the case of molecules with more than one atom, a smooth bijection between a non-cartesian grid and the quantics representation is again non-trivial to find.

Concluding all named aspects, TCI will without the use of optimized DFT grids not outperform state of the art methods. Whether or not the usage of curvilinear grids will change this statement may be the topic of future research.

Appendix A

Zero plateau at $\mathbf{r}_1 = \mathbf{r}_2$

The integrand functions $f(\mathbf{r}_1)$ from the *pytchint* package have a property that hinders the integration with TCI. The functions appear to be smooth in $\mathbf{r}_1$ without any cusps or discontinuous points. If we look for example at the orbital (1,1) with $\mathbf{r}_2 = [0, 0, 0]$, we would expect a value unequal zero at $\mathbf{r}_1 = [0, 0, 0]$ by continuity. But if we evaluate the point $\mathbf{r}_1 = \mathbf{r}_2 = [0, 0, 0]$, we obtain a function value of zero. The function not only vanishes at $\mathbf{r}_1 = \mathbf{r}_2$, but also in a small ball with radius $r \approx 10^{-6}$. This unexpected behaviour is a general feature of the functions $f_{ij}(\mathbf{r}_1, \mathbf{r}_2)$. For $\mathbf{r}_1 = \mathbf{r}_2$ we get a function value of zero for all orbital combinations. This property is intended by the package developers but is undesired for our TCI integration approach. Since the functions are never evaluated at the kernel on DFT grids, the zero region was never a problem. Further, we have to mention that it is not intended to neglect the volume at $\mathbf{r}_1 = \mathbf{r}_2$ in the integral. The contribution of this tiny area is so small that it does not appear as an integral error.

If we apply TCI to the function, this property also remains unnoticed until the grid resolution

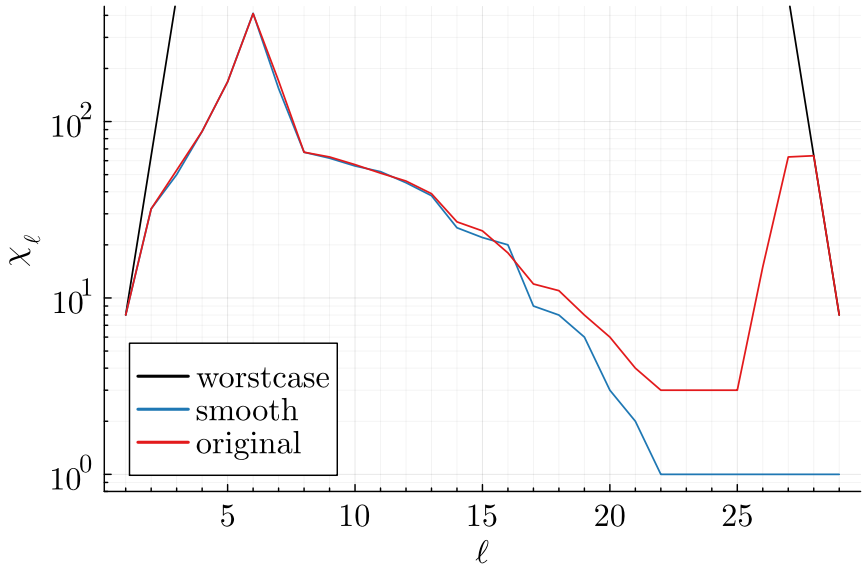


Figure A.1: Rank of TT for orbital (1,1) and $\mathbf{r}_2 = [0, 0, 0]$ with and without a smooth kernel

is high enough to resolve the plateau in the centre. The unexpected discontinuity leads to an increase of the rank. New pivots must be added to include the structure in the approximation. While the new feature does not affect the precision of the integral, the bond dimensions and hence the runtime increase significantly. Figure A.1 shows the orbital (1,1) with and without the discontinuity.

We can simply resolve this problem by interpolating the function in the affected region. The (1,1) orbital has a practical constant slope on the scale of the zero plateau. By extending this slope over the area. This eliminates the new structure, created by the zero values.

Note that this interpolation is only valid for the (1,1) orbital. It is not possible to find a general solution. One has to find for every function a unique suitable extension.

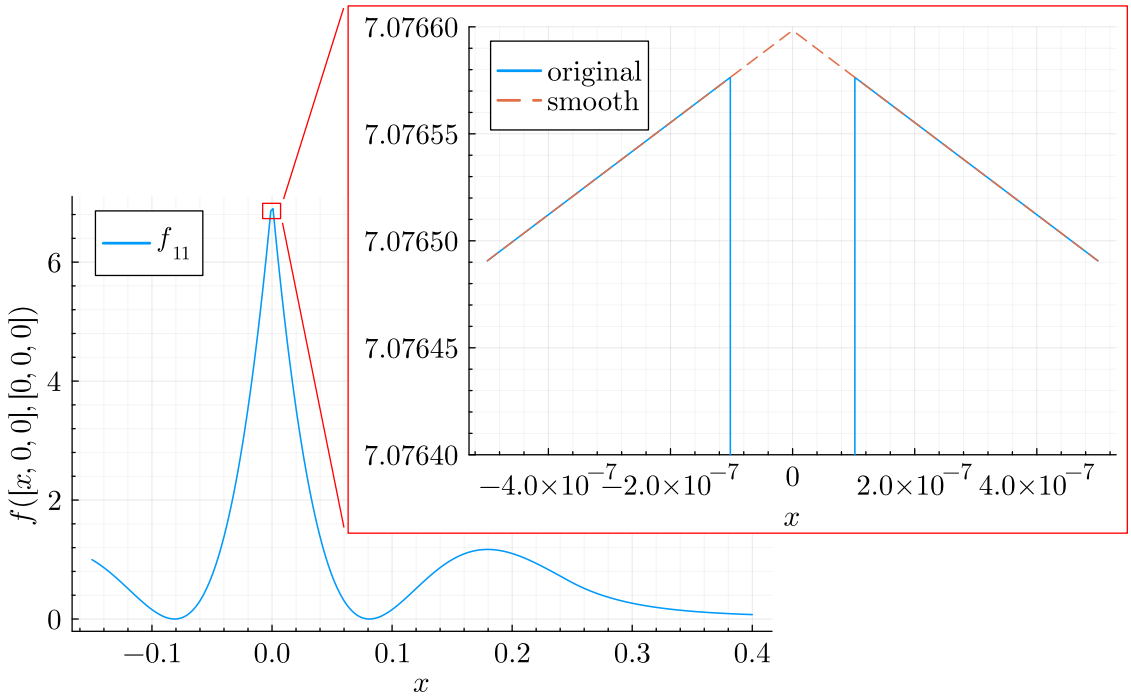


Figure A.2: Zero plateau and smoothed function

Bibliography

[1] Gustavo J. R. Aroeira, Matthew M. Davis, Justin M. Turney, and Henry F. III Schaefer. "Fermi.jl: A Modern Design for Quantum Chemistry". In: *Journal of Chemical Theory and Computation* 18.2 (2022). PMID: 34978451, pp. 677–686. DOI: 10.1021/acs.jctc.1c00719. eprint: https://doi.org/10.1021/acs.jctc.1c00719. URL: https://doi.org/10.1021/acs.jctc.1c00719.

[2] Volker Blum, Ralf Gehrke, Felix Hanke, Paula Havu, Ville Havu, Xinguo Ren, Karsten Reuter, and Matthias Scheffler. "Ab initio molecular simulations with numeric atom-centered orbitals". In: *Computer Physics Communications* 180.11 (2009), pp. 2175–2196. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2009.06.022. URL: https://www.sciencedirect.com/science/article/pii/S0010465509002033.

[3] Asbjörn Manfred Burow. "Methoden zur Beschreibung von chemischen Strukturen beliebiger Dimensionalität mit der Dichtefunktionaltheorie unter periodischen Randbedingungen". PhD thesis. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät I, 2011. DOI: http://dx.doi.org/10.18452/16415.

[4] Aron J. Cohen, Hongjun Luo, Kai Guther, Werner Dobrautz, David P. Tew, and Ali Alavi. "Similarity transformation of the electronic Schrödinger equation via Jastrow factorization". In: *The Journal of Chemical Physics* 151.6 (Aug. 2019). ISSN: 1089-7690. DOI: 10.1063/1.5116024. URL: http://dx.doi.org/10.1063/1.5116024.

[5] David Feller. "The role of databases in support of computational chemistry calculations". In: *J. Comput. Chem.* 17 (1996), pp. 1571–1586. DOI: 10.1002/(SICI)1096-987X(199610)17:13<1571::AID-JCC9>3.0.CO;2-P.

[6] Yuriel Núñez Fernández, Marc K. Ritter, Matthieu Jeannin, Jheng-Wei Li, Thomas Kloss, Thibaud Louvet, Satoshi Terasaki, Olivier Parcollet, Jan von Delft, Hiroshi Shinaoka, and Xavier Waintal. *Learning tensor networks with tensor cross interpolation: new algorithms and libraries*. 2024. arXiv: 2407.02454 [physics.comp-ph]. URL: https://arxiv.org/abs/2407.02454.

[7] W. J. Hehre, R. F. Stewart, and J. A. Pople. "Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals". In: *J. Chem. Phys.* 51 (1969), pp. 2657–2664. DOI: 10.1063/1.1672392.

[8] Christof Holzer. "An improved seminumerical Coulomb and exchange algorithm for properties and excited states in modern density functional theory". In: *The Journal of Chemical Physics* 153.18 (Nov. 2020), p. 184115. ISSN: 0021-9606. DOI: 10.1063/5.0022755. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/

5.0022755/15580656/184115_1_online.pdf. URL: https://doi.org/10.1063/5.0022755.

[9] Hirone Ishida, Natsuki Okada, Shintaro Hoshino, and Hiroshi Shinaoka. *Low-rank quantics tensor train representations of Feynman diagrams for multiorbital electron-phonon models*. 2024. arXiv: 2405.06440 [cond-mat.str-el]. URL: https://arxiv.org/abs/2405.06440.

[10] Nicolas Jolly, Yuriel Núñez Fernández, and Xavier Waintal. *Tensorized orbitals for computational chemistry*. 2023. arXiv: 2308.03508 [cond-mat.str-el]. URL: https://arxiv.org/abs/2308.03508.

[11] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Phys. Rev.* 140 (4A Nov. 1965), A1133–A1138. DOI: 10.1103/PhysRev.140.A1133. URL: https://link.aps.org/doi/10.1103/PhysRev.140.A1133.

[12] Jörg Kussmann, Matthias Beer, and Christian Ochsenfeld. "Linear-scaling self-consistent field methods for large molecules". In: *WIREs Computational Molecular Science* 3.6 (2013), pp. 614–636. DOI: https://doi.org/10.1002/wcms.1138. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1138. URL: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1138.

[13] Benjamin P. Pritchard, Doaa Altarawy, Brett Didier, Tara D. Gibsom, and Theresa L. Windus. "A New Basis Set Exchange: An Open, Up-to-date Resource for the Molecular Sciences Community". In: *J. Chem. Inf. Model.* 59 (2019), pp. 4814–4820. DOI: 10.1021/acs.jcim.9b00725.

[14] Marc K. Ritter, Yuriel Núñez Fernández, Markus Wallerberger, Jan von Delft, Hiroshi Shinaoka, and Xavier Waintal. "Quantics Tensor Cross Interpolation for High-Resolution Parsimonious Representations of Multivariate Functions". In: *Physical Review Letters* 132.5 (Jan. 2024). ISSN: 1079-7114. DOI: 10.1103/physrevlett.132.056501. URL: http://dx.doi.org/10.1103/PhysRevLett.132.056501.

[15] Kohtaroh Sakaue, Hiroshi Shinaoka, and Rihito Sakurai. *Learning tensor trains from noisy functions with application to quantum simulation*. 2024. arXiv: 2405.12730 [quant-ph]. URL: https://arxiv.org/abs/2405.12730.

[16] Rihito Sakurai, Haruto Takahashi, and Koichi Miyamoto. *Learning parameter dependence for Fourier-based option pricing with tensor trains*. 2024. arXiv: 2405.00701 [q-fin.CP]. URL: https://arxiv.org/abs/2405.00701.

[17] Dmitry V. Savostyanov. "Quasioptimality of maximum-volume cross interpolation of tensors". In: *Linear Algebra and its Applications* 458 (2014), pp. 217–244. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2014.06.006. URL: https://www.sciencedirect.com/science/article/pii/S0024379514003711.

[18] Karen L. Schuchardt, Brett T. Didier, Todd Elsethagen, Lisong Sun, Vidhya Gurumoorthi, Jared Chase, Jun Li, and Theresa L. Windus. "Basis Set Exchange: A Community Database for Computational Sciences". In: *J. Chem. Inf. Model.* 47 (2007), pp. 1045–1052. DOI: 10.1021/ci600510j.

[19] Qiming Sun, Timothy C. Berkelbach, Nick S. Blunt, George H. Booth, Sheng Guo, Zhendong Li, Junzi Liu, James D. McClain, Elvira R. Sayfutyarova, Sandeep Sharma, Sebastian Wouters, and Garnet Kin-Lic Chan. "PySCF: the Python-based simulations of chemistry framework". In: *WIREs Computational Molecular Science* 8.1 (2018), e1340. DOI: https://doi.org/10.1002/wcms.1340. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1340. URL: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1340.

[20] Florian Weigend and Reinhart Ahlrichs. "Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: Design and assessment of accuracy". In: *Phys. Chem. Chem. Phys.* 7 (2005), p. 3297. DOI: 10.1039/b508541a.

[21] Florian Weigend, Filipp Furche, and Reinhart Ahlrichs. "Gaussian basis sets of quadruple zeta valence quality for atoms H-Kr". In: *J. Chem. Phys.* 119 (2003), pp. 12753–12762. DOI: 10.1063/1.1627293.

[22] Erika Ye and Nuno F. G. Loureiro. "Quantum-inspired method for solving the Vlasov-Poisson equations". In: *Phys. Rev. E* 106 (3 Sept. 2022), p. 035208. DOI: 10.1103/PhysRevE.106.035208. URL: https://link.aps.org/doi/10.1103/PhysRevE.106.035208.

Disclaimer

I confirm that this thesis type is my own work and I have documented all sources and material used.

Munich, Author