

Sparse-Tensor Methods in Physics

by

Julio J. Candanedo

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2023 by the
Graduate Supervisory Committee:

Oliver Beckstein, Co-Chair
Christian Arenz, Co-Chair
Cynthia Keeler
Onur Erten

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

In this thesis, applications of sparsity, specifically sparse-tensors are motivated in physics. An algorithm is introduced to natively compute sparse-tensor's partial-traces, along with direct implementations in popular python libraries for immediate use. These applications include the infamous exponentially-scaling (with system size) Quantum-Many-Body problems (both Heisenberg/spin-chain-like and Chemical Hamiltonian models). This sparsity aspect is stressed as an important and essential feature in solving many real-world physical problems approximately-and-numerically. These include the original motivation of solving radiation-damage questions for ultrafast light and electron sources.

Alison Parker

ACKNOWLEDGMENTS

I thank my late adviser John C.H. Spence, for encouraging me to do a PhD in physics with him. It was a great experience, especially before the COVID-19 Pandemic, and his hospitably for staying with him briefly in Berkeley during summer 2019. He showed me the importance of the van-Cittert-Zernike and Shannon theorems, and the communications/information background for lensless-imaging, as well as motivating the importance of radiation-damage for information loss.

I also greatly thank Oliver for taking me under his wing, after the passing of my adviser his extreme patience with me during this time. Araceli, Physics Department for funding. I thank Christian (co-adviser), and committee Onur and Cynthia.

In addition, to John and Oliver, I'm grateful for direct influences from David-Gurevich (for teaching me chess), Matthias (for encouraging me to explore pure mathematics), Maulik (teaching me the path-integral), and Dmitri (for bringing to light the nonadiabatic nature of radiation-less transitions). Also to Maxim and Sergei for helping me as an undergraduate.

While I stayed in Uppsala, Sweden, I thank early support from Calle, Nic, and others at Ångströmlaboratoriet, for a friendly and supportive atmosphere. Including for securing a STINT-grant for me. I thank Ken, for his hospitably in Montréal, Québec, to escape the summer heat, only to stay until winter! During an excellent school, TDDFT2019, at Rutgers I met a good-friend Daniel, whom taught me much of computational-chemistry, and I am eternally grateful for this collaboration. And his hospitably while I visited Memphis in 2021.

After, the passing of my adviser, I taught 1st year physics, and I thank Darya,

for her support during this time, including letting me modify curriculum to include python virtual physics labs. I would also like to thank my students, whom I taught over the years, including Esme, Siandelle, Tegan, Sepideh, Neil, Sophia, Cedrick, and Eurico.

I thank fellow friends whom are also PhD inmates, I mean students; Tom, Adam, Megan, Armin, Billy, Hanan, Sanjana, Shujie, Chenou, Ricky, Sean, Ian, and even the other Ian, Reza, Andrew and the other Andrew. I think most of us have escaped now. Also non-PhD student friends: Nadia, Stella, Derek, Joe, Chufeng, Rick, Kevin, and Ganesh, I enjoyed our conversations. And last but not least: Araceli, Gio, Kyle, Pedro, Lauren, and Russell.

Most importantly, I could not have done this without the support of my immediate family: Mother, Father, and Sister.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	xii
CHAPTER	
1 CASE FOR SPARSITY	1
1.1 Natural World and Information.....	1
1.2 Naturalness = Sparsity?	2
1.3 Occam’s Razor & the Wave-Function-Collapse	2
1.4 Network Science.....	4
1.5 Hypergraphs & Simplicial-Complexes	6
1.6 Finite-Element	8
1.7 Bridging theory & High-Performance-Computing	9
1.7.1 NUMPY’s einsum Function	10
1.7.2 Strassen Theory & AlphaTensor	11
1.7.3 Tensor Processing Units.....	11
1.8 Sparse-Tensor Research	12
1.9 Physical Problems	12
1.9.1 Nonlinear Dynamics	12
1.9.2 Quantum Many Body Problems	13
1.9.3 Radiation Damage.....	14
1.10 Organization and Goals	14
2 SINGULAR-VALUE & SCHMIDT DECOMPOSITION	16
2.1 How to Define a Wavefunction?	16

CHAPTER	Page
2.2 What is a Tensor?	17
2.2.1 What is a Sparse-Tensor?	20
2.3 The Tensor-Network	21
2.3.1 Tensor Diagrammatic Representation	21
2.3.2 Singular-Value-Decomposition	23
2.3.3 Tensor Decompositions	24
2.3.4 External-index Surfaces	25
2.4 Tensor-Network as a Sparse-Tensor	26
2.5 Unstructured Sparse-Arrays	26
2.5.1 Well-ordering	29
2.6 Tensor Operations	31
2.6.1 Composition, Sub-indexing, and Permutations	31
2.6.2 Array-labeling	31
2.6.3 Cartesian-product	32
2.6.4 Intra-intersection/Slice	35
2.7 Duality: Unstructured-sparse & Tensor-network Sparse Tensors ..	36
2.8 Tensor Types	36
3 SPARSE-PARTIAL-TRACING	38
3.1 Partial-Trace	38
3.2 Ordered Network Structure	39
3.2.1 List of Tensors	40
3.3 List-of-Labels Trace Representation	41

CHAPTER	Page
3.3.1 Network CONtraction (ncon).....	42
3.4 Hyper-trace/Hyper-contraction.....	42
3.4.1 Slicing Hypergraph.....	43
3.4.2 Sparse-tensor Representation.....	44
3.4.3 Simplicial-complex?.....	45
3.5 Dense-Sparse Partial-Trace.....	46
3.6 Sparse-Dense Network.....	47
3.6.1 Dense-Sparse Networks Continued.....	49
3.7 Sparse-Sparse Partial-Trace (algorithm 1).....	50
3.7.1 Sparse Reshaping.....	50
3.7.2 Modified Shape.....	52
3.7.3 Tuples-to-Integers.....	53
3.7.4 Integers-to-Tuples.....	53
3.7.5 Tuples-to-Tuples.....	54
3.7.6 Binary Summation (bisum).....	54
3.8 Sparse-Sparse Partial-Trace (Algorithm 2).....	55
3.8.1 Direct-Intersection.....	55
3.8.2 Shoelace.....	57
3.8.3 Multi-Intersection (Hyperedge).....	58
3.9 Surjective-map.....	60
3.10 A Word of Caution!.....	62
3.11 Output Considerations.....	62

CHAPTER	Page
3.12 Partial Partial-Trace	63
4 SPARSITY IN TENSOR-NETWORKS	64
4.1 The Heisenberg Model	64
4.1.1 Exchange Interaction	66
4.1.2 2-site Heisenberg Example	67
4.2 Generalized Heisenberg-model	68
4.2.1 Generalization & Comments	69
4.2.2 $[\mathfrak{su}(N)_1] \cong [[\mathfrak{su}(2)_{N-1}]]$	70
4.2.3 Let's Consider $N = 3$ and $s = 1$	71
4.2.4 Low-energy Excitations	71
4.3 NN-Factorization	72
4.4 Tensor Operator Element	75
4.5 Fundamental-Representation of $\mathfrak{su}(N)$	78
4.5.1 Complex Representation	78
4.5.2 Adjoint Representation	79
4.5.3 Jordan-Wigner Transformation	80
4.6 Exact-Diagonalization	80
4.6.1 Explicit-Matrix Diagonalization	81
4.6.2 Iterative Diagonalization	82
4.6.3 MPO-ED (linear-chain)	82
4.6.4 Generic-network ED	84
4.7 Approximations to Ψ	86

CHAPTER	Page
4.8 Density-Matrix-Renormalization-Group	88
4.8.1 State	89
4.8.2 The Energy	89
4.8.3 Tensor Network Derivative & Optimization	89
4.8.4 Update & Sweep	90
4.8.5 Generalized-DMRG	91
4.9 Mean-Field Theory	91
4.10 Sparsity of Tensor-Network Pieces	92
4.10.1 Sparsity of H	92
4.10.2 Sparsity of W	93
4.10.3 Sparsity of Ψ	93
4.11 Sparsity & Entanglement-Entropy	95
4.12 Entanglement-Entropy	96
4.12.1 Bond-Entropy	97
4.12.2 Block-Entropy	97
4.12.3 Central-Charge	98
4.13 Procedure	98
4.14 Results	99
4.14.1 $c(m)$ Calculation	100
4.14.2 $m(L)$ Calculation	102
4.14.3 Conjecture	103
4.14.4 Symmetry imposed Ψ_{MPS}	104

CHAPTER	Page
4.15 Error Fidelity	105
5 SPARSITY IN CHEMICAL SYSTEMS	108
5.1 Orbital Interaction	109
5.2 Full Configuration-Interaction	110
5.3 Active-Space Configuration-Interaction.....	110
5.3.1 Multi-Configuration-SCF (MCSCF).....	112
6 MOLECULAR DYNAMICS	113
6.1 Freshman's Dream	113
6.2 Atomic Dynamics	115
6.2.1 Weak Molecular Interactions	115
6.2.2 Strong Molecular Interaction	117
6.2.3 Computational Methods	118
6.3 Potential Energy Surface.....	119
6.3.1 Black-Box Decomposition of the Potential Energy Surface .	120
6.3.2 Atomic Cluster Expansion & Simplicial-Decomposition ...	121
6.3.3 PES Hopping	124
6.4 Cell-list and NN-interactions	124
6.5 Classical Molecular Dynamics Algorithm	127
6.5.1 $\sim N^2$ Pair-force Calculation	127
6.5.2 Sparse Pair-force Calculation	128
6.5.3 For Many-body Force Calculation.....	129
6.5.4 An Example Run	129

CHAPTER	Page
6.6 Application in Radiation-Damage.....	130
6.6.1 The Exciton & Excimer Model	132
6.7 Real-Time Molecular-Dynamics	134
6.7.1 Real-Time H_2^+ Coherent Excitation	135
6.7.2 Ehrenfest Dynamics	136
6.8 Real-Time Radiation-Damage	137
7 CONCLUSION	141
REFERENCES	144
APPENDIX	
A SPARSE-EINSUM BISUM [†]	156
B CHEMRXIV PAPER	162
C JCP PAPER	190
D JCP PERMISSION	196

LIST OF FIGURES

Figure		Page
1.	Out of All Possible Images, Given a Pixel-dimension, Sparse/natural Images form an Infinitesimally-small Subset. (Note that TV-static Should in the Figure Should have 256 Colors)	3
2.	This is a Table of Various Animals and Their Neural-network Density.	6
3.	Six Simplicial-complexes Emulating the Sphere, S^2 , with Increasing Simplex-count. Note that the Highest-resolution Representation of the Sphere is the Sparsest (this is Because out of All the Possible Connections with More Nodes Fewer are Used). Note in the Limit of the Perfectly-continuous-mathematical-sphere we Obtain the Infinite-sparse Limit. Number of Total Possible Edges is: $\binom{n}{2} \sim n^2$, while Number of Presenting Edges is $6n$. While Number of Triplets are $\binom{n}{3} \sim n^3$, and Number of Faces are $\sim n$	7
4.	Various Graphical Representations of Tensors/Arrays.	21
5.	The Decomposition of a Sparse-tensor (Previously Known as a Dense-tensor) into Network of Dense-tensors.	25
6.	The Decomposition of the Sparse-array into a Shape, Array-of-tuples, and Data Arrays.....	29
7.	Three Examples of Tensor-index Hypergraphs.	43
8.	Three Examples of Slicing-prescription Hypergraphs.	44

Figure	Page
9. This Figure Shows an Example of a Direct-intersection (the Collection of All the Edges) Between Two Arrays' (Red and Blue) Elements (<i>i.e.</i> the Nodes), Represented by a Sum of Disjoint Complete Bipartite-graphs.	57
10. (left) An Example for the Lace Algorithm on 4 Lists/Arrays. (Right) Showing the Contraction-hyperedge and the Searching Path (Shown to the Left) Shown by the Solid Line.	59
11. This Figure Shows an Example of the Complete Operation Between Two Surjective-maps, two Ordering, and a Direct-intersection Map.	60
12. Shown is a Log-plot Showing the Bond-dimension, m , of an MPS. The Filled-in Portion, Shows the MPS is Modelling Compared to the Entire ED-wavefunction-tensor. This is Plotted For a Region of 40 Sites. Note Although the Non-filled in Region Appears Smaller, Its In-fact Exponentially Larger Than the Filled in Portion Due to the Vertical Axis-scale.	94
13. Examples of the Entanglement-Entropy of the $\mathfrak{su}(2)_1$ Model (Left Plot) for $m = 60$ (Lower Curve) and $m = 200$ (Upper Curve) Respectively, and Their Difference (Right Plot).	100
14. Examples of the Entanglement-entropy as a Function of the Bipartite Divider ℓ on a 1-dimensional $L = 120$ Site Lattice Spin-chain. The Upper Curve Shows the Results for $[\mathfrak{su}(4)_1]$, the Middle Curve Shows Results for $[\mathfrak{su}(3)_1]$, and the Lower Curve Shows the Result For $[\mathfrak{su}(2)_1]$	101

Figure	Page
15. This Plot Shows the Central-charge as Fitted to DMRG Data, the MPS Over $L = 60$ Spin-sites. For $\mathfrak{su}(2)_1$ (Blue) Which Converges to $c = 1$ as Expected With Bond-dimension $m \approx 20$. $\mathfrak{su}(3)_1$ (Orange) Approaching $c = 2$, Also as Expected, with Bond-dimension Roughly $m \approx 100$. And $\mathfrak{su}(4)_1$ (Green), With the Central-charge Increasing Not Converging to the Expected $c = 3$ Value Yet.	101
16. This Plot Shows the Constant-entropy Term, $C1$, in Eq. 4.8 Least-squares-fitted to Numerical DMRG Data (the MPS). For Systems With $L = 60$ Spin-site Chains in $\mathfrak{su}(2)_1$, $\mathfrak{su}(3)_1$, and $\mathfrak{su}(4)_1$ Theories.	102
17. The “Converged” (Such That the Central-charge Does Not Change <i>Much</i>) Bond-dimension m (MPS in DMRG) For A XXX Spin-chain of Length L . DMRG Data is Fitted With Function $m = b * L^a + c$, With Least-squares Parameters: $a = 0.51001911$, $b = 5.0420591$, $c = -13.39164775$	103
18. The Converged Energy \mathcal{E} (MPS in DMRG) For A XXX Spin-chain of Length L . DMRG Data is Fitted With Function $m = a * L + b$, With Least-squares Parameters: $a = -0.3183050050730525$, $b = 0.17962704375639882$	104
19. When Constraining For Symmetry, Within Each m^2 -sub-matrix May Be Reduced to Block-diagonal Form.	105
20. Shown is the Fidelity of the MPS State (for Sparsities) when Compared to the EDS, for $L = 10, 14, 22$ Lattice Sites, with Blue, Orange, and Green Dots Respectively.	107

Figure	Page
21. Shown is the energy difference of the EDS and MPS (for Various Sparsities) states, for $L = 10, 14, 22$ Lattice Sites, with Blue, Orange, and Green Dots Respectively.	107
22. Above is a Cartoon Depiction of a Hypothetical Potential Energy Surface (PES), the Vertical-axis Being Energy, While the Horizontal-axis Being a Reaction Coordinate (Defined by Nuclei Distances). In this Figure, Four Curves are Shown: Ground-State Singlet (Solid Curve, S_0), its Associated Triplet State (Dotted Curve, T_0), An Excited Singlet State (S_1), and another triplet state (dashed curve, T_1). The Gray Shading Region, Bounded By the Uppermost Curve is the Ionization/Continuum Threshold.....	120
23. Plotted is the Energy-difference (When Compared to the First Time-step) in Time. Ideal Situation is a Perfectly Horizontal Line. This is On 1 CPU For A 10 Minute Simulation of Approximately 4300 Argon Atoms for 5000 Time-steps. The Algorithm is the Aforementioned FIT-MD, With Linear-interpolation.	130
24. Plotted is the Radial-Distribution-Function (RDF) of the 1st Time-step (Blue) and Last (Orange), of the Same Simulation in Fig. 23. The Initial Stage is a Perfect Crystal, and the Last Instance is a Slightly Thermalized Sample.	131

Figure	Page
25. This Figure Shows the EELS spectrum of RGSs as Published in Klein & Venebles, for Neon (Red), Argon (Cyan), Krypton (Yellow), and Xenon (Violet). Vertical-axis is Re-scaled With-respect-to Each RGS Spectra (Highest Peak of Each is Rescaled to '1').	133
26. Above is the PES of H_2^+ /HF/STO-3G Basis, Showing Both Electronic States, σ and σ^*	135
27. Above are Shown the Irradiation By the Electric-field Shown (Top) of the 2-state H_2^+ , By Real-Time Runge-Kutta (An Approximation for the Unitary Integration) Integration (bottom).	139
28. The Top Plot Shows the Interatomic Distance of the Same Cation, Displaying the Dynamics Under Various Pulse Intensities. This Clearly Shows an Increase to the Molecules Vibrational Period, Until an Infinite Period is Achieved at the Asymptotic Limit Leading to Dissociation. While the Bottom Plot Displays a Function of Vibrational Period to the Exposure Electric Field Strength.	140
29. This plot shows a timing comparison between the <code>torch.einsum</code> (solid line, averaged over 2 samples) function, the sparse-dense <code>bisum</code> trace (connected dots, averaged over 5 samples) function, and finally the sparse-sparse <code>bisum</code> tensor contraction: $A_{qjwhkrjd}B_{krqljdmn}$ (each tensor of shape $\left(\begin{matrix} 14 & 14 & 14 & 14 & 14 & 14 & 14 & 14 \end{matrix} \right)$) on a single CPU.	161

Chapter 1

CASE FOR SPARSITY

1.1 Natural World and Information

In physics, our objective is to understand the natural world. We must first label objects and phenomena. While the universe doesn't provide a measuring stick at all locations, that doesn't hinder us from using units and coordinates. Through these units, we can associate a collection of numerical values (arrays) with physical phenomena. Furthermore, it is well-known that our human perceptions can be misleading and limited. Therefore, we rely on data obtained from various instruments and sensors, which output data in the form of numerical arrays. Examples include audio/sound, digital images, spreadsheets of labeled data, sport statistics, or financial information. In physics, we work with trajectories, position vectors, state vectors, diffraction patterns, spectra, and more.

On the flip side, computers have proven to be an invaluable tool in science. In computational simulations, we often need to organize real-world data into multi-dimensional arrays. For instance, a 2-dimensional image can be transformed into a 3-dimensional array comprising three 2-dimensional pixel arrays (one for each color), with each numerical value representing the intensity of that color. Many array manipulation techniques can be reduced to the manipulation of indices, such as linear algebraic techniques (matrix multiplication). In the modern information age, we

witness the digitization of nature, which we refer to as 'data.' Our objective is to introduce algorithms to manipulate this natural, physical data. It's important to note that although reality is dense, our understanding of it is achieved through sparsity.

1.2 Naturalness = Sparsity?

As we shall show, real-world data is anything but random. Let's introduce a Gedankenexperiment/thought-experiment. Suppose we wish to determine how many images may be formed from 1024×768 pixels (old XGA/standard-resolution circa 2000) with 2 xor 256 colors? In total there are 786,432, pixels so the number of images is $2^{786,432}$ or $256^{786,432}$. Either number is absolutely enormous. Figure 1 attempts to illustrate that the vast majority possible images (configurations of those pixels) are reminiscent of *tv-static* (infinite temperature Ising-model), we call this the dense-state.

1.3 Occam's Razor & the Wave-Function-Collapse

In understanding the physical world, perhaps the most important principle is *Occam's Razor*. This widely known principle¹ is named after William of Ockham, a 14th-century English scholar. Occam's Razor is often paraphrased as "Entia non sunt multiplicanda praeter necessitatem" or "More things should not be used than are

¹However, it's likely that this principle has been around for much longer.

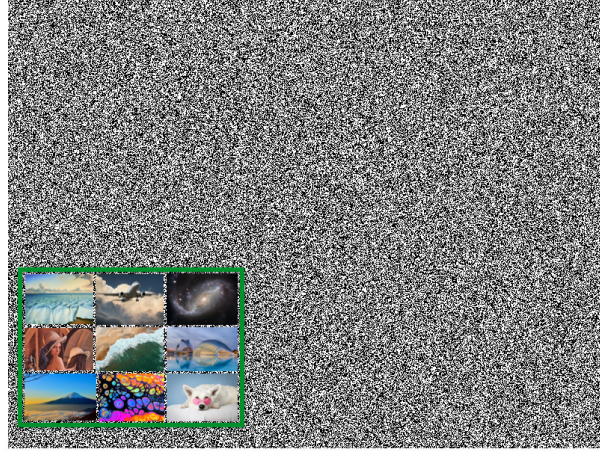
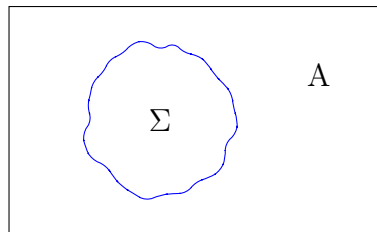


Figure 1. Out of All Possible Images, Given a Pixel-dimension, Sparse/natural Images form an Infinitesimally-small Subset. (Note that TV-static Should in the Figure Should have 256 Colors)

necessary”, or as *lex parsimoniae* or the law of parsimony. For us, this is the guiding philosophy of sparse methods. We aim to solve problems with unknowns, potentially in an underdetermined fashion, for the sparsest possible solution. Let’s consider an example system:



Suppose we have complete knowledge of the state of A , i.e., ψ_A , with absolute certainty, meaning that the entropy $S = 0$. This state encounters a completely unknown region, Σ , while the systems interact with the boundary of ∂A . If we can make a few assumptions about Σ , for example, if it shares similar characteristics with

A , then the entropy of Σ is proportional to the number of all possible states² that match the boundary condition given by ψ_A , i.e. $S \propto \log N$. However, Occam's razor guides us in determining which states are the most important. It leads us to favor states with the least complexity, the sparest ones. While there might be many states ψ_Σ with similar sparsity, we can reduce our uncertainty or entropy by the logarithm of the number of these important states.

1.4 Network Science

Suppose we have a system consisting of N particles, bodies, or systems, and we can characterize their interactions by their proximity to each other. This proximity is represented by an edge, linking two particles together. The study of edges and nodes together is studied mathematically, and is referred to as *Graph Theory*, see Bollobás 1998. In computer-science related fields this is called *network-science*, see El Gamal and Kim 2011. We will make a subtle distinction between them; graphs are abstract-mathematical-objects (with *sets*), while networks are realized graphs (with *ordered-sets*/arrays/lists). Networks carry an *enumeration* for their nodes and hence edges³. This enumeration allows for concrete algorithm description, e.g. a

²There is an interesting program by Maxim Gumin on GitHub: <https://github.com/mxgmn/WaveFunctionCollapse> which obtains these possible states.

³A mathematical Graph is defined by two sets: node set and edge set. While networks are defined by two arrays.

definition of the adjacency-matrix. The network description is only really useful when discussing relatively sparse networks/graphs. In addition to sparsity, properties such as: *small-worlds*, as shown in Watts and Strogatz 1998 and *power-law* as shown in Barabási and Albert 1999 become important.

Furthermore, in many applications, these arrays require significant memory due to the complexity of real-world data. However, conveniently, many entries are numerically zero. This lends itself to sparse representation, where only non-zero entries are stored along with their locations (their indices) in the dense representation. This sparse representation contains all the information content of the dense representation.

Neural-networks

One application for networks is the description of natural neurological systems. Here *neurons* are the nodes, while *synapses* are the edges. There have been many studies to preform a neuron-count for various animals, however unfortunately less research has been done on the synapses-count. When animals perform voluntary actions, they demonstrate an understanding of the-physics in their environment, and process new data accordingly, similar to operators. We may compute the connectivity-density of neural-networks by: $\text{density} = \frac{2s}{n^2}$, s being the synapse-count and n being the neuron-count. Because both synapse-count and neuron-count are truly astronomical, research has only rough estimates can be made for their values. Nonetheless, empirically a few patterns emerge, the smarter the species the *sparser* their neural-network. Although its difficult to assign a particular number

to intelligence, sparsity seems to do a good job, see figure 2, it is intuitively clear based on our interactions with animals, we can determine which are more intelligent relative to each other.

animal	neuron count	synapse count	density
human	8.61×10^{10}	1.00×10^{14}	2.70×10^{-8}
human-(3yo)	8.61×10^{10}	1.00×10^{15}	2.70×10^{-7}
brown rat	2.00×10^8	4.48×10^{11}	2.24×10^{-5}
cat	7.60×10^8	1.00×10^{13}	3.46×10^{-5}
house mouse	7.10×10^7	1.00×10^{12}	3.97×10^{-4}
honey bee	9.60×10^5	1.00×10^9	2.17×10^{-3}
drosophila	1.00×10^5	1.00×10^8	2.00×10^{-2}
fruit-fly	2.50×10^4	2.00×10^7	6.40×10^{-2}
roundworm	3.02×10^2	7.50×10^3	1.64×10^{-1}
sea-squirt	2.31×10^2	8.62×10^3	3.23×10^{-1}

Figure 2. This is a Table of Various Animals and Their Neural-network Density.

The following table of shows various animals and their neural-network density. Sources/citations for fig. 2 are referenced by: humans in Azevedo et al. 2009, brown-rat in Herculano-Houzel, Mota, and Lent 2006, cat in Ananthanarayanan et al. 2009, mouse in Herculano-Houzel and Lent 2005, honeybee in Menzel and Giurfa 2001, drosophila inSchlegel et al. 2023, fruit-fly inRaji and Potter 2021, roundworm in White et al. 1986, and sea-squirt in Ryan, Lu, and Meinertzhagen 2018.

1.5 Hypergraphs & Simplicial-Complexes

The networks we have considered so far consist of 0- and 1-dimensional objects, i.e., nodes and edges, respectively. However, a straightforward generalization is possible to n -dimensional hyper-edges, which include n nodes at once. This generalization yields

objects known as hypergraphs or hyper-networks. These objects represent a significant extension of graphs and networks but have not been as thoroughly studied. A related structure that has been extensively studied in differential geometry and algebraic topology is the *Simplicial-Complex*, which includes simplices (higher-dimensional triangles) analogous to the aforementioned hyper-edges. In many instances, an n -dimensional differential manifold can be cast into a Simplicial Complex with simplices of at most n dimensions. This process is known as *triangulation*⁴. In addition to hypergraphs, simplicial complexes have a nested-intersection structure⁵.

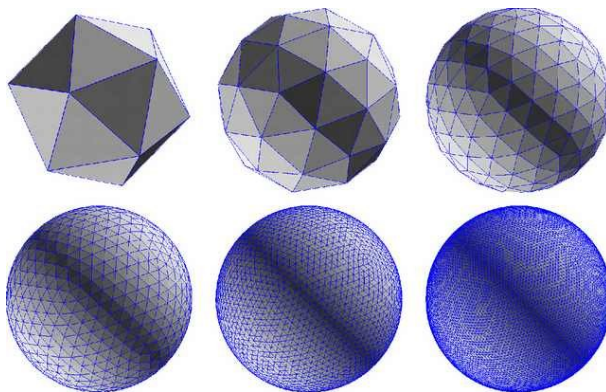


Figure 3. Six Simplicial-complexes Emulating the Sphere, S^2 , with Increasing Simplex-count. Note that the Highest-resolution Representation of the Sphere is the Sparsest (this is Because out of All the Possible Connections with More Nodes Fewer are Used). Note in the Limit of the Perfectly-continuous-mathematical-sphere we Obtain the Infinite-sparse Limit. Number of Total Possible Edges is: $\binom{n}{2} \sim n^2$, while Number of Presenting Edges is $6n$. While Number of Triplets are $\binom{n}{3} \sim n^3$, and Number of Faces are $\sim n$.

All networks, hyper-networks, and simplicial-complexes are sparse data structures,

⁴The Hauptvermutung Conjecture

⁵If a n -dimensional simplex K_n is included, then its boundary simplices, $\{K_{n-1}\}$, must also be included.

because of their usefulness in describing only a small subset of all possible connections, this is supported by fig. 3. Note, if we have the complete hypergraph, the inclusion of all possible hyperedges on n nodes/vertices, and we have physical interactions on all hyperedges, we once again obtain the exponential scaling problem:

$$2^n \sim \sum_i \binom{n}{i} .$$

1.6 Finite-Element

In the study of continuum mechanics and field theory, we consider functions defined within some space. In physics, we often simplify shapes to obtain analytic results, following the concept of a ‘spherical cow’. However, when dealing with real complexities, we may utilize grid methods to enumerate all locations in space and time, as mentioned in our introduction. However, a dense array of representative field and continuum values quickly becomes intractable. For instance, with three dimensions of space and one dimension of time, algorithms scale approximately as N^4 just to store a coarse representation. Instead, Finite Element Methods (FEM) first introduced by Turner et al. 1956, allow us to consider arbitrary real geometries used in engineering. Much like the ‘Hauptvermutung’, real-world complexities can be captured via triangulation. Today, FEM is standard in many engineering packages that utilize computers to prototype various situations. This development is interesting, because if any material can be decompose into a simplicial complex, it may also be related to a Discrete Exterior Calculus (DEC). Research in this field goes by the name

of Finite-Element Exterior Calculus, as developed by Arnold, Falk, and Winther 2006. Similar methods can also be applied to Lattice Gauge Theory calculations in Finite Element Gauge Theory, see Brower et al. 2016; Brower et al. 2021.

1.7 Bridging theory & High-Performance-Computing

Quick-and-naïve direct program implementations of theoretical-equations, while relatively easy to understand, often suffer from low-computational-performance. This is because High-Performance Computing (HPC) algorithms typically rely on the specific architecture of the computer and low-level programming languages that obscure the physics of the problem. In addition to common approximations in the theoretical equations embedded in their efficient implementation, these “makeshift” solutions are often highly specialized for a particular problem and cannot be easily generalized to others. The problem-solving paradigm typically falls into two categories: either it’s easy for humans (straightforward implementation following closely from theory) but difficult for computers (high computational complexity), or it’s difficult for humans (complicated implementation with contrived factors and approximations) but easy for computers. In the past decade, Python’s popularity has surged due to its capacity to quickly connect highly specialized, often compiled, black-box subroutines. Many theoretical problems can be transposed into multidimensional array problems, necessitating efficient multidimensional array manipulation.

1.7.1 NUMPY's einsum Function

Albert Einstein is perhaps best known for his discovery of General Relativity in 1915. In developing his theory, he generalized vector analysis into tensor analysis to study differential geometry. He recognized the recurring theme of summing over certain tensor indices. To address this, a common approach in *pen-to-paper* work is to label all tensor indices belonging to a list of tensors with alphabetical letters. Matching indices are assumed to be summed over, following the Einstein-Summation convention (see Einstein 1916). This operation is a broad generalization of the matrix-product, as in $A@B = A_{ik}B_{kj} = C_{ij}$. Therefore, this operation is perhaps the most fundamental and universal operation for working with two or more generic multidimensional arrays. Examples of the Einstein convention within the Einstein field equations include (with g as the metric tensor):

$$\begin{aligned}\Gamma_{ij}^k &= \frac{1}{2} g^{kl} (\partial_i g_{jl} + \partial_j g_{il} - \partial_l g_{ij}) \\ R_{\mu\nu} &= \partial_i \Gamma_{\mu\nu}^i - \partial_\mu \Gamma_{\nu i}^i + \Gamma_{ip}^i \Gamma_{\mu\nu}^p - \Gamma_{\mu p}^i \Gamma_{i\nu}^p, \\ R_{\mu\nu} - \frac{R}{2} g_{\mu\nu} &= \frac{8\pi G}{c^4} T_{\mu\nu} \quad .\end{aligned}$$

The `einsum`-function is one of the most important and powerful algorithms in computer-science and science more generally. It allows for the efficient manipulation of tensors/arrays in their native degrees-of-freedom, *i.e.* their indices. This algorithm is user-friendly and computer-friendly. For a summary of deep-learning applications of `einsum`, I would refer to the reader to an article by Rocktäschel 2018

1.7.2 Strassen Theory & AlphaTensor

Since Strassen’s 1969 discovery, Strassen 1969, that matrix-matrix multiplication can be achieved in less than $\mathcal{O} \sim N^3$ operations, matrix-matrix multiplications have been an active area of research. A major contributor has been the work of AI firm DeepMind’s AlphaTensor, Fawzi et al. 2022, and subsequent work by others. The “hope” or aspiration of this research is whether matrix multiplications can be reduced to $\mathcal{O} \sim N^2$ time-complexity.

1.7.3 Tensor Processing Units

These partial-tracing operations on dense matrix-matrix and dense matrix-vectors are very common in deep-learning. Given the widespread impact on deep-learning on current-society, motivated the development of Application-Specific-Integrated-Chip (ASIC) known as the Tensor-Processing-Unit (TPU), Jouppi et al. 2017 to preform these operations. There have been 4 generations of TPUs, with the latest being called TPUv4, as developed by Jouppi, Yoon, Ashcraft, et al. 2021, using 7 nm chip-technology. TPUv4i is optimized to preform 128×128 matrix-multiplications natively, with a performance of 138 TFLOPs (in 16-bit float/8-bit int). Despite the TPU being a ASIC, this one operation is ubiquitous in science that it may be applied to electronic-structure problems, as in Ganahl et al. 2023 and Pederson et al. 2022, and fluid mechanics problems, as shown in Q. Wang et al. 2022.

1.8 Sparse-Tensor Research

As emphasized in the previous sections, the significance of sparsity and tensors in science suggests a growing interest in sparse tensor methods. While not widely explored, a few groups have delved into these methods. The development of block-sparse tensor methods began with physical chemists, particularly the Krylov group, which introduced a new implementation Epifanovsky et al. 2013 for computing post-Hartree-Fock correlated electronic structure methods. This was followed by work in the 2010s by the Solomonik group, with publications such as Solomonik et al. 2014; Solomonik and Hoefler 2015; Kanakagiri and Solomonik 2023, emphasizing parallelism. More recently, in the 2020s, Choy et al. of NVIDIA have worked on sparse tensor networks and machine vision Choy, Gwak, and Savarese 2019.

1.9 Physical Problems

1.9.1 Nonlinear Dynamics

The fact that natural images and data may be rendered in a sparse format (given a suitable basis), leads to their sparse-format being used for efficient numerical-algorithms. Recently there also have been many advances in modelling nonlinear dynamics using sparsity, we give a brief example of this algorithm. Suppose we are given dynamic (in-time) random/experimental data-points, R_{tx} (time and Cartesian-

coordinates). Our objective is to determine which function fits this data the best. Following the work of **brunton2022**, we will focus on understanding nonlinear dynamics. Suppose we have nonlinear temporal data R_{tx} , with this we may compute its temporal derivative \dot{R}_{tx} . We may also be able to compute many functions directly from the data, R_{tx}^2 , $\partial_i R_{tx}$, $\sin R_{tx}$, and etc... This may be collected into a matrix with the columns as functions and the rows being the temporal data:

$$\Theta_{tf} = \begin{pmatrix} R^2 & R^3 & \partial R & \sin R & \dots \end{pmatrix} \quad .$$

Within this basis we may consider, the 1st order time dynamical differential equation:

$$\dot{R}_{tx} = \Theta_{tf} \xi_{fx} \quad .$$

Where ξ_{fx} are the coefficients for the basis-array Θ . This equation is under-determined, and thus may have many possible solutions in the given basis. The desired solution is *magically* the sparsest-version of ξ_{fx} within the basis-of-functions given by Θ_{tf} , to generate the best model. Recently, there have been mathematical algorithms, Candès, Romberg, and Tao 2006, to promote these sparse-solutions, and these techniques may be used to solve the under-determined equation above. This algorithm was shown to reproduce/discover the Navier-Stokes Equations!

1.9.2 Quantum Many Body Problems

Similar to nonlinear dynamics of the previous subsection, albeit linear, Quantum-Many-Body (QMB) Problems also pose an insurmountable problem without any assumptions. Fortunately, as shown in later chapters (chapters 4 and 5) the Density-

Matrix-Renormalization-Group (DMRG) or Matrix/Tensor-Product ansatz allows for a dramatic reduction in computational-complexity. From exponential to polynomial or even linear computational complexity!

1.9.3 Radiation Damage

The original motivation of this work was in understanding *radiation-damage* mechanisms in ultrafast electron and x-ray (light) sources. Radiation-damage at the ultrafast time-scale involves the severing of covalent bonds. These covalent bonds are many-electron systems which constitute a strong-correlation problem, which must be resolved by a suitable QMB problem, especially when away from the vibrational equilibrium position (the transition from strong \rightarrow weak coupling, see chapter 6). This fact is a major impediment to understanding radiation-damage and the behavior of molecules on the ultrafast time-scales. In many avenues in this research the sparse approximation would need to be invoked as suggested later in this work.

1.10 Organization and Goals

It is well-known that if the matrices are sparse, their matrix-multiplication algorithm can scale much less than $\mathcal{O} \ll N^3$, scaling instead linearly with the number of nonzero/sparse elements. Algorithms which scale linearly ($\sim N$), or sometimes pseudolinearly ($\sim N \log N$), have a technical term: *fast*. In this thesis we explore how

to generalize this concept to multidimensional-arrays, this forms the basis of chapter 2 and 3. This should prove to be an invaluable tool for solving computationally difficult tasks such as the infamous Quantum-Many-Body (QMB) problems (in particular the $\mathfrak{su}(N)$ Heisenberg-model), introduced in chapter 4. Examples of numerical solutions of these problems are demonstrated in chapter 4 and 5. We complete our discussion by addressing the original motivation of this work: Radiation Damage, via a chapter 6 on Molecular Dynamics. Although, the physical universe is *dense*, all long-range interactions between all particles exist, the idea is that this universe can be understood in the sparse limit. And therefore the ultimate idea is to solve physical problems under the constraint of sparsity (in addition to others, e.g. symmetry). These physical problems may be defined by some operator, e.g. Hamiltonian or Koopman. Either way, this is a guess because there is no *a priori* need for this condition, and is solely taken by faith, by the *simplicity-doctrine* or *principle-of-parsimony*, *i.e.* Occam's Razor.

Chapter 2

SINGULAR-VALUE & SCHMIDT DECOMPOSITION

2.1 How to Define a Wavefunction?

According to *Matrix-Mechanics*, an early theory of quantum-mechanics invented by Heisenberg, Born, and Jordan, as shown in Born, Heisenberg, and Jordan 1926; Born 1955, the wavefunctions capture the coherent-ignorance of a system over some known *orthonormal-basis*. This orthonormal-basis can be almost anything, *e.g.* molecular-orbital states, polarization/spin states, scattering states; just like classical-incoherent probabilities that can be applied to almost all real-world situations. In this framework, the wavefunction merely captures coherent-probabilities (amplitudes) as an array of complex-numbers, a vector (matrix) of complex-numbers. These are determined by another matrix of complex-numbers, the Hamiltonian. Either way, these array-of-numbers are the intrinsic fundamental elements of this theory.

The task of quantum-theory is to compute the wavefunction Ψ (modulo a global-phase), or properties from it, *i.e.* the observables, this yields the predictive properties of the theory. It is truly fascinating that a fundamental theory is intrinsically linear. When we consider Quantum-Many-Body (QMB) systems, we may keep the vector picture for Ψ , or the more generic *tensor* representation will be shown to be more convenient in manifesting the effective 1-body, 2-body, \dots , n -body representations of the wavefunction under consideration.

2.2 What is a Tensor?

There is much confusion over this question, likely due to the ubiquitous nature of tensors across many disciplines. The history of tensors can be traced back to the work of Ricci-Curbastro, Levi-Civita, and Gibbs. Tensors are rigorously defined as multi-linear maps. We start with the following definition of an array, applicable to multidimensional arrays as used in computer science.

Definition 1: Array (math/computer-science)

An **Array** is a regular multi-dimensional cubic spacing of $n_1 \times n_2 \times n_3 \times \cdots \times n_N$ elements with entries from a **Ring** R (for positive integers $n_i \in \mathbb{Z}^+$, indicating the quantity of elements in each dimension).

The collection of numbers $(n_1, n_2, n_3, \cdots, n_N)$ form the *shape* of the Array.

An Array is said to live in the R -module (essentially a generalized vector-space).

Following definition 1, a *matrix* in the mathematical-context (Linear-Algebra) is 2-dimensional Array, with entries taking a **field**⁶. In computer-science and other fields, these **arrays** are sometimes called *tensors*. Next let's discuss this main character, *tensors* and *sparse-tensors*. Unlike Arrays which live in module-space, tensors live in *vector-spaces*. The definition of tensors is given in many math, see Needham

⁶A **field** is particular type of **ring**. Fields notably exclude many types of \mathbb{Z} sets.

2021; Dummit and Foote 2004; Lee 2012 and physics, Thorne, Wheeler, and Misner 2000, books, but here we consider a slight generalization that is often used to include modules (versus the more common vector-spaces).

Definition 2: Tensor (math/physics)

Suppose we have a module V and its dual-module V^* , then a Tensor is defined on a Cartesian-product of this space, and is defined by the following multilinear map:

$$T : \underbrace{V^* \times \cdots \times V^*}_{N \text{ copies}} \times \underbrace{V \times \cdots \times V}_{M \text{ copies}} \rightarrow F \quad .$$

That is a tensor can be viewed as a function (with slots, analogous to a multi-variable continuous function, *e.g.* $f(x, y, \dots, z)$):

$$T[_, _, \dots, _; _, _, \dots, _]$$

The slot-variables for the multi-linear-functions are called indices.

The definitions above help relate us back to computer-science and concrete implementations of theories we wish to implement. Next, let's discuss an important aspect of tensors, their decompositions.

Definition 3: Tensor-Rank

Suppose we have a n -axis tensor T . We define the *tensor-rank*, with-respect-to the tensor-product (\otimes). Let's represent an arbitrary 1-axis/dimensional tensors by X_i , then we can decompose T (as a sum \sum of

tensor-products (\otimes):

$$T = \sum_j^{\text{rank}} \bigotimes_i^n X_{[i]}^{(j)} \quad . \quad (2.1)$$

Then T 's rank is the **tensor-rank** ($\in \mathbb{N}$), is the smallest possible value of rank (in the equation above). Equation 2.1, is also called: the *tensor-rank-decomposition*.

Let's give an example of a tensor, $M \in \mathbb{C}^n \otimes \mathbb{C}^n \otimes \mathbb{C}^n$, its tensor-rank-decomposition is given by (collections of 1-tensors $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$):

$$M = \mathbf{a}_1 \otimes \mathbf{b}_1 \otimes \mathbf{c}_1 + \mathbf{a}_2 \otimes \mathbf{b}_2 \otimes \mathbf{c}_2 + \mathbf{a}_3 \otimes \mathbf{b}_3 \otimes \mathbf{c}_3 + \mathbf{a}_4 \otimes \mathbf{b}_4 \otimes \mathbf{c}_4 + \mathbf{a}_5 \otimes \mathbf{b}_5 \otimes \mathbf{c}_5 \quad .$$

Let's give an explicit example of a rank-1: 2-dimensional tensor:

$$M = \begin{pmatrix} 4 & 5 & 6 \\ 8 & 10 & 12 \\ 12 & 15 & 18 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \otimes \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$$

Even though the original 2-dimensional tensor seemingly appears to have *random/independent* entries, its clearly shown they can be decomposed into two 1-dimensional tensors. Therefore, M can be considered 'sparse'. While a tensor is 'dense' if it has full or close to the maximum tensor-rank. The maximum tensor-rank for any given tensor is still an unsolved problem. Naively, one would think this rank is equal to the number of entries, but due to the structure of tensor products, many entries can be simultaneously addressed, therefore dramatically decreasing this value. An example is given in maximum-rank, by Buczyński and Landsberg 2013, of a tensor in $\mathbb{C}^N \otimes \mathbb{C}^N \otimes \mathbb{C}^N$ is $N^2 - N - 1$, e.g. for $N = 3$, max-tensor-rank is 5, a tensor of rank 5 is 'dense'.

2.2.1 What is a Sparse-Tensor?

Namely, Sparse-Tensors/Arrays have the ability to be easily factored. Sparse-tensors/arrays are properly tensors xor arrays, however its pure information content is usually much lower than face-value.

Definition 4: Sparse-Tensor

Sparse-tensors are *tensors* (suppose of dimension n & axes size N) whose *tensor-rank*, r , allows for a smaller representation (dense-tensors/arrays have N^n entries):

$$N^n > rnN \quad .$$

Its commonly referred as an attribute of tensors/arrays, however as we shall see these sparse-tensors/arrays exist in their own interesting paradigm, inheriting network/graph structure. A common symptom of sparse tensors/arrays is they are mostly filled with the trivial multiplicative *ideal*⁷, *i.e.* ‘0’, and hence the purpose for def. 5. These definitions are somewhat related because of zero’s special property, and that tensor-products of tensors with mostly zeros, multiply, let A and B be tensors with nonzero densities: ρ_A and ρ_B respectively, then $A \otimes B$ has nonzero density $\rho_A \rho_B \leq \min\{\rho_A, \rho_B\}$.

⁷Zero has the marvelous property that $x \times 0 = 0$, for all numbers x .

Definition 5: Sparse-Array (computer-science)

Sparse-arrays are *arrays* consisting of mostly '0' (zeros).

2.3 The Tensor-Network

Here we define an abstract and generic sparse-data-structure class, which can be tailored to many problems.

2.3.1 Tensor Diagrammatic Representation

Here let's define tensors by multidimensional arrays with multiple-axes representing each dimension. We may represent these graphically by a node with multiple legs, as shown in fig.4. Each *leg* is mathematically referred to as a *half-edge*, and represents each axis. Note if we have N elements over a given axis, then a n -axis tensor has N^n elements.

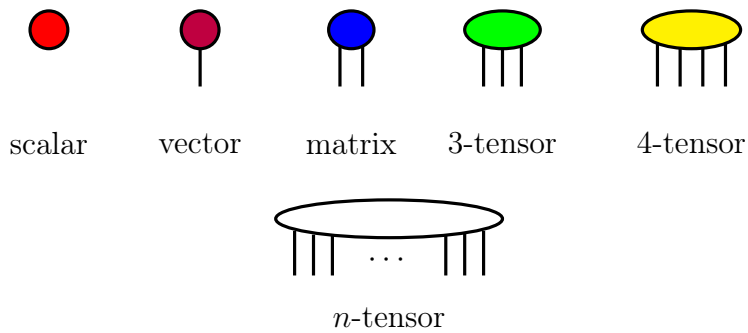


Figure 4. Various Graphical Representations of Tensors/Arrays.

In relativity studies, tensors may have two kinds of legs: contravariant and covariant. In the Penrose-tensor-convention, shown in Penrose and Rindler 1984, contravariant legs face towards the top-of-the-page while covariant face towards the bottom-of-the-page. In many-body quantum systems, each leg is associated with a 1-particle Hilbert space, as these are complex objects also come into two varieties: \mathcal{H} and its complex-conjugate $\bar{\mathcal{H}}$. There are many excellent resources detailing these kind of diagrams, these include Bridgeman and Chubb 2017; Roberts et al. 2019. Additionally, some authors use different node symbols (and colors) for different types of tensors, *e.g.* in the context of tensor-decompositions. Let's give an example of the representation of the infamous many-body Schrödinger-equation using these kind of diagrams:

$$H\Psi = \mathcal{E}\Psi \quad \rightarrow \quad \begin{array}{c} | \\ \circlearrowleft H \\ | \\ \circlearrowright \Psi \end{array} = \mathcal{E} \quad \begin{array}{c} | \\ \circlearrowright \Psi \end{array} .$$

Let's notice two kinds of edges: *internal* and *external*. All *internal*-edges are contracted/partial-traced over. However this diagrammatic picture obfuscates the many-body nature of the problem, and instead of this we may unfold/reshape all n -particles' degrees-of-freedom into the tensor-network:

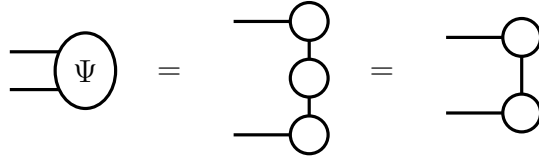
$$\begin{array}{c} | \quad \dots \quad | \\ \text{---} H \text{---} \\ | \quad \dots \quad | \\ \text{---} \Psi \text{---} \end{array} = \mathcal{E} \quad \begin{array}{c} | \quad \dots \quad | \\ \text{---} \Psi \text{---} \end{array} .$$

The above diagram directly illustrates the difficulty of the problem, each tensor (both H and Ψ) scales exponentially, N^n , in size (number of particles). Within the past

few-decades it was realized that these tensors may be decomposed or factored into smaller pieces that may be solved-sequentially, White 1992.

2.3.2 Singular-Value-Decomposition

The most famous form of tensor, or rather matrix, decomposition is the Singular-Value-Decomposition (SVD). The SVD algorithm despite being invented over a century ago, plays an increasingly important role in the current information-era, **brunton2022**. The SVD factors any matrix into 3 smaller matrices, very much like the eigendecomposition. Diagrammatically this is represented by (using the bipartition of the wavefunction):



The middle-matrix/the singular-values is diagonal (with only internal-indices), and whose node may be represented by some by a diamond, \diamond . This singular-matrix may be combined with either matrix with an external-index, this yields the final diagram.

SVD is defined for matrices⁸, for singular-values λ_i :

$$W_{ab} = \sum_i^r \lambda_i (a_i \otimes b_i)$$

Because SVD, may be applied to non-square matrices, it may be viewed as a generalization of matrix-diagonalization.

⁸The computation of the SVD of a generic matrix A reduces to the computation of L , R , and λ from the following eigenvalue-equations: $(A @ A^T)L = \lambda_L L$ and $(A^T @ A)R = \lambda_R R$, with the choice: $C = \min\{|\lambda_L|, |\lambda_R|\}$ (L xor R), and value: $\lambda = \sqrt{\text{nonzero}(\lambda_C)}$.

Although, W (above) might be sparse, L, R matrices are unlikely to be sparse (arrays mostly consisting of 0s). However, often the number of entries is much smaller⁹ than the original matrix: $|A| \gg (|L| + |R|)$. The general n -tensor may also be 1-tensor-decomposed:

$$W_{abc\dots z} = \sum_i^r \lambda_i (a_i \otimes b_i \otimes c_i \otimes \dots \otimes z_i) \quad .$$

Graphically, the generalized SVD (tensor-rank-decomposition) is given as:

$$\boxed{\text{Oval with 4 lines}} = \lambda_0 \boxed{\text{4 circles}} + \lambda_1 \boxed{\text{4 circles}} + \dots + \lambda_r \boxed{\text{4 circles}}$$

note that all first-order tensors are not in-general the same (equal to each other), they are all distinct.

2.3.3 Tensor Decompositions

Generalizations of SVD have been developed for generic tensors (rather than matrices, *i.e.* 2-tensor), sometimes this is called High-Order (HOSVD). This is straight-forwardly implemented by reshaping of tensor-axes¹⁰. The essential idea is the external-edges denote the boundary of a tensor-network, and this tensor-network holds many internal-indices relevant to a given problem. Then these are fully-traced/contracted we obtain the original tensor (sometimes approximately, if truncations were considered), nevertheless the tensor-network form allows for efficient storage and manipulations. A general HOSVD/tensor-network decomposition may resemble:

⁹Hence, if L and R are padded with 0s to match the size of A , they will mostly be zeros and hence fall under *sparse* in this definition.

¹⁰However conceivably there are alternative tactics for achieving this.

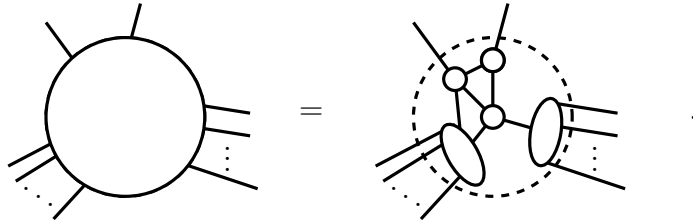
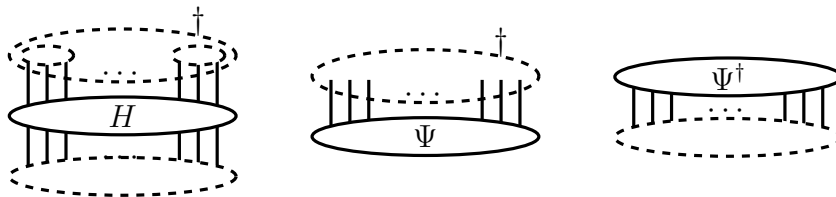


Figure 5. The Decomposition of a Sparse-tensor (Previously Known as a Dense-tensor) into Network of Dense-tensors.

2.3.4 External-index Surfaces

The collection of all external-indices for a tensor or tensor-network constitutes the *surface* or *boundary* of the tensor/tensor-network. In quantum-many-body systems we may partition the surface into two kinds of boundaries, these are legs associated to the 1-particle Hilbert-space, \mathcal{H} and its complex conjugate, $\bar{\mathcal{H}}$. Many-body-quantum-wavefunctions are *typically*¹¹ associated to only one kind of boundary, while operators are associated to both. This is readily seen by (the dashed ovals indicate the *surface/boundary*):



Any sub-collection external-indices also constitutes a boundary which may be identified with another tensor/tensor-network. In the tensor-network literature, these kind of

¹¹Many-Body wavefunctions, Ψ , may belong to a mixture of $\{H, \bar{H}\}$, but on a given site it can only belong to one of those Hilbert-spaces.

external indices connecting to the wavefunction are called the *physical*-indices. While, the intra-operator internal-indices will serve as the underlying graph of the theory.

2.4 Tensor-Network as a Sparse-Tensor

Let's formalize and summarize our findings, as shown in fig. 5. A sparse-tensor may be defined as an list/array of tensors and a network-structure, called the tracing-rule (or tracing-prescription). The network structure (in the name-sake of tensor-*network*), identifies the various tensor axes to each other, indicating the partial-tracing structure. As will be shown, this kind of sparse-tensor, not only leads to storage savings, but also computational savings. This is because this tensor can be worked in its compressed form, or sparse form.

2.5 Unstructured Sparse-Arrays

In this section we describe a different kind of sparse-tensor. Here we directly capture the structure of unstructured sparse-multidimensional arrays (sparse array), also commonly referred to as *sparse-tensors*. Dense arrays (regular arrays) have their data stored directly adjacent to each other in physical contiguous memory. Here we consider unstructured-Sparse-arrays, these will be represented by 3-dense-arrays. Sparse-Arrays are meant to represent dense-arrays perfectly, with the assertion that elements which are zero are neglected¹². The sparse-arrays may be broken into 2

¹²sometimes approximately zero, in which case the representation is approximate.

primary parts: A 2-dimensional dense index-array and 1-dimensional dense data-array. The sparse index-array is represented by:

$$\text{Sparse Index-Array} \equiv A [\quad | \quad] \quad ,$$

with the vertical-line, $|$, separating this 2d array's two indices. With the left-side representing the dense-index columns¹³. And the right-side indexing the enumeration of the sparse-tuple entry. In order to draw parallel with the dense-representation, the dense-indices are often shown explicitly included. For example for a tensor A , with dense-indices $ijkl$, and sparse-index I (enumerating all the non-zero values):

$$\begin{aligned} \text{Dense Representation} &\equiv A [i, j, k, l] \\ \text{Sparse Representation} &\equiv A [i, j, k, l | I] \quad . \end{aligned}$$

Notice, the dense representation does not have the I (sparse) index. Additionally, a small bit of additional information is needed for the sparse-to-dense map, the *dense-shape*, or *shape*, of the dense-representation of the array. Therefore the complete attributes of the sparse-array include: shape-array (s), array-of-tuples/index-array (A_{index}), and the data-array (A_{data}). The full anatomy of a sparse-array is thus given in fig. 6. There are multiple implementations of these objects, sparse-arrays/tensors, in popular python libraries, such as PyTorch (Paszke, Gross, and al. 2019), JAX (Bradbury et al. 2018), and TensorFlow (TensorFlow et al. 2015). However, as of now mid-2023, they do not carry the functionality described here.

¹³Crucially with entries columns now representing the dense-array's axes.

Shape

The shape-array is a 1-dimensional array which gives the shape of the array in its dense-representation. This is useful for the mapping between sparse-arrays and dense-arrays.

Index-array

The sparse-array index-array or *indices* or *array-of-tuples* is by convention lexicographically-ordered. This helps in identifying duplicates, and other sparse-array operations. The index-array captures the structure of the array.

Data

For us the *data-arrays* are 1-dimensional with a length matching the index-arrays, and being in 1-1 correspondence with the index-array. That is the list-index (of the index-array) matches the data-array index. These arrays are typically of `float` datatypes, and have no restriction on duplicates, with a slot-representation of $A_{\text{data}} [\]$. A more general setting; e.g. *block-sparse* which allows for data-arrays to be arbitrary dimensional multi-dimensional with 1-dimension matching the index-arrays length (indexed by I).

Density

The density is defined by (assuming A_{index} is in a proper format, only filled with unique entries):

$$\begin{aligned} \text{density} &= \frac{\text{total elements in proper sparse-tensor}}{\text{total elements in dense-tensor}} \\ &= \frac{\text{len}(A_{\text{data}})}{\prod_i \text{shape}[i]} = \frac{1}{\text{sparsity}} \quad . \end{aligned}$$

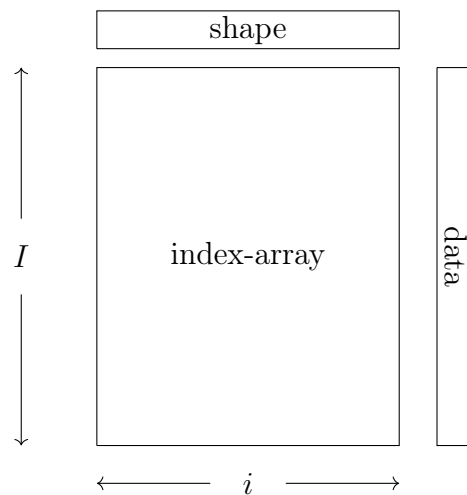


Figure 6. The Decomposition of the Sparse-array into a Shape, Array-of-tuples, and Data Arrays.

2.5.1 Well-ordering

Suppose we have a lexicographically-sorted index-array, $A_{\text{index}} \llbracket \quad | \quad \rrbracket$, and it's unques array, u (collection of integers indicating the first appearance, index, of a

unique element in the sorted array). Then the duplicate elements are all the elements in the lexicographically-sorted sparse-tensor between two successive unique elements, $u[i] : u[i+1]$ (not including $u[i+1]$). Therefore, we sum all these elements of $A \llbracket \quad | \quad \rrbracket$ for all successive-elements in u . This creates an array with unique data values for each index-tuple:

$$A_{\text{index}} \llbracket \quad | \quad \rrbracket = A_{\text{index}} \llbracket \quad | \quad u \rrbracket \quad ,$$

$$A_{\text{data}} [\quad] = \sum_{i \in |u|} A_{\text{data}} [u[i] : u[i+1]] \quad .$$

Note $u[i+1] = |u|$ for the last unique-array's index, $i = |u|$.

Sparse-arrays can only faithfully represent dense-arrays (bijectively) iff their row-wise elements of the index-array are unique. In order to verify that this is the case for an arbitrarily defined index-array, they are put canonically sorted into lexicographic-order (refer to the appendix) to ensure they are in fact *well-ordered*. *Well-ordering* is a strictly ordered/sorted list, i.e. using the relation $<$. Such that for-all entries $a, b \in A$ (A being a list/array) with element indices i and j respectively, then $a[i] < b[j]$ for $i < j$. A weaker statement is *partial-well-order*, this indicates the possibly for duplicates, while retaining lexicographic-order, i.e. \leq . We will refer to *well-ordering* an array, when discussing summation over duplicate tuple elements, while retaining lexicographic-order¹⁴.

¹⁴Alternative names could be *coalesce*.

2.6 Tensor Operations

2.6.1 Composition, Sub-indexing, and Permutations

Given two 1-dimensional arrays A and B , we can compose/subindex one by the other. Mathematically, this is written over functions with $A \circ B$, alternatively we may use a square-bracket notation, e.g. $A[B]$. If B can be bijectively mapped to A , all entries in B are unique, then this is called a *permutation*. Often to sort arrays, say A , we determine the permutation array, B , such that when composed, $A[B]$, the result is ordered. This is useful when talking about sparse-arrays, because of their interconnected array structure.

2.6.2 Array-labeling

We may associate a character/letter to every axis of a tensor/array. For instance, without-loss-of-generality if A is a 4-dimensional array, then $A[i, j, k, \ell]$ denotes a labeling of A , for labels $\{i, j, k, \ell\}$. These characters are completely arbitrary, and are merely used to match certain indices/axès to others to create edges (or hyperedges more generally, such that every edge is associated to a type of character).

2.6.3 Cartesian-product

The *cartesian-product* is all the possible 2-tuples created by all entries in A (1st tuple entry), and all the entries in B (2nd tuple entry). The cartesian-product may be generalized to products of n lists-of-tuples to obtain a list of n -tuples of tuples. The tuple-of-tuples may be straightforwardly reduced to a larger order m -tuple, whereby $m > n$. Further, details of this operation are given in §5 of Munkres 2000. Originally, an operation in set-theory, when applied to lists/arrays they maintain *wellorderedness* iff both original arrays were well-ordered. If we consider the cartesian-product on partially-ordered arrays. Their cartesian-product is neither well-ordered nor partially-ordered. Naïvely we would hope that the product could produce a partially-ordered list-of-tuples. There is a straight-forward permutation correction without having to resort to an array sort.

Direct-product (Dense Arrays/Tensors)

Perhaps the most general operation on arrays is the direct-product (*a.k.a* tensor or Kronecker-product), it is the product of all entries in one array to the other. The result is a new tensor/array with a rank equal to the sum of the constituent ranks.

An example of this operation over two 2-dimensional arrays:

$$A_{\mu\nu} = \begin{pmatrix} A_{00} & A_{01} & \cdots & A_{0n} \\ A_{10} & A_{11} & \cdots & A_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m0} & A_{m1} & \cdots & A_{mn} \end{pmatrix} \quad B_{\kappa\lambda} = \begin{pmatrix} B_{00} & B_{01} & \cdots & B_{0\ell} \\ B_{10} & B_{11} & \cdots & B_{1\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{k0} & B_{k1} & \cdots & B_{k\ell} \end{pmatrix} .$$

Then the direct-product, \otimes , is explicitly defined as:

$$A_{\mu\nu} \otimes B_{\kappa\lambda} = \begin{pmatrix} A_{00} \begin{pmatrix} B_{00} & B_{01} & \cdots & B_{0\ell} \\ B_{10} & B_{11} & \cdots & B_{1\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{k0} & B_{k1} & \cdots & B_{k\ell} \end{pmatrix} & \cdots & \cdots & A_{0n} \begin{pmatrix} B_{00} & B_{01} & \cdots & B_{0\ell} \\ B_{10} & B_{11} & \cdots & B_{1\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{k0} & B_{k1} & \cdots & B_{k\ell} \end{pmatrix} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_{m0} \begin{pmatrix} B_{00} & B_{01} & \cdots & B_{0\ell} \\ B_{10} & B_{11} & \cdots & B_{1\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{k0} & B_{k1} & \cdots & B_{k\ell} \end{pmatrix} & \cdots & \cdots & A_{mn} \begin{pmatrix} B_{00} & B_{01} & \cdots & B_{0\ell} \\ B_{10} & B_{11} & \cdots & B_{1\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{k0} & B_{k1} & \cdots & B_{k\ell} \end{pmatrix} \end{pmatrix} .$$

Symbolically this may be represented by:

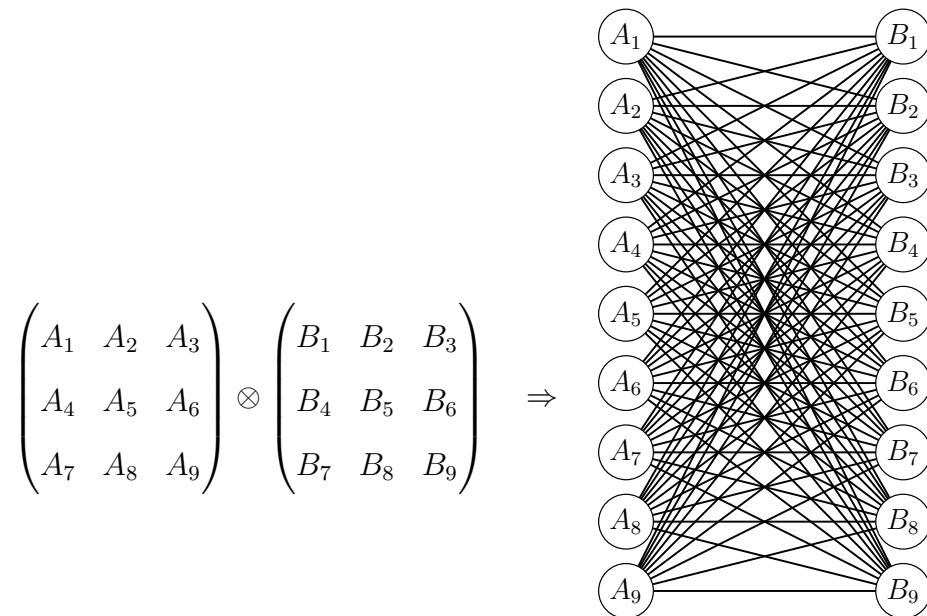
$$A[\mu, \nu] \otimes B[\kappa, \lambda] = AB[\mu, \nu, \kappa, \lambda] .$$

The general direct-product over n -dimensional arrays is similarly:

$$A^{(0)}[m_0] \otimes A^{(1)}[m_1] \otimes \cdots \otimes A^{(n)}[m_n] = C[m_0, m_1, \cdots, m_n] .$$

This operation may be represented by a network/graph. If every data-entry in A is given a node, and likewise for B . The products are associated to edges connecting

them, then the complete-bipartite graph captures this product. For the general direct-product involving n tensors/arrays the resulting graph is the complete- n -partite graph.



Direct-product (Sparse Arrays/Tensors)

Suppose we have two well-ordered A & B sparse-arrays, then their direct-product is: the binary-cartesian-product over their respective list-of-tuples, with a reshaped/flattened direct-product of their data, and concatenated shape-tuple:

$$\begin{aligned}
 A^{\text{index}} \times B^{\text{index}} &= C^{\text{index}} \\
 \text{reshape}(A^{\text{data}} \otimes B^{\text{data}}) &= C^{\text{data}} \\
 A^{\text{shape}} \parallel B^{\text{shape}} &= C^{\text{shape}} \quad .
 \end{aligned}$$

with $||$ denoting the array concatenation operation over the tuple-axis (e.g. $A_I^t || B_I^{t'} = C_I^{t''}$, with $t'' = t + t'$). This readily generalizes to n -sparse-arrays, $A^{(n)}$:

$$\begin{aligned} A_{\text{index}}^{(0)} \times A_{\text{index}}^{(1)} \times \cdots \times A_{\text{index}}^{(n)} &= C_{\text{index}} \\ \text{reshape} \left(A_{\text{data}}^{(0)} \otimes A_{\text{data}}^{(1)} \otimes \cdots \otimes A_{\text{data}}^{(n)} \right) &= C_{\text{data}} \\ A_{\text{shape}}^{(0)} || A_{\text{shape}}^{(1)} || \cdots || A_{\text{shape}}^{(n)} &= C_{\text{shape}} \quad . \end{aligned}$$

Note all $A_{\text{index}}^{(n)}$ arrays all have to be well-ordered for C_{index} to be *well-ordered*.

2.6.4 Intra-intersection/Slice

Now suppose we are given a single tensor/array, and we would like to identify some columns/axès with each-other (via labels). This reduction in the array's degrees-of-freedom is called slicing, or intra-intersecting. Because these are internal operations, these are best done first, as intra-intersecting can only reduce the size. Unfortunately, if the intersecting columns are not adjacent to each other or the intra-intersecting indices are internal, they must be re-well-ordered (at least from the first intra-intersection column/index/axis to the rest). This operation requires an element-wise search of all rows in a given sparse-array's index-array.

2.7 Duality: Unstructured-sparse & Tensor-network Sparse Tensors

Let's map the SVD definition of sparse to the unstructured definition. In SVD/HOSVD/TN we have sums of 1-dimensional-tensors. Clearly, the *worse*¹⁵ representation is the Cartesian-basis, but it provides the map, a 1d-tensor with binary/Boolean values $\{0,1\}$, with a single entry being the nonzero entry: '1'. These basis-vectors may be perfectly/bijectively mapped to the integers denoting the location of the '1'. This can be done over all basis-vectors, I , of the tensor-product-decomposition: $W_{ijk\dots l} = \bigotimes_I^n X_i^I$. We thus may associate a product of basis-1-tensors by a tuple addressed to a data entry (the definition of unstructured sparse). An example is shown below:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \leftrightarrow \begin{pmatrix} 3 & 2 & 0 \end{pmatrix} .$$

2.8 Tensor Types

We have discussed the dense-tensors, however, often these are not the best representations for storing or manipulating the tensor's data.

- **dense** : tensor data is storage is continuous, immediately-adjacent in the data

¹⁵As we've mentioned the tensor-rank is typically much less than N^n , with sides of length N and dimension n

entries corresponds to being immediately-adjacent in physical computer memory. All data entries are stored, and thus dense-tensors carry inherit similarly sized rows (all rows are the same size.).

- **sparse (unstructured)** : tensor data is stored in a 1-dimensional representation, with every data-entry addressed by a unique (ideally) integer n -tuple (for an n -dimensional dense tensor).
- **blocksparse** : generalizing the unstructured sparse, by replacing data entries to include entire dense tensors in there place. Thus a unique n -tuple addresses an entire dense tensor. All data-entry-dense-tensors are of the same size and shape.
- **sparse (structured)** : tensor data is once again stored in a convenient format, but the addresses are given by a rule. This tensor format is the most subjective, and may take many forms. *E.g.* band-matrices are a good example of structured sparse, as each band (diagonal slices of matrices) may be stored in memory and used as needed.

Chapter 3

SPARSE-PARTIAL-TRACING

Following our discussion about the importance of sparsity in describing the physical and natural-world, we would like a method to natively contract higher-dimensional sparse data structures such as sparse-arrays or sparse-tensors. Similarly to the principle-of-sparsity, tensor-contractions are ubiquitous in physics, and in this section we shall show a method to precisely achieve this for sparse-tensors/arrays. This work was first posted on the arXiv in Candanedo 2023b.

3.1 Partial-Trace

The last operation described was the summation over the identification of two axes, e.g. if we have a 2d-array, a matrix, the diagonal elements correspond to the part where the matrices' indices match (are identified). In theory, this can also be done over the tensor-product of many tensors. The partial-trace or tensor-contraction (also may be described as a quotient) corresponds to the operation:

$$\text{Tr} \left(\bigotimes A^{(n)} \right) \quad .$$

In practice the partial-trace may be computed without resorting to the complete tensor-product. The usefulness of computing the partial-trace efficiently is of high-

importance, as described in the introduction. In order to implement this operation, the information of which tensor and its specific axis/index is identified with another is required. This identification is a tedious part of partial-tracing, but this bookkeeping plays an important role in defining this operation. This operation hence may be represented by an ordered-graph or network, *i.e.* a sparse-tensor itself, as described below.

3.2 Ordered Network Structure

Let G be a graph, $V(G)$ be the vertex-array (enumeration of the vertex-set) of G , and $E(G)$ be the edge-set of G , and $(i, j) \in V(G) \times V(G)$, denoting the i th and j th vertex. Then we may define a matrix on the Cartesian-product (page 54 in Bollobás 1998):

$$A_{ij} = \begin{cases} 1 & (i, j) \in E(G) \\ 0 & (i, j) \notin E(G) \end{cases} .$$

This matrix is called the *Adjacency-Matrix*. We may make a similar argument analogously for hypergraphs and tensors. Let H be a hypergraph, $V(H)$ be the vertex-array, and $E^n(H)$ the n th order hyperedge set (a collection of n -tuples, including $n = 1$, *i.e.* the usual edge, $n = 2$). Let $t \in \prod^n V(H)$, then the n th-order adjacency-tensor is:

$$A^n[t] = \begin{cases} 1 & t \in E^n(H) \\ 0 & t \notin E^n(H) \end{cases} .$$

Note in our convention, if $t \in E^n(H)$, then all permutations of t are included in $E^n(H)$ as well, as was assumed in the graph/adjacency-matrix. The entire adjacency-structure of the hypergraph is captured by all the adjacency-tensors. Instead this structure may be captured by a single-tensor; this will require the enumeration of the edges, to form the edge-array. The resulting matrix is the *Incidence-Matrix*, and its defined over $V(H) \times E(H)$:

$$I_{ie} = \begin{cases} 1 & V[i] \in E[e] \\ 0 & V[i] \notin E[e] \end{cases} .$$

This matrix works equally well for graphs and hypergraphs. This is related to the adjacency-matrix for 2-hypergraphs (regular graphs) by matrix-product: $A_{ij} = \sum_e I_{ie}I_{je}$. Therefore using the adjacency-matrix, we may associate a graph to a matrix (2-dimensional tensor), and a n -hypergraph (a hypergraph is only n -order hyperedges) to a n -axés-tensor.

3.2.1 List of Tensors

Here we consider a equations which manipulate a many of tensors, thus for organizational purposes, let's arrange all these relevant tensors into a list. This list is enumerated/indexed for later use, but of arbitrary order. Let's suppose we have n tensors of different dimensions and sizes enumerated by a parenthesis-superscript: $A^{(n)}$, then we may form a list (an ordered set) \mathcal{A} as:

$$\mathcal{A} = [A^{(0)}, A^{(1)}, A^{(2)}, \dots, A^{(n-1)}] .$$

3.3 List-of-Labels Trace Representation

Now in order to specify the tensor-network or contraction-structure we must identify a certain tensor's indices with another tensor's indices. A common approach in *pen-to-paper* work, is to enumerate/label all tensor-indices (belonging to all tensors in the list-of-tensors) with alphabetical-letters, any matching indices are assumed to be summed over; this is the *Einstein-Summation* convention, Einstein 1916. Note this convention misses summation over any single-index. This convention may be extended by matching with an expression to the other-side of the equal sign. As any tensor expression may be equated to a single tensor (the resultant), the labeling of this tensor can supplement the Einstein-Summation convention. This is the convention in the popular Python library NumPy et al. 2020 for the `einsum` module, we shall call this the `einsum` convention. E.g. suppose we are given A, B, C tensors, we sum over any indices not appearing on the left-side (in the resultant) of the equal sign:

$$D[\alpha, \beta] = A[\alpha, i]B[\beta, i, j, k]C[i, j]$$

$$R_{\chi}^{\text{LoL}} = [\alpha i, \beta i j k, i j, \alpha \beta] \quad .$$

Therefore given N tensors in the list-of-tensors, we may specify $N + 1$ strings-of-characters (one for each tensor, and one optional entry for the output tensor) to represent the tracing-prescription. For the generic list-of-tensors, \mathcal{A} we have (for labels ℓ):

$$R_{\chi}^{\text{LoL}} = [\ell^{(0)}, \ell^{(1)}, \ell^{(2)}, \dots, \ell^{(n-1)}] \quad .$$

3.3.1 Network CONtraction (`ncon`)

Instead of letters, *e.g.* $\{\alpha, \beta, i, j, k\}$ above, the `ncon` notation, developed by Pfeifer et al. 2014, uses an array/list of labels, with the labels represented by positive integers (excluding '0') to denote internal/dummy indices, with negative integers denoting an external indices (not summed over).

3.4 Hyper-trace/Hyper-contraction

Now let's consider the identification of tensor's axes with nodes/vertices of a network. Tracing-prescriptions involving many tensors, the aforementioned list-of-labels representation (Einstein notation) may become cumbersome (in the use of too many arbitrary letters). Instead a graphical/visual form of a given tracing-prescription may be introduced. We begin by associating tensor-indices as nodes/vertices/points in our network. Nodes are connected by a *hyperedge*, a grouping indices, if they belong to the same tensor. More precisely, they are connected by a ordered-hyperedge. We give an illustration on figure 7, which shows 3 independent examples of these hypergraphs. These kinds of hypergraphs are examples of *disconnected* hypergraphs, and have the designation: *trivial multipartite hypergraph* (each disjoint hyperedge partition is a tensor). Note, this is the same structure as the *tensor-diagrams* discussed earlier. The collection of a *set*-of-nodes and a *list*-of-hyperedges forms a ordered hypergraph/hypernetwork, see chapter 1, 4, 6 in Bretto 2013. If the list-

of-tensors and their axis are ordered, then this information is captured by a list of lexicographically-ordered 2-tuples, denoting the (tensor, axis of tensor).

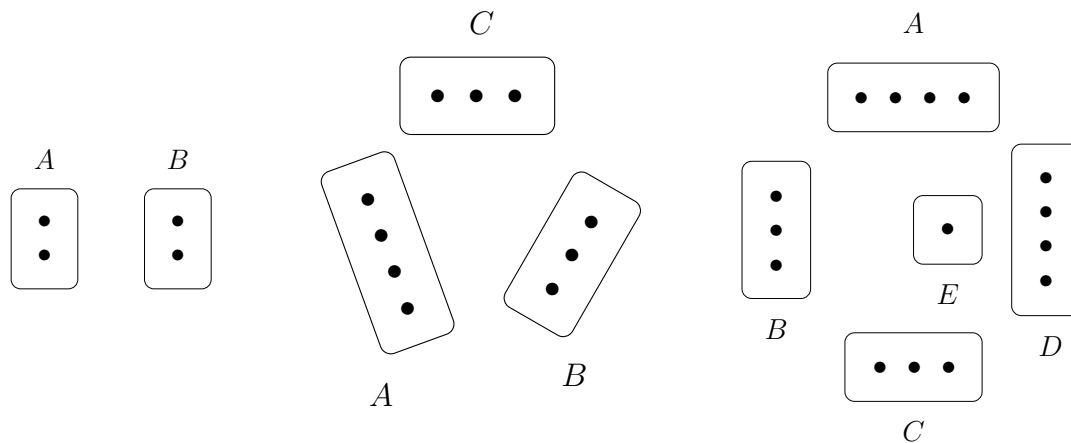


Figure 7. Three Examples of Tensor-index Hypergraphs.

3.4.1 Slicing Hypergraph

Many references refer to many kinds of different graph/network tensor representations, however this intuitive graphical picture was likely first introduced by **Penrose1971**. Graphs/Networks have two components: vertices/nodes and edges, Bollobás 1998, we previously introduced the vertices. Here we would like to represent the contraction-structure via an edge as is usually done. However, we allow for the generalization to the hyperedge (an edge connected to integer many nodes). Examples of these slicing-prescriptions hypergraphs are shown in fig. 8, which gives 3 independent examples applied to the hypergraphs in fig. 7. One crucial property is that the hypergraph should partition all the nodes (including 1-node hyperedges).

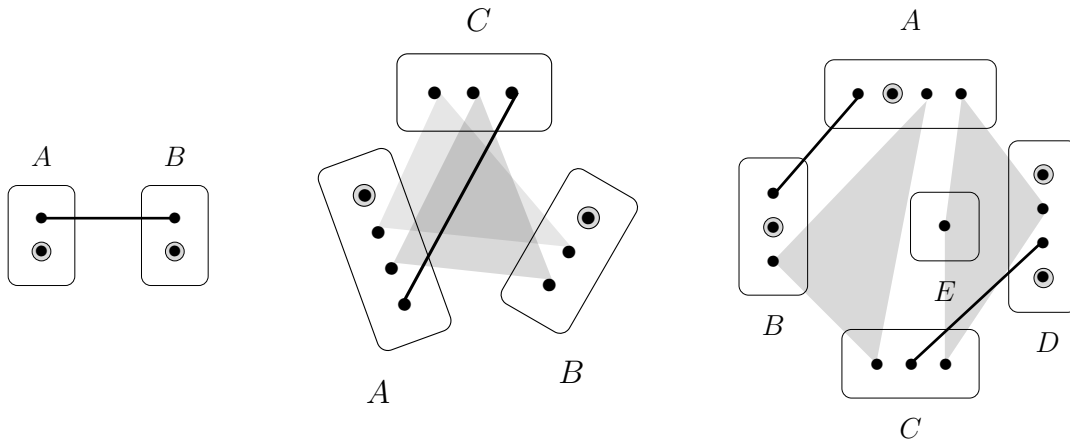


Figure 8. Three Examples of Slicing-prescription Hypergraphs.

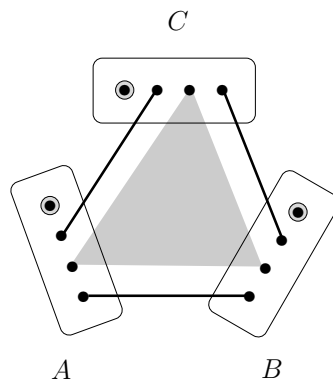
3.4.2 Sparse-tensor Representation

Now we would like to associate the graphical figures into matrices themselves (to promote computational automation). This may be done via the hypergraph/tensor adjacency-map introduced. However, this would imply a direct-sum of many adjacency-sparse-tensors, *e.g.* $A = A^1 \oplus A^2 \oplus A^3 \oplus \dots \oplus A^n$. This notation can become cumbersome, and therefore alternatively a *incidence-matrix* representation may be introduced, in this representation is by definition *sparse* if multi-edges are neglected. The incidence-matrix is a map between the node-tuples and the tracing/slicing hyperedges. That is we enumerate the hyperedges (of all orders, *e.g.* 1-edge, 2-edge, etc...) by integers. As with the `ncon` label, positive integers denote (tracing) and negative integers represent external (sliced) indices. Note that these hyperedges also

partition all axes of all the tensors. If these tensors are themselves sparse or dense, the network itself is a sparse-tensor.

3.4.3 Simplicial-complex?

A natural question when dealing with hypergraph structure is: whether we may realize *simplicial-complex* structure, defined in Hatcher 2005. In simplicial-complexes the n -hyperedge are identified with an $n - 1$ -simplex (0-node hyperedges cannot be mapped). However, unlike the generic hypergraph, simplicial-complexes are a *topological-space*, and thus have self-containing self-intersecting structure, if $a, b \in \Omega$, then $a \cap b \in \Omega$, for simplicial-complex Ω . Therefore for example a simplicial-complex may be realized by:



3.5 Dense-Sparse Partial-Trace

Here we demonstrate the partial-tracing between a purely sparse-array (S) and dense-array (D) to yield another dense-array O . The assumption that the output array is dense, O , follows that is it is essentially inevitable, else the input dense-array D was likely sparse-initially. Let's suppose both the sparse-array (S) and dense-array (D) have both kinds of indices: external and internal. Let's refer to the collection of external indices by \mathbb{I}_E and \mathbb{J}_E for both the sparse and dense-array respectively, and likewise for internal-indices \mathbb{I}_I and \mathbb{J}_I . Such that both arrays can be brought into the form (via transpositions)¹⁶: $S[\mathbb{I}_E, \mathbb{I}_I]$ and $D[\mathbb{J}_E, \mathbb{J}_I]$. Then the resulting dense-array O is $O[\mathbb{I}_E, \mathbb{J}_E]$, consisting of the external-indices only. These three tensors/array may be brought into matrix form via reshaping (see next sections). The entries of O are simply the *iterative-sum* (sum over every sparse-entry I) of the product of compositions:

$$O[\mathbb{I}_E, \mathbb{J}_E] \stackrel{\Sigma}{=} D[\mathbb{J}_E, \hat{\mathbb{I}}_I] S^{\text{data}}[I] \quad .$$

Notice the sparse internal-index tuple \mathbb{I}_I is injected/composed into the dense-array. The sub-array $D[\mathbb{J}_E, \hat{\mathbb{I}}_I]$ is multiplied by $S^{\text{data}}[I]$, and summed to the existing entries of $O[\mathbb{I}_E, \mathbb{J}_E]$. The algorithm is summarized as (e.g. \mathbb{J} represents a collectively many

¹⁶Due to syntax limitations, the implementation considered here involves reshaping all internal-indices for a given array into 1-axis, and all external-indices into another. Therefore the sparse-array becomes a sparse-matrix (a standard data structure, *i.e.* $\mathbb{I}_E \rightarrow I_E$), and the dense-array becomes a dense-matrix. The same composition-procedure is then applied to this setting.

axës, while J represents one reshaped axis):

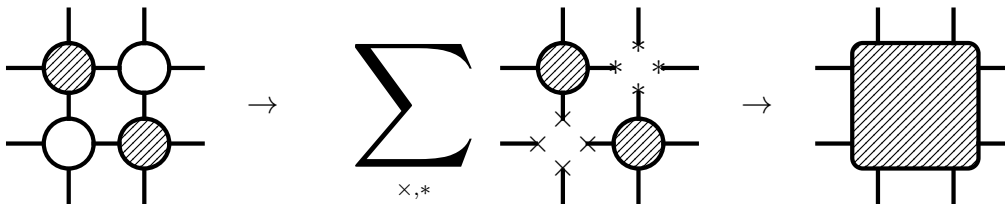
$$\begin{array}{ccccccc}
D & \xrightarrow{\text{swapaxes}} & D[\mathbb{J}_E, \mathbb{J}_I] & \xrightarrow{\text{reshape}} & D[\mathbb{J}_E | J_I] & & \\
S & \xrightarrow{\text{swapaxes}} & S[\mathbb{I}_E, \mathbb{I}_I] & \xrightarrow{\text{reshape}} & S[\mathbb{I}_E, I_I] & \xrightarrow{\text{reshape}} & S[I_E, I_I] \\
\left\{ O[\mathbb{J}_E, I_E] \stackrel{\Sigma}{\equiv} S^{\text{data}}[I] D[\mathbb{J}_E, I_E] \right\}_I & \longrightarrow & O & & & & .
\end{array}$$

Reshaping and transposing (swapaxes) for dense-arrays carry trivial computational expense, $\mathcal{O}(1)$. Although transpositions for sparse-arrays carry the same cost if we neglect the resorting to place into a canonical lexicographically-ordered form. Reshaping is seemingly an unavoidable $\mathcal{O} \sim N$ (which corresponds to the number of sparse-entries). For each sparse-entry we compute its composition into the dense-matrix as inserted into the output dense-array.

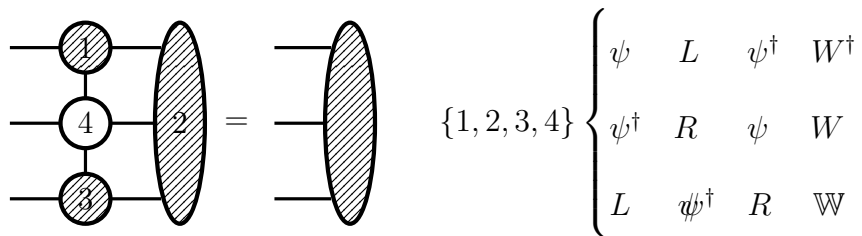
3.6 Sparse-Dense Network

In this section, we consider tensor-networks of sparse and dense tensors together. Essential mechanism is that the sparse-tensors' index-arrays select parts of the dense-array/tensor, to work with and contract, placing them at the appropriate address in the dense output array. Let's consider a network made from dense and sparse arrays/tensors. For instance, if we implement a simultaneous-dense-trace the following network its time-complexity scales $\sim N^{12}$ (with all axës containing N entries), if we do a sequential-dense-trace this may be lowered to $\sim N^{10}$. Let's suppose each of the two sparse-tensors has η sparse (nonzero) entries compared to the dense-tensor's N^4 entries. Therefore, for a simultaneous sparse-trace, this should be lowered to $\sim N^4 \eta^2$ (note

when $\eta = N^4$, we recover the simultaneous-dense-trace time-complexity). Note that $\eta \leq N^3$ to match the sequential-dense-trace time-complexity. A curious observation is that the optimal dense-partial-trace should be a least $\sim N^8$, thus this suggests that the result may be sparse itself for sufficiently small η .



Now let's consider the necessary part for the 1D-MPS/DMRG-algorithm (introduced in a later chapter). In quantum-equilibrium, $\Psi^\dagger = \Psi$, we have two tensor-networks: the MPS, $\Psi = \{\psi\} = (\Psi^\dagger)^\dagger$, with on-site pieces ψ . And the MPO, $H = \{W\}$, with on-site pieces W . It consists¹⁷ of dense-tensors $\{L, \psi^\dagger, \psi, R\}$, and sparse-tensor W . To compute the DMRG-algorithm we must compute the following tensor-contraction¹⁸.



Again for this tensor-network the dense-partial-trace has time-complexity of $\sim m^4 M^2 N^2$, however when done sequentially $\sim 2m^3 M N + m^2 M^2 N^2$. As m is usually

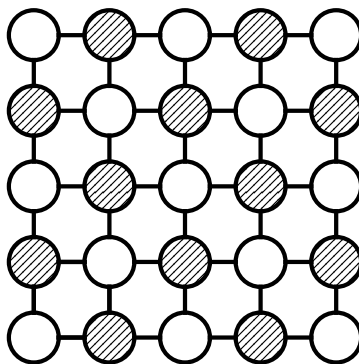
¹⁷ L and R are environmental tensors, the partial-trace of all previous sites to the left and right respectively.

¹⁸With $\psi\psi^\dagger = \text{Tr}(\psi\psi)$, the 2-site-wavefunction, and $\mathbb{W} = \text{Tr}(W W)$; the 2-site-Hamiltonian.

the largest bond-dimension, its highest power is usually the limiting factor, *i.e* the $\sim m^3$ term¹⁹. For the simultaneous sparse-partial-trace, we have two-matrix-products along the perimeter dense-tensors $\sim 2m^3$, with their addition and coalescence of $\eta \sim MN$ terms, leading to the same computational time-complexity as sequential-dense-partial-trace $\sim 2m^3MN$. The sequential-sparse-partial-trace does not lead to much improvements, because the large-bond dimension m is always along the dense-tensors.

3.6.1 Dense-Sparse Networks Continued

It appears that in order to leverage the sparse-tensors we need for them to disconnect the dense-tensor-network appreciably. Let's consider the bipartite chessboard of sparse and dense tensors.



The sparse-tensors slice the dense-tensors, and therefore the entire trace is equal to the sum of the products; one would expect this to have a time-complexity to scale $\sim \eta^n$, with n sparse-tensors with η being the number of sparse (nonzero) elements.

¹⁹For the Heisenberg-model $M \sim N^2$ (e.g. N for $\mathfrak{su}(N)$, or $N \sim 2s + 1$ for spin- s - $\mathfrak{su}(2)$), for chemical-models (less-local) we have $N = L$ sites (molecular-orbitals), then $M \sim N^2 \sim L^2$.

3.7 Sparse-Sparse Partial-Trace (algorithm 1)

This algorithm resembles the previous one, and relies on the reshape algorithm, about to be discussed. The idea of the algorithm, is to contract one edge of the partial-trace-network at a time. Each edge corresponds yields external and internal indices on every sparse-tensor. Every relevant sparse-tensor then may be reshaped and transposed natively sparse index-arrays to established sparse-matrices. The sparse-matrix operations may then be used to yield a resultant-matrix (a sparse xor dense matrix). This resultant-matrix may be reshaped/transposed back into its desired form.

3.7.1 Sparse Reshaping

In this section we would like to reshape sparse-arrays, as is commonly done for dense-arrays. *Reshaping* is the merging or partitioning of axes for a given array. Unlike dense-arrays, which store data contiguously in real-space memory, sparse-arrays store the data into 3-dense-arrays. Two of these dense-arrays: shape and array-of-tuples, control the structure of the data, here we introduce dense-operations on these two arrays to alter the structure of sparse-data. The structure of sparse-data is managed by array-of-tuples. For our discussion, let A and B be arrays-of-tuples: $A, B \subset \prod_i \mathbb{Z}_i$ (*i.e.* Cartesian-product of many integer array-subsets of \mathbb{Z}). Thus we would like to

define a function h , that maps an array-of-tuples to another array-of-tuples:

$$h : A \longrightarrow B \quad .$$

We choose to take an intermediate step, and instead solve the simpler problems of mapping tuples-to-integers and integers-to-tuples:

$$A \longrightarrow \mathbb{Z}_N \longrightarrow B \quad .$$

Ultimately, we desire a finite-pairing functions, let $\mathbb{Z}_N \subset \mathbb{N}$ such that $|\mathbb{Z}_N| \neq \infty$ (*i.e.* \mathbb{Z}_N has finitely many elements), then the pairing-function is:

$$\begin{aligned} f : A &\longrightarrow \mathbb{Z}_N \\ g : \mathbb{Z}_N &\longrightarrow A \quad , \end{aligned}$$

such that: $f(g(\mathbb{Z}_N)) = \mathbb{Z}_N$ or $g(f(A)) = A$. With the total number of dense-elements being: $N = \prod_i s[i]$. And the range of integers (in order): $\mathbb{Z}_N = \left(0 \ 1 \ 2 \ \dots \ N - 1 \right)$. Additionally, it is worth noting that the reshape-function defined here does not rely on any kind of ordering. There are many kinds of pairing-functions, we choose the *C-language* reshaping-order ²⁰.

²⁰Let i denote rows and j denote columns of A then:

$$\begin{aligned} A_{ij} = A[i, j] &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \\ \xrightarrow{\text{"C"-order}} &(1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8) = A[i'] \quad . \end{aligned}$$

3.7.2 Modified Shape

For both tuples-to-integers and integers-to-tuples maps, we need to modify the shape array, $s[i]$, into an auxiliary array $\tilde{s}[i]$. In each case the shape-array means: current-array shape xor desired-array-shape respectively. Either way, this involves permuting the shape array in a particular fashion, with a cumulative-product. The modification of $s[i]$ is the following²¹:

$$s_i = \left(s_0, s_1, s_2, s_3, \dots, s_{n-1} \quad s_n \right)$$

$$\tilde{s}_i = \left(\prod_{i \in [1,n]} s_i \quad \prod_{i \in [2,n]} s_i \quad \prod_{i \in [3,n]} s_i \quad \dots \quad s_{n-1} s_n \quad s_n \quad 1 \right) \quad .$$

Injectivity, is given by the fact that this auxiliary-shape array is strictly decreasing, *i.e.* $\tilde{s}[i] > \tilde{s}[j]$ for $j > i$. And surjectivity stems from the precise spacing of the auxiliary-array. For example, suppose we have: $a_i = \left(a_0 \quad a_1 \quad a_2 \quad \dots \quad a_n \right)$, with each a_i value having a ceiling value of s_i (the shape), our integer is, z :

$$z = a_0 \left(\prod_{i \in [1,n]} s_i \right) + a_1 \left(\prod_{i \in [2,n]} s_i \right) + a_2 \left(\prod_{i \in [3,n]} s_i \right) + \dots + a_{n-1} s_n + a_n \quad .$$

²¹In practice, this may be done with:

$$\begin{aligned} s[0] &= 1 \\ s[i] &= \text{flip}(s[i]) \\ s[i] &= \text{roll}(s[i], 1) \\ s[i] &= \text{cumulative-product}(s[i]) \\ s[i] &= \text{flip}(s[i]) \quad . \end{aligned}$$

flip, entails: $(s_0, s_1, \dots, s_{n-1}, s_n) \rightarrow (s_n, s_{n-1}, \dots, s_1, s_0)$
roll, entails: permute, via right-shift

3.7.3 Tuples-to-Integers

The conversion tuples-to-integers may be achieved by a dense-matrix-vector product, for a suitably auxiliary-shape-array (\tilde{s}_i), array-of-tuples A_I^i (with i the tuple-index), to an array-of-integers a_I :

$$a_I = A_I^i \tilde{s}_i \quad .$$

This operation thus has time-complexity of $\mathcal{O} \sim nN$, with n being the number of columns (axès), and N being the number of sparse-elements, this will have linear-scaling as usually $n \ll N$.

3.7.4 Integers-to-Tuples

Given the modified shape-array, we may do the following dense-operations to obtain the new array-of-tuples, from an array-of-integers (or 1-tuples):

$$\begin{aligned} A[I, i] &= A^I \otimes \mathbf{1}^i \\ A[:, 1 :] &= A[:, 1 :] \pmod{\tilde{s}[: -1]} \\ A[I, i] &= \left\lfloor \frac{A[I, i]}{\tilde{s}[i]} \right\rfloor \quad . \end{aligned}$$

These include, computing the integer direct-product of an array-of-integers with an array consisting of all ones, $\mathbf{1}^i$, with the size of the desired tuple. The result is A^{Ii} a 2-dimensional array. Then the modulo-division (remainder) is applied along the second axis with the modified-shape array (along a subset of entries along that axis, *i.e.* the first/zeroth entry of $A[:,0]$ is ignored as is the last shape element). Then the

complementary-function, integer-division, is applied along the same axis, but this time over all elements. Finally the resulting array is transposed to obtain the array-of-tuples into a *canonical* form (axës following the sparse-array data entry index), hence $A[i, I]$. This operation like, it's inverse, scales linearly in time: $\mathcal{O} \sim nN + nN + nN + 1 \sim 3nN$.

3.7.5 Tuples-to-Tuples

It can be easily shown that to map a tuple into another tuple of different size, using the intermediate integer is a bijective map. This is because of the well-known relation that the composition of bijective functions is itself bijective.

3.7.6 Binary Summation (bisum)

Now that we have the sparse-reshaping algorithm, we can convert any binary (consisting of two) tensor-operation into a matrix-product, for sparse-tensors this is the sparse-sparse matrix-product. This allows for the utilization of existing sparse-sparse-matrix-product algorithms. We will explain how this algorithm works. Suppose we have two sparse-tensors, A and B , with data $\{A_I^i, a_I, s_i\}$ and $\{B_I^i, b_I, t_i\}$. And we also have the sparse-index adjacency-matrix (indicating which axës of A and identified with axës in B). Then the sparse-tensors may be reshaped gathering external-axës and internal-axës together, yielding sparse-matrices: $\{A_I^\sigma, a_I, s_\sigma\}$ and $\{B_I^\sigma, b_I, t_\sigma\}$ that may be multiplied normally to yield sparse-matrix $\{C_I^\sigma, c_I, u_\sigma\}$. This matrix C may intern be reshaped into the desired tensor $\{C_I^j, c_I, u_\sigma\}$, with j being

the external-indices/axës in both A and B . More details and results are given in an appendix chapter.

3.8 Sparse-Sparse Partial-Trace (Algorithm 2)

In algorithm 1, we outsourced the sparse-sparse-matrix-product to another external algorithm. We also were limited by partial-tracing only two tensors at a time²². Here we consider its computation natively for an arbitrary amount of tensors. In order to perform this operation, we need to identify certain axës with others (in a general tensor partial-trace). Therefore it appears we desire an *intersection*.

3.8.1 Direct-Intersection

In set-theory, the operation to determine a subset which belongs to both sets is called the intersection. Let A and B be sets, then $A \cap B = C$ such that for all elements of $c \in C$ are simultaneously in A and B . The intersection is desirable, in our discussion, because inter-array partial-tracing only occurs over internal entries which agree in numerical value. For arrays (sets with some arbitrary order, with potentially duplicate entries), we desire instead the indices (from each array) which contribute to the intersecting element, we desire the *argument-intersection*. That is,

²²Therefore, requiring sequential contractions for a tensor network, *i.e.* much like the dense-array situation.

given two arrays A and B and 1 intersecting element c between them, I_A and I_B indicate the indices appearance of this element, c , in both A and B . Therefore the composition/sub-index into A and B :

$$A[I_A] = B[I_B] = c \quad .$$

Now if A and B have multiple intersecting elements, c^0, c^1, \dots, c^n , then we require an array-of-indices given by: $[I_A^{(0)}, I_A^{(1)}, \dots, I_A^{(n)}]$, and $[I_B^{(0)}, I_B^{(1)}, \dots, I_B^{(n)}]$ for B 's indices, such that $A[I_A^{(j)}] = B[I_B^{(j)}] = C^{(j)}$. If there exists duplicates for each intersecting elements, then each $I_A^{(i)}$ are list-of-indices themselves. Unlike the entries the array-of-indices is unique, and therefore if well-ordered their Cartesian-product will also be well-ordered. Leading to the net result of the direct-intersection, for 2 arrays, all the pairs (2-tuples) of indices that agree. As arrays have no restriction on duplicated elements, the natural extension for the set-intersection, to arrays is the direct-intersection.

The direct-intersection is useful because it allows us to compute the partial-trace (if we remove internal-indices, with this operation denoted by \bigcirc , from the direct-intersection, and well-order, with this operation given by the superscript \langle):

$$\text{Tr}\left(\bigotimes \mathcal{A}\right) = \left(\bigcirc \bigcap_{\text{edges}} A^{(n)}\right)^{\langle} \quad .$$

The result will contain contributions from all the possible products which contribute to the partial-trace.

We would now like to realize the direct-intersection, the intersection of elements between arrays/tensors. The direct-intersection is the direct-product over the intersecting elements of two (or more) tensors/arrays, and deletion of non-intersecting

elements. Or a particular slice/intra-intersection of their full direct-product. Visually, for two arrays this is given by a disconnected-union of many complete-bipartite-graphs, each complete-bipartite-graphs acting on the intersecting elements.



Figure 9. This Figure Shows an Example of a Direct-intersection (the Collection of All the Edges) Between Two Arrays' (Red and Blue) Elements (*i.e.* the Nodes), Represented by a Sum of Disjoint Complete Bipartite-graphs.

3.8.2 Shoelace

Suppose we have 2 arrays A and B , and we would like to compute their direct-intersection. Then we shall need to sort each A and B , in order to utilize the binary-search on both arrays. We begin by starting at one of the arrays, and searching for the first element in the other via a left-side and right-side binary-search. If found, the left-side and right-side search yields the range of values that match, else if not found the left-side and right-side search matches in value and is known as the *insertion-point*. This interval-search can be done by going between the two arrays. If we assume both arrays are sorted, then if we binary-search between the arrays, and a found element that skips over many, all these elements are by definition not included in both arrays. This even applies if the searched element is not found, as the insertion point provides another constraint to future searches. Furthermore, these

arrays need-not have unique elements, but merely sorted. Additionally, all required binary-searches become increasingly constrained. Therefore if there are M unique entries in the intersection of A and B , $|A \cap B| = M$, with N total entries in each the algorithm should have time-complexity of $\mathcal{O} \sim M \log N$.

3.8.3 Multi-Intersection (Hyperedge)

The previous algorithm applies to a 2-node edge, *i.e.* between 2-arrays. This is readily generalizable to a n -node hyper-edge, *i.e.* between n -arrays. The main idea here is to layout the arrays in an arbitrary 1-dimensional²³ order with two pivot arrays. These pivot-arrays are special as they guide the search. Suppose we start the search on an edge array A , for some element $a \in A$, we search with the adjacent array if found we proceed to search its adjacent array, this is repeated until the last array (an edge array, D). If the search makes it all the way through all arrays, then their intersecting elements are direct-producted together and saved. Else (if any search fails) $a \in A$ is searched in B for some insertion point i . This insertion point defines the next search, $d = D[i]$, and the algorithm repeats going towards edge array A . The algorithm to linking edge-to-edge may be called a *sweep*.

The algorithm is depicted in fig. 10, for the case of four list-of-tuples/arrays. On that figure, all lines (dotted, dashed, solid) are binary-searches $\mathcal{O} \sim \log N$, dashed lines are searches which “failed”, yielding the insertion point (both searches yield the

²³Suppose we have 4 arrays as in fig. 10, orders of $ABCD$, $ACDB$, $CDBA$, or any permutation should yield the same results.

same location). After a failure, one edge is compared to the other directly, the next unique element(s) in that edge start a new search. While the dotted & solid lines compute the left and right search respectively, and correspond to a successful search. Once we reach the end of any array the algorithm is finished.

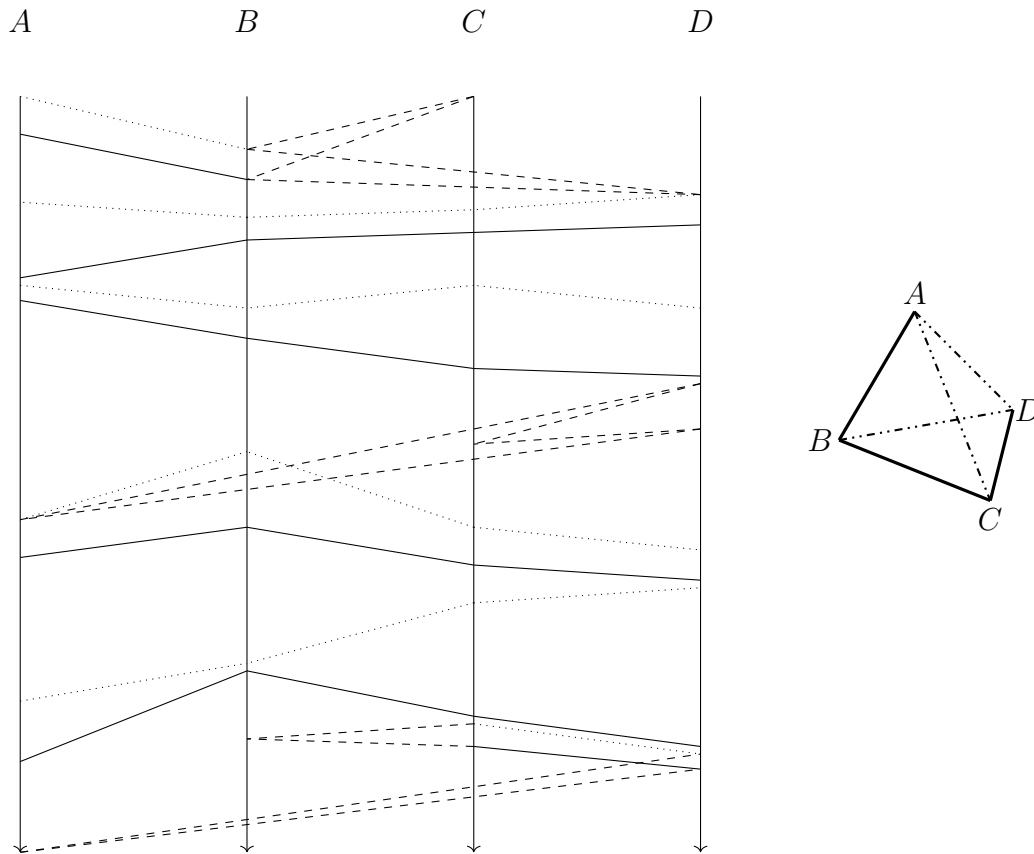


Figure 10. (left) An Example for the Lace Algorithm on 4 Lists/Arrays. (Right) Showing the Contraction-hyperedge and the Searching Path (Shown to the Left) Shown by the Solid Line.

3.9 Surjective-map

So far the direct-intersection computes the intersecting-indices associated to the ordered arrays. This is not quite what is desired, this is because the internal-indices, which are intersected, are generally in a different order than that of the external-indices. Therefore we would like to obtain the direct-intersection according to the corresponding ordered external-indices.

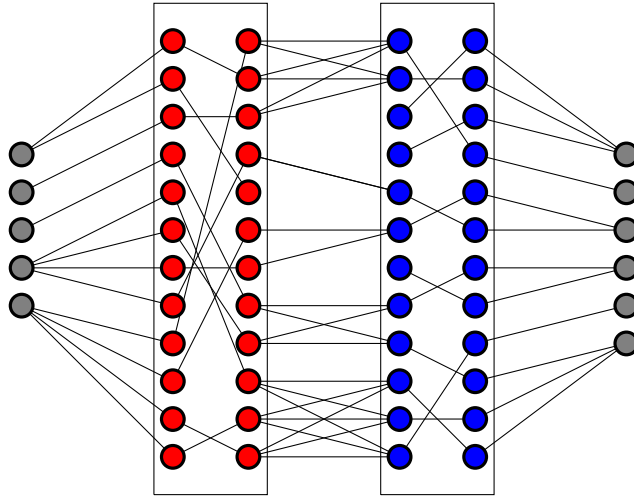


Figure 11. This Figure Shows an Example of the Complete Operation Between Two Surjective-maps, two Ordering, and a Direct-intersection Map.

An important but subtle ingredient is the surjective-map between the external-indices and internal-indices. Suppose we have sorted the internal and external-indices (F) together such that the internal-indices (D) are lexicographically-ordered $D \leq$, $D \leq ||F$ (let $||$ denote concatenation), in general this means the external-indices F are not ordered. Because the external-indices yield our result, they must be well-

ordered, and hence ordered. This may be achieved by a permutation i , to achieve partial-ordering. However, we must go further and determine which indices in the partially-ordered array are unique, these indices are given by a sorted-array u . When surjectively-composed/sub-indexed into F^{\leq} this yields $F^{<}$. This array u , cannot directly be used with elements of D^{\leq} which carry a different order and size. Therefore, we may generate an auxiliary array with the order of F^{\leq} , but with entries according to the indices of the unique entry they correspond to in F^{\leq} , e.g. without-loss-of-generality the 8th unique element in F^{\leq} all have entry 8, this auxiliary array is in bijective agreement with F^{\leq} and is named f^{\leq} . The inverse of the sorting permutation array i^{-1} may be composed to revert this into the order of D^{\leq} .

$$D^{\leq} || F \xrightarrow{i} F^{\leq} \xrightarrow{u} F^{<} \\ F^{\leq} \xrightarrow{1-\downarrow} f^{\leq} \xrightarrow{i^{-1}} f \quad ,$$

note f is in the original order of F , and hence ordered like D^{\leq} . Therefore, f provides a surjective map from every element in D^{\leq} , to an element in $F^{<}$:

$$f : D^{\leq} \rightarrow F^{<} \quad .$$

This is important because, the result, given by the external-indices must be well-ordered, whilst they are apart of potentially many products. Therefore, it can be shown that, if $d^{(0)} \subset D^{(0)\leq}$ and $d^{(1)} \subset D^{(1)\leq}$ are ordered subsets of two different sparse-array list-of-tuples, such that their corresponding external-tuples are unique then:

$$d^{(0)}[f^{(0)}] \times d^{(1)}[f^{(1)}] \subset F^{(0)<} \times F^{(1)<} \quad ,$$

is a well-ordered subset of the result.

3.10 A Word of Caution!

Although, the aforementioned algorithms seem promising (with algorithmic complexity decreasing proportional to its nonzero density), a few words are in order. Most, existing algorithms and implementations (both in hardware and software) detail with the dense contractions. These have parallelization and branch-less benefits, and thus amendable to GPU and TPU acceleration. The most tedious part of the sparse-sparse algorithms is the computation of direct-products of irregular-pieces and their subsequent summation. These irregular-pieces fundamentally hinder, at least straightforwardly, direct parallelization efforts. Additionally, the searching for unique internal-index pieces, i.e. the direct-intersection, requires a branched-program, in order to cut the irregular-pieces. Lately, sorting is essential, and these are typically non-parallelizable. Although, a particular sorting-algorithm von Neumann's stable *merge-sort* is somewhat parallelizable, and therefore essential for efficient implementation of the aforementioned algorithms.

3.11 Output Considerations

Let's consider two sparse-arrays with index-arrays A and B . For a partial-trace we have external/free and internal/dumb parts for each index-array. The unique tuples

$\bar{A}^{\text{external}}$ and $\bar{B}^{\text{external}}$ define an external dense-array $\bar{A}^{\text{external}} \times \bar{B}^{\text{external}} = C$ where the partial-trace result is defined. As long as A^{data} and B^{data} arrays do not contain 0s, then the resultant data array C^{data} also does not contain 0s (sparse), and hence is purely, 100%, dense (within that external basis). Therefore we know *a priori* if the resultant is sparse xor dense, this can be used to choose the appropriate algorithm for performance considerations.

3.12 Partial Partial-Trace

In addition, to parallelization issues, the aforementioned irregular pieces could potentially produce outer-products (direct-intersection) which exceed cache memory. A solution around this is an algorithm like Gustavson's algorithm, Gustavson 1978 which computes rows (xor columns) of the output matrix (array), and thus constraining the size of these outer-products (that are later summed).

Chapter 4

SPARSITY IN TENSOR-NETWORKS

In this chapter we will give an example where using a sparse-data-structure, namely the *tensor-network* (TN), as an excellent method to obtain approximate solutions to the exponentially-hard, NP-hard, Quantum-Many-Body (QMB) Problems. Of particular interest here are two class of models: *Heisenberg* and *Chemistry*. Both of these models have been studied with various variations and generalizations, whose solutions are of great-interest. The Chemistry-model is a fermionic-occupation model, with the chemical-Hamiltonian²⁴. While the Heisenberg-model intrinsically has Nearest-Neighbor (NN) interactions. However, our primary focus is on the generalized Heisenberg-model, and will be our prototypical example. This may be solved with 3 levels of detail: Exactly, DMRG/TN, or via Mean-Field. Furthermore, we will give the sparsity-level of solutions to this model based on the system's size, and discuss results as it relates to the entanglement-entropy.

4.1 The Heisenberg Model

The Heisenberg model, defined in Heisenberg 1928, was introduced for the purpose of explaining ferromagnetism, it is a generalization of the earlier Ising Model. The setting of the model is on a lattice of spins. With their fundamental interaction of

²⁴This is meant to model electron distribution in the Adiabatic approximation.

spin-spin interactions, because this interaction strength decays exponentially with real-distance, a crucial approximation is the Nearest-Neighbor (NN) interaction, yielding a mesh (interaction edges connecting nearby lattice points). The original model was on a 1-dimensional linear-chain of spin- $\frac{1}{2}$ particles, and has been solved using the Bethe-Ansatz, Bethe 1931. Despite its model's simplicity, the Heisenberg model has led to many amendments, as shown in Pauli 1927 and Tomonaga 1997. We may assert a slight generalization with J_{ij} being a *weighted*-adjacency-matrix over a general-graph/network (i, j being site indices, for eq. 4.1, only going over upper-triangular part of the matrix, else a factor of $\frac{1}{2}$ is produced):

$$H_{\alpha\beta} = \frac{1}{2} J_{ij} S_{\alpha\gamma}^{ia} \bar{S}_{\gamma\beta}^{ja} \quad . \quad (4.1)$$

for a 1D spin-chain we have $J_{ij} = \delta_{i,i+1}$. With Latin-indices a, b are used to denote the Lie-algebra generators, *e.g.* the Pauli matrices. Where S^{ia} is the spin-operator of the system over the system for sites i and Lie-Algebra element a . With all the Lie-Algebra generators, λ_{pq}^a , located i th entry of

$$S_{\alpha\gamma}^{ia} = \left(\mathbb{1} \otimes \cdots \otimes \mathbb{1} \otimes \lambda_{ith}^a \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} \right)_{\alpha\gamma} \quad (4.2)$$

With Greek-indices (α, β) are used to denote the total-product of spinor-indices, *i.e.* over all sites, system-spinor indices. We wish to solve, the Schrodinger-equation:

$$H_{\alpha\beta} \Psi_\beta = \mathcal{E}_\alpha \Psi_\alpha \quad .$$

This model is an example of an exponentially-complex problem, whose Hamiltonian scales to $2^L \times 2^L$, with L sites. This Model-comes in two varieties depending on the exchange-integral's sign, to yield the antiferromagnetic xor ferromagnetic case.

4.1.1 Exchange Interaction

Let \cdot be the inner/dot-product, and $@$ be the matrix-product, then the spin-exchange interaction is given by:

$$S_{\alpha\gamma}^{ia} S_{\gamma\beta}^{ja} = S_i \cdot S_j = \sigma_i^x @ \sigma_j^x + \sigma_i^y @ \sigma_j^y + \sigma_i^z @ \sigma_j^z \quad .$$

$$\sigma_i^x = \mathbf{1} \otimes \cdots \otimes \mathbf{1} \otimes \underbrace{\sigma^x}_{i\text{th}} \otimes \mathbf{1} \otimes \cdots \otimes \mathbf{1} \quad .$$

Without-loss-of-generality, let's consider a 6-site-model, and examine the σ^x term between site 0 and 1:

$$\begin{aligned} \sigma_0 @ \sigma_1 &= (\sigma^x \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1}) @ (\mathbf{1} \otimes \sigma^x \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1}) \\ &= (\sigma^x @ \mathbf{1}) \otimes (\mathbf{1} @ \sigma^x) \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \\ &= \sigma^x \otimes \sigma^x \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \quad . \end{aligned}$$

Above is possible because of the ability to pass the matrix-product element-wise through the tensor-product.

$$\sigma_0 @ \sigma_1 = (\sigma^x \otimes \sigma^x \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1}) @ (\sigma^x \otimes \sigma^x \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1})$$

The 1-dimensional spin- $\frac{1}{2}$ representation of $\mathfrak{su}(2)$ model has been solved using the combinatorial Bethe-Ansatz, Bethe 1931.

4.1.2 2-site Heisenberg Example

Let's calculate the 2-site Heisenberg-model explicitly for a demonstration. We begin with our adjacency-matrix, describing the connectivity of our sites:

$$J_{ij} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Next we construct our spin-matrices:

$$S_{\alpha\beta}^{0a} = \mathbf{1} \otimes \sigma^a = \left(\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & -i \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \right)$$

$$S_{\alpha\beta}^{1a} = \sigma^a \otimes \mathbf{1} = \left(\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \right)$$

Clearly these are sparse, and may be traced with the adjacency-matrix to yield the Hamiltonian:

$$\begin{aligned}
 H_{\alpha\beta} &= \frac{1}{2} J_{ij} S_{\alpha\gamma}^{ia} S_{\gamma\beta}^{ja} \\
 &= \frac{1}{2} S_{\alpha\gamma}^{0a} S_{\gamma\beta}^{1a} \\
 &= \pm \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .
 \end{aligned}$$

The \pm sign on the Hamiltonian-matrix determines the model-type: $(-)$ for ferromagnetic and $(+)$ for antiferromagnetic. Either way this matrix may be easily diagonalized, and yields spectrum:

$$\begin{aligned}
 \mathcal{E}_{\text{antiferromagnetic}} &= \left(-\frac{3}{2} \quad +\frac{1}{2} \quad +\frac{1}{2} \quad +\frac{1}{2} \right) \\
 \mathcal{E}_{\text{ferromagnetic}} &= \left(-\frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \quad +\frac{3}{2} \right)
 \end{aligned}$$

These eigenvalues correspond to two types of eigenstates:

$$\left. \begin{array}{l} |1, 1\rangle = \uparrow\uparrow \\ |1, 0\rangle = \frac{1}{\sqrt{2}}(\uparrow\downarrow + \downarrow\uparrow) \\ |1, -1\rangle = \downarrow\downarrow \end{array} \right\} s = 1 \quad (\text{triplet}) \quad \left. \begin{array}{l} |0, 0\rangle = \frac{1}{\sqrt{2}}(\uparrow\downarrow - \downarrow\uparrow) \end{array} \right\} s = 0 \quad (\text{singlet}) .$$

4.2 Generalized Heisenberg-model

The original Heisenberg-model may be generalized in a variety of ways. Principally, this is through the generalization of the Pauli-matrices to some other Lie-algebra,

namely representations of $\mathfrak{su}(N)$. As we have infinitely possible representations for a given $\mathfrak{su}(N)$ -algebra enumerated by the spin s (over the half-integers) this is also a source of generalization. Therefore, the s -spin isotropic $\mathfrak{su}(N)$ Heisenberg-model, as shown in Affleck 1985; Gauthé 2019²⁵ is denoted: $[\mathfrak{su}(N)_{2s}]$, and is our prototypical QMB-problem we wish to solve.

Furthermore, we may introduce anisotropy, this may be done in the original Heisenberg-model (*e.g.* XY or XZ models), however, in the generalized version this amounts to: $J_{ij} \rightarrow J_{ij}^{ab}$ or for a 1D spin-chain: $J_{ij}^{ab} = \delta_{i,i+1} \delta^{ab}$. This newly expanded adjacency-matrix may be referred to a Killing-form J_{ij}^{ab} , defined on all the network/graph's edges (ij) . Ultimately our model becomes:

$$H_{\alpha\beta} = \frac{1}{N} J_{ij}^{ab} S_{\alpha\gamma}^{ia} \bar{S}_{\gamma\beta}^{jb} \quad , \quad (4.3)$$

$$H_{\alpha\beta}^{[\mathfrak{su}(N)_k]} = \frac{1}{N} J_{ij} S_{\alpha\gamma}^{ia} \bar{S}_{\gamma\beta}^{ja} = \frac{J_{ij}}{N} (\mathbb{S}_i \cdot \mathbb{S}_j) \quad , \quad (4.4)$$

with spin-matrices $S_{\alpha\gamma}^{ia}$ generalized with the Lie-algebra generators (of representation $\frac{k}{2}$) in-place of the Pauli-matrices, see eq. 4.2.

4.2.1 Generalization & Comments

Above eq. 4.4, may be generalized to include kinetic-energy and potential-energy 1-body terms, or additional pair-wise terms, etc...:

$$H_{\alpha\beta} = H_{\alpha\beta}^{[\mathfrak{su}(N)_k]} + T(S) + V(S) + V(S, S') + \dots$$

²⁵We consider this to the antiferromagnetic Heisenberg model, unless other-wise stated.

These new models lead to new physics and phases, that are no longer known to be analytically-solvable, e.g. with the Bethe-Ansatz. In particular, Haldane 1983, argued that integer Heisenberg spin-chains are insulating. However, because of NN-interactions the Heisenberg-Hamiltonian is very sparse, is there anything that be can done to leverage this? In the brilliant work of 1992 by White 1992, showed how this can be done with an algorithm called: Density-Matrix-Renormalization-Group (DMRG). We will know describe the generalization of his idea in modern (2023) terminology²⁶.

4.2.2 $[\mathfrak{su}(N)_1] \cong [[\mathfrak{su}(2)_{N-1}]]$

For the generalized Heisenberg-model: $SU(N) \cong SU(2; s = \frac{N-1}{2})$ with additional exchange-terms²⁷, this is shown in Beach et al. 2009

$$H_{\alpha\beta} = \prod_{\ell=1}^{2s} \left(1 - 2 \left(\frac{s(s+1) + S_{\alpha\gamma}^{ia} S_{\gamma\beta}^{ia}}{\ell(\ell+1)} \right) \right) .$$

²⁶Although, this was motivated by previous work by Wilson, Baxter and transfer-matrix-technique to factor partition-functions, e.g. 2D Ising-Model, White's work was the phase-transition in our understanding.

²⁷Like seen in the Uimin-Lai-Sutherland, Majumdar-Ghosh, Affleck-Kennedy-Lieb-Tasaki (AKLT) models.

4.2.3 Let's Consider $N = 3$ and $s = 1$

$$H_{ij} = (1 - (2 + S_i \cdot S_j)) \left(1 - \left(\frac{2 + S_i \cdot S_j}{3} \right) \right)$$

$$3H_{ij} = -1 + 2 S_i \cdot S_j + (S_i \cdot S_j)^2 \quad .$$

Many 1-dimensional models have exact solutions, and therefore these models act as an effective playground for testing numerical methods for higher dimensional situations.

4.2.4 Low-energy Excitations

The Heisenberg-model above, $[\mathfrak{su}(N)_k]$, is related to a couple of well-studied models in Conformal-Field-Theory (CFT): N -free (1+1)D Dirac-Fermion-model and the Wess–Zumino–Novikov–Witten (or WZW) model. The WZW-model is defined on a Lie-group manifold, *e.g.* $SU(N)$, a coupling-constant, λ , and integer level k . While, the N -free-(1+1)D-Dirac-Fermion-model has $U(N)$ symmetry which may be decomposed into $U(1) = SU(N) \times U(1)$, for spin-sector and charge-sector parts respectively. This spin-sector part is equivalent, Fradkin 2013, to the WZW($\mathfrak{su}(N)_1$)-model at the fixed-point²⁸: $\lambda^2 = 4\pi/k = 4\pi$. Furthermore, it can be shown in Fühlinger et al. 2008, that the low-energy limit of the Heisenberg-model is the WZW-model: $[\mathfrak{su}(N)_k] \sim \text{WZW}(\mathfrak{su}(N)_k)$. This is nice because WZW/free-Dirac-Fermion

²⁸A relation between the coupling constants found via beta-function techniques.

models have been thoroughly studied, and therefore many exact results exist which help guide/calibrate the DMRG computations.

4.3 NN-Factorization

The intra-operator connectivity denote interactions of the Hamiltonian. These

The intra-operator connectivity of these models are typically given via a mesh (an infinite graph) or finite graph/pair-wise-network. In our consideration, we also make the restriction to pair-wise interactions. This yields a network or graphs, that describe the connectivity of the model. These models ultimately yield a Hamiltonian matrix, of the form:

$$H_{ij} = J_{ij}^{ab} A_a^i B_b^j \quad .$$

We shall demonstrate how to factorize this Hamiltonian into matrix-products (partial-tracing over two axes) of the site-tensor-operators:

$$H = \text{Tr} (WW \cdots W) \quad .$$

This has been extensively studied for purely 1-dimensional linear configurations in the DMRG literature, and this factorization is known as the Matrix-Product-Operator (MPO). It is usually given in two distinct ways (with first vectors transposed, for

visualization):

$$\begin{pmatrix} 1 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 & \dots\dots & 0 \\ & & \vdots \\ & & \vdots \\ & & 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 1 \end{pmatrix} \quad (4.5)$$

$$\begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 & & \\ \vdots & & \\ \vdots & & \\ 0 & \dots\dots & 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \quad . \quad (4.6)$$

Where empty spaces are zero, and \dots represent all pair-wise interactions between two sites, and 0 represents the non-trivial-zero. These forms are usually called upper-triangular and lower-triangular respectively. However, here we will designate these by a different name. Let $L = \begin{pmatrix} 1 & \dots\dots & 0 \end{pmatrix}$ and $R = \begin{pmatrix} 0 & \dots\dots & 1 \end{pmatrix}$, eg ‘1’ on the left xor ‘1’ on the right. Then the upper-triangular-matrix is represented by LR and the lower-triangular-matrix is RL (these L/R names are arbitrary, *e.g.* may be called $+/-$). A few observations are in-order, if we matrix-multiply many of these matrices (all the same, suppose of the RL type) we obtain:

$$(RL)^n = (RL)(RL)\dots(RL) = \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ \sum_i^n (\Sigma_i) & \dots\dots & 1 \end{pmatrix} = RL + \begin{pmatrix} \\ \\ \\ 1 \end{pmatrix} \left(\sum_i^n (\Sigma_i) \right)$$

where $\Sigma_i = L \cdot R$, ie the full pair-wise interaction. This recursive operation, with the cumulative sum saved in the original nontrivial-zero location, allows for the

factorization of the NN-Hamiltonians via an MPO, and thus efficient-evaluation via the DMRG algorithm. This is pleasant result, next we would like to generalize this for an arbitrary pair-wise network/graph, however now we run into some complications. The aforementioned matrices may only connect two sites at a time, we cannot connect N sites to a single site. In graph-theory, this node-connectivity is known as the *degree*. We would like to expand this node, this may be intuitively represented by a higher-dimensional-matrix or tensor. There we wish to understand the pattern of their construction.

Visually the aforementioned matrix may be visualized by a square, whose entries of interest are those which belong to the perimeter of the matrix. The nontrivial-zero element only belongs to the upper-right and lower-left entry. This begs the question, whether it is possible that the nontrivial-zero belongs to the upper-left and lower-right elements too? It turns out the answer is yes²⁹, and these matrices may be called LL and RR respectively:

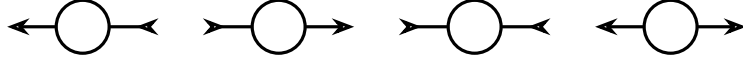
$$(LL) = \begin{pmatrix} 0 & \dots\dots & 1 \\ \vdots & & \\ \vdots & & \\ 1 & & \end{pmatrix} \quad (RR) = \begin{pmatrix} & & & 1 \\ & & & \vdots \\ & & & \vdots \\ 1 & \dots\dots & & 0 \end{pmatrix} . \quad (4.7)$$

Note $R@(LL)@R = H_3$ and similarly $L@(RR)@L = H_3$, the 3-site 2-NN-Hamiltonian H_3 . Therefore, we may create the TNO with any 4 of these matrices, note however in order to combine them with an inner-product we must *kiss* a L -axis with a R -axis of another site-matrix. Therefore we observe the following pattern:

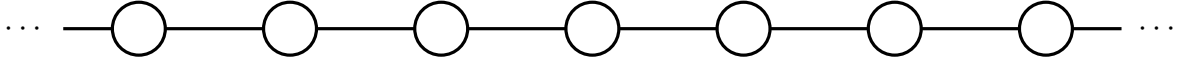
²⁹And required for Periodic-Boundary-Conditions.

potentially interesting elements exist on the perimeter of this site-matrix-operator.

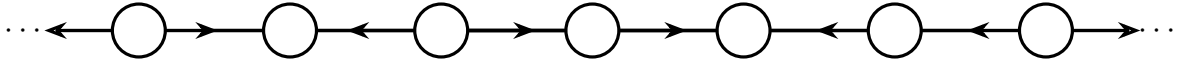
Graphically, we may associate these matrices to (RL, LR, LL, RR) respectively):



Now let's return to our chain MPO model:



This may be decomposed into an oriented-graph (directed-graph without multi-edges), in order to define the matrices above:

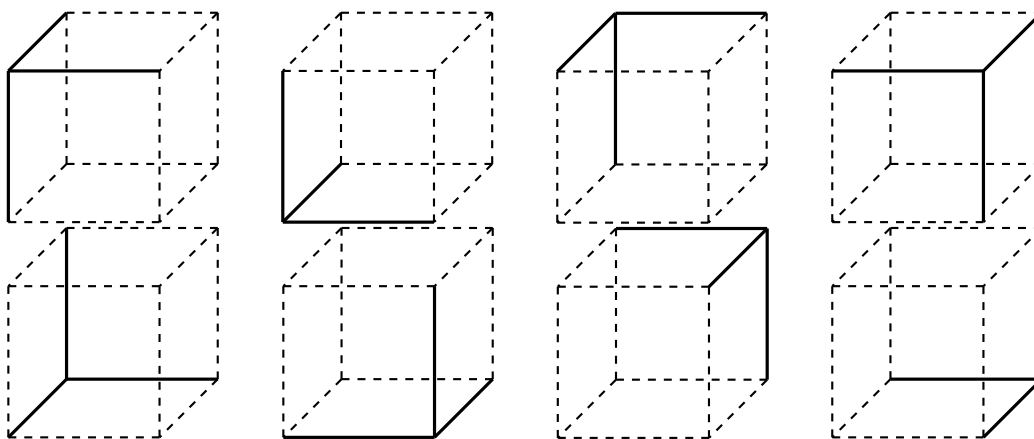


Ultimately, any oriented (*directed*)-graph is a valid factorization, in-fact this is 1 of 2^{edges} many. All these are valid factorizations of the system. However, there are two configurations that yield only 1-type of matrix (completely-consisting of LR xor RL), these are the factorizations widely used in the MPO literature, for example **pirvu2010**; Verstraete, Murg, and Cirac 2008, and many other sources (around the work of Verstraete).

4.4 Tensor Operator Element

Now let's focus on the construction of the tensor connecting a site to 3-others, this is a 3-dimensional tensor W_{ijk} . After a contraction of any of the axes, we

shall obtain a 2-dimensional matrix like before. Therefore this suggests a similar structure as before. We may guess that the nontrivial elements lie on the edges of a 3-dimensional cube, with nontrivial-zeros potentially occupying each vertex. This hypothesis is once-again correct. We may construct tensors with all 8 parities $\{LLL, LLR, LRL, RLL, RRL, RLR, LRR, RRR\}$. Their direct visualisation is given below (dark lines indicate the non-zero entries, while dashed-lines indicate the tensors' dense-array shape).

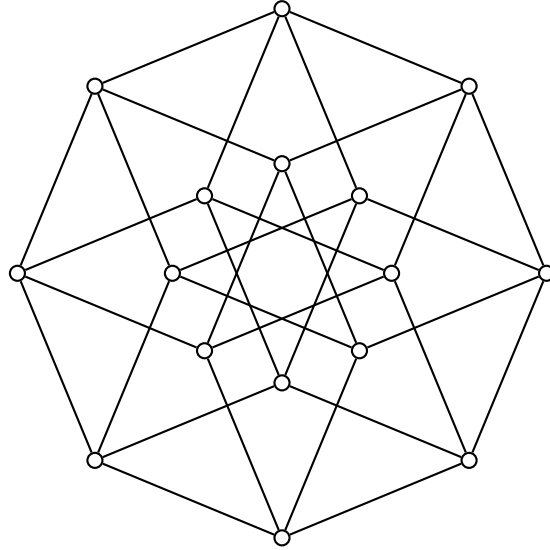


With 3-edges emanating from this nontrivial-zero having this same pattern. What about an axis connected to 4 other sites or more? for this we have to consider the hyper-cube graph for some visualization and intuition for general higher dimensional tensors.

d-Hypercube-graph Perspective

We started with visualization with a square and a cube, this generalizes for a generic hypercube. The *d*-hypercube (*d*-dimensional) graph is a graph on 2^d vertices,

such that each vertex has the same connectivity/degree, *i.e.* *regular*, with $2^{d-1}d$ edges. Defined by the n -fold-Cartesian-product on the 2-vertex complete-graph K_2 , *i.e.* $Q_n = \prod^n K_2$.



Any of the vertices of the d -hypercube can house the-nontrivial-zero, which once-defined can uniquely define all the d edges. Therefore the perimeter for an d -dimensional hypercube is its 1-dimensional skeleton, *i.e.* graph.

d -Bitstrings

In computer-science, bits are objects which take two values, *i.e.* Boolean. Our L and R decomposition has the same character, let $L \leftrightarrow 0$ and $R \leftrightarrow 1$. Each axis of a tensor has an orientation, the collection of all axes yields an array-of-orientations, this will be referred to as a *bitstring*. For instance, we may associate each entry in

the following array to an axes in a tensor (denoting its orientation):

$$[L, R, L, L, L, R, L, R, R, \cdots R, R, R] \leftrightarrow [0, 1, 0, 0, 0, 1, 0, 1, 1, \cdots 1, 1, 1] \quad .$$

4.5 Fundamental-Representation of $\mathfrak{su}(N)$

4.5.1 Complex Representation

The complex-representation of $\mathfrak{su}(N)_1$ is the following straight-forward generalization of the Pauli-matrices. Details may be found in well-known book by Georgi 1999. For the Lie-algebra $\mathfrak{su}(N)$, we have $N^2 - 1$ generators. There are two kinds of generators which make up the complete Lie-algebra: Cartan ($N - 1$ mutually commuting generators) and Non-Cartan.

Cartan

In $\mathfrak{su}(N)_1$, the Cartan-operators take the form of completely diagonal matrices. The following matrices G_i are constructed for $i \in [2, N]$:

$$G_i = \frac{1}{\sqrt{\frac{(i-1)i}{2}}} (\delta_{pq}^{i-1} \oplus (1 - i)) \quad .$$

Non-Cartan

Within $\mathfrak{su}(N)_1$, the Non-Cartan generators take a simple form of Hermitian Matrices taking a single off-diagonal entry at a time. The $\frac{N(N-1)}{2}$ purely matrices take the following form:

$$\begin{pmatrix} & 1 & & \\ 1 & & & \\ & & & \\ & & & \end{pmatrix}, \begin{pmatrix} & & 1 & \\ & & & \\ 1 & & & \\ & & & \end{pmatrix}, \dots, \begin{pmatrix} & & & 1 \\ & & & \\ & & & \\ 1 & & & \end{pmatrix}, \begin{pmatrix} & & & & \\ & & & & 1 \\ & & & & \\ & & & & \\ 1 & & & & \end{pmatrix}, \dots$$

With the $\frac{N(N-1)}{2}$ imaginary versions with i in the lower-triangular part, and $-i$ in the upper-triangular part. These matrices generalize the 2×2 Pauli-matrices:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & i \\ 0 & -1 \end{pmatrix} \quad .$$

4.5.2 Adjoint Representation

Given the generators λ_i , they must satisfy the following commutator:

$$[\lambda_{pr}^a, \lambda_{rq}^b] = f^{abc} \lambda_{pq}^c \quad ,$$

for structure-constants f_{abc} with respect to the Lie-algebra $\mathfrak{su}(N)$. These structure-constants, may thus be found given the fundamental-representation by:

$$f^{abc} = -\frac{i}{4} [\lambda_{pr}^a, \lambda_{rq}^b] \lambda_{qp}^c \quad .$$

These structure-constants f_{abc} , subsequently define a real-representation of the Lie-algebra itself, known as the adjoint-representation:

$$\Lambda_{bc}^a = -i f_{bc}^a \quad .$$

4.5.3 Jordan-Wigner Transformation

The Jordan-Wigner (JW) transformation maps spins (Lie-algebra generators) to Fermions. Explicitly, it transforms the off-diagonal generators into creation/annihilation operators the generalized $\mathfrak{su}(N)$ -JW transformation is given by Yu and Ge 2016.

4.6 Exact-Diagonalization

Now we would like to consider the solution of linear-problems, *e.g.* $H\Psi = B$ for Ψ , these are ubiquitous in physics and especially in quantum-theory. Ultimately, these problems amount to diagonalizing $H \rightarrow H'$ is a suitable eigenbasis, $H' = UH U^\dagger$. This diagonal version yields the solution of the aforementioned linear-problem, and H' 's diagonal-elements are called eigenvalues, while U 's contain the corresponding eigenvectors Ψ , and our beloved many-body wavefunction. This is the Exact-Diagonalization (ED) method, and corresponds to the numerically-exact solution (within machine-precision). In practice, as mentioned our matrices, H and Ψ grow exponentially as the problem scales linearly, and thus this problem quickly becomes intractable. However,

as we shall see in a bit, not all ED-algorithms are created equal. We will start by briefly mentioning the naive explicit-matrix-ED method as it is the most straight-forward. Next, we will mention the iterative-methods, specifically the Davidson-method, and discuss how this transposes the ED problem into a matrix-vector product. Finally, we discuss how to realize this matrix-vector-product under a factorization of the matrix H , via tensor-decompositions for a linear-chain-structure (Matrix-Product-Operator) and a generic-structure. This ultimately, reduces into many sparse-matrix/tensor and dense-vector contractions. In contrast, the ED procedure we cannot make assumptions about Ψ 's internal structure, *i.e.* correlations, and therefore it must be assumed to be a dense-tensor. Because Ψ is dense, and scales exponentially with system size the major limitation of the ED-methods is the physical-memory storage of Ψ and its contraction-intermediaries.

4.6.1 Explicit-Matrix Diagonalization

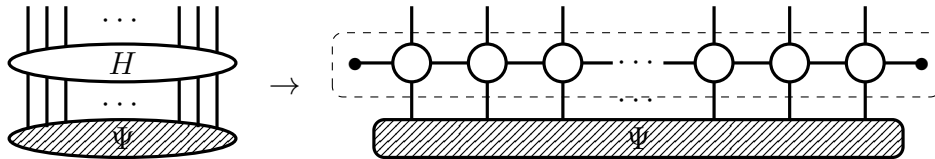
For this method, we need to explicitly construct the matrix, we wish to diagonalize. Let's suppose we consider the stereotypical QMB system, our L -spin spin-chain (potentially multidimensional). The resulting Hamiltonian-matrix is of dimension: $2^L \times 2^L$, even for modestly-large numbers this becomes intractable to densely-store. Furthermore, naïve diagonalization scales as $\mathcal{O} \sim N^3$, therefore the time-complexity is $\mathcal{O}_{\text{time}} \sim 2^{3L}$ with space-complexity $\mathcal{O}_{\text{space}} \sim 2^{2L+1}$. We may notice this matrix is sparse, and use sparse-diagonalization algorithms for a slight improvement; with time-complexity is $\mathcal{O}_{\text{time}} \sim 2^L$ and space-complexity $\mathcal{O}_{\text{space}} \sim 2^{L+1}$.

4.6.2 Iterative Diagonalization

Unlike the explicit-matrix-diagonalization scheme, which obtains all eigenstates, we may opt to select a few low (or high) lying eigenstates. These may be obtained one at a time using Krylov-subspace, *iterative-subspace*, methods, *e.g.* the Davidson-Algorithm, Davidson and Thompson 1993; Saad 2003. This transposes the diagonalization-problem into to a partial-tracing problem (many effective matrix-vector products). Furthermore, we view the matrix, as an operator, and hence we never have to explicitly construct it. If this operator can be decomposed into a MPO/*tensor-network* structure, we can compute the partial-traces sequentially. Such that we can avoid ever constructing the actual operator explicitly (in either sparse xor dense format). If H has some known symmetry, the eigenvector Ψ , may be broken into a block-sparse structure, and independently diagonalized in each sector.

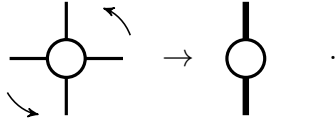
4.6.3 MPO-ED (linear-chain)

In order to illustrate our strategy for the general-case, lets attempt solving models exactly represented by an MPO. For an MPO, our problem reduces to:

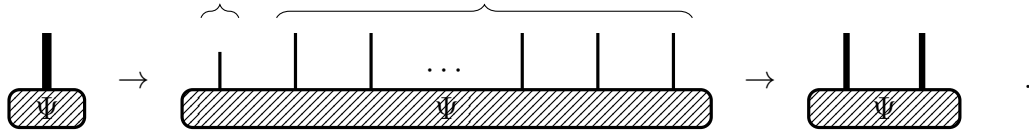


Note that the intra-contraction of the MPO tensor-network yields the explicit exponentially-large Hamiltonian tensor. The linear-chain is an example of the most-sparse-yet-fully-connected-network. Because its simplicity we can implement the

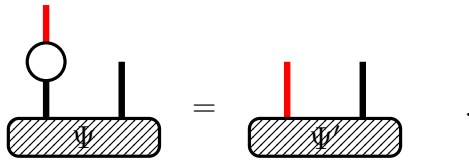
algorithm assuming this uniformly linear structure. It can be shown that this algorithm may be reduced to the sparse-and-dense-tensor products. If sparse-partial-contractions are used we may directly obtain the sparse-representation of this operator, this may be sequentially reshaped into the usual sparse matrix-representation of this operator. Again explicitly-building the operator is a major space/memory ask. Instead, because of our factorization of H , we may sequentially apply each MPO-element. To begin, the MPO-element being a sparse-tensor may be reshaped into a sparse-matrix (combining a physical-index to an intra-operator index, the reshaping yields: $(N, N, M, M) \rightarrow (N \times M, N \times M)$):



Next, lets reshape Ψ to become a dense-matrix:

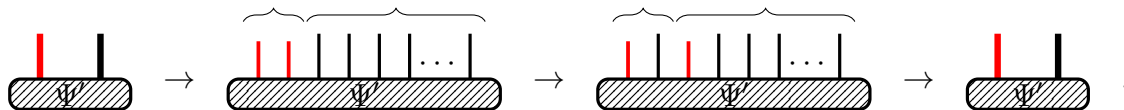


Given the reshaping above, we introduce the *zipper*-algorithm, in which MPO-elements are sequentially applied on the trial-vector Ψ , from start-to-end. We begin, by computing the sparse-dense matrix-product (between the 1st MPO element) and the trial-vector Ψ :

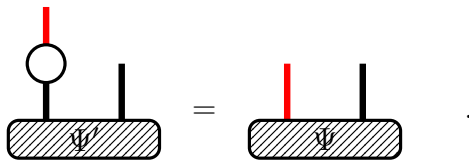


An important observation is that Ψ' contains another axis, corresponding to an intra-MPO index, and therefore the number of elements of Ψ' is greater by the size

of this intra-MPO axis, $|\Psi| < |\Psi'|$. Next, we reshape- Ψ' , swap axes, and reshape back-into a dense-matrix:



Note that the trial-vector, Ψ , is dense, and may be reshaped and axes-swapped at $\mathcal{O} \sim 1$ expense. The sparse-dense matrix-product is then recomputed for the next MPO-element, this process is repeated until we've swapped all axes, *i.e.* the end of the MPO chain in which no intra-MPO indices are left. The final step looks like:



The major limitation of this algorithm is the physical-memory storage requirements of Ψ and its partial-contraction Ψ' , with $\mathcal{O}_{\text{memory}} \sim MN^L$ for a chain of L sites.

4.6.4 Generic-network ED

In the previous subsection, we considered the case of a linear-chain. Because of its homogeneous and simple structure, the zipper-algorithm can exploit these similarities, such that the ED-algorithm is simple itself. Now we would like to consider a complication, the ED of generic tensor-network-operator (TNO). In order to facilitate this over an arbitrary-network, we require knowing the optimum-contraction-path and a tracing-module that returns the index/axis-label (which nodes correspond to

which edge) *on-the-fly*. Let's define our tensor-network:

$$H = \text{Tr} \left(\bigotimes W^{(i)} \right) \quad .$$

Pair-contraction

Often the simultaneous-contraction of many dense-tensors is best done pair-wise. Therefore we sequentially apply the pieces of the TNO to the wavefunction Ψ to create intermediate wavefunctions Ψ' :

$$\begin{aligned} \Psi' &= \text{Tr} (W^{(i)} \Psi) \\ \Psi' &= \text{Tr} (W^{(j)} \Psi') \\ &\vdots \\ \Psi &= \text{Tr} (W^{(k)} \Psi') \quad . \end{aligned}$$

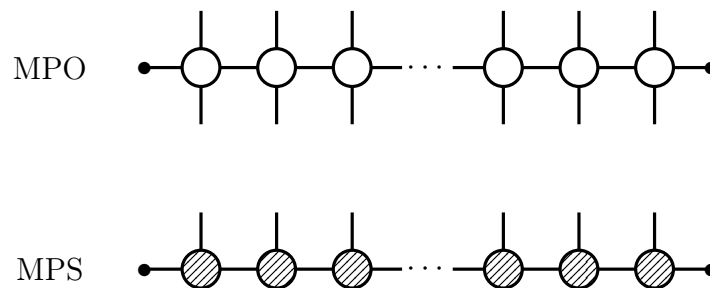
The contraction-algorithm efficiency depends on the order W 's are applied. The optimal-contraction path is an NP-hard problem. This problem may be given by a weighted network, with weights being the width of each contraction-edge. An approximate solution is provided by the *Greedy Algorithm*. Greedy Algorithm starts with node with lowest degree, and then follows with a Breath-First-Search (BFS) with lowest graph-node degree. The major limitation of this algorithm is the memory of the intermediate wavefunctions Ψ' . This is especially true for higher-degree (more edges per node) networks underlying the Tensor-Network-Operator (TNO). In the linear-chain we have an intermediary memory of $\sim M(N^L)$, because of an intermediate TNO external-axis/index of size M . For instance, for the complete-graph on n nodes,

K_n , the pair-wise-contraction leads to an intermediate- Ψ of size $\sim M^{\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil} (N^L)$. Therefore, it should be optimal for a purely-sparse intra-TNO partial-trace to reduce intra-TNO connectivity.

4.7 Approximations to Ψ

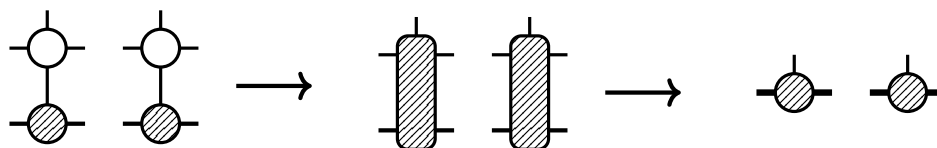
We reviewed in the last section how to exactly solve these QMB problems, including with Hamiltonians/Operators which exponentiate, but may be exactly-factored into a *base* factors, *i.e.* $H = W^\otimes$. We noticed that ED algorithms were able to be improved with this exact-factorization. However, we are still limited by the unassuming nature of Ψ , this tensor is still exponential in size.

Now, we wish to *a priori* assume the correlation (tensor-network-state connectivity) structure of Ψ . Typically correlation is local, and hence the typical/canonical choice is to have Ψ 's tensor-network structure to match its parent Hamiltonian/Operator. For example, given an MPO-network, the “matching” network is the Matrix-Product-State (MPS).



The MPS unlike the MPO does not exactly represent its parent tensor. However, if correlation structure is chosen correctly, this hopefully yields a good approximation to its parent tensor. This subtle fact greatly limits the applicability of the MPS. However, we would like to observe: that despite the apparent increase in complexity, the MPS actually has much less entries than the original eigenvector. This is also true for any tensor-network approximate-factorization.

This approximate-factorization controls the sparsity of Ψ , and hence our objective is to diagonalize H (the MPO) given this sparse-constraint. This can easily be achieved in Krylov-subspace-methods if we compute its effective matrix-vector product, *e.g.* the merging of MPO into the MPS to form a new MPS (as shown above). So this clearly involves tensor-partial-trace. However, there is one subtle issue, our result includes internal indices, which once merged start to grow the tensor-network once again. Therefore, after the partial-trace, re-compression³⁰ (SVD) is necessary to return the resulting MPS to the original bond-dimension m . For example, after applying a MPO layer to an MPS, we obtain an MPS of higher bond-dimension of precisely $m' = M + m$, this $\text{MPS}(M + m)$ needs to be compressed, as shown in McCulloch 2007 to $\text{MPS}(m)$. The MPO on MPS application for two generic sites is shown below (note the larger bond-dimension).



After this “matrix-vector” product is estimated, we may obtain the iterative-Krylov-

³⁰This compression may also be obtained using iterative methods!

subspace ED on a generic QMB-state tensor-network decomposition (beyond MPS and MPOs), with sub-exponential scaling.

4.8 Density-Matrix-Renormalization-Group

Now we shall discuss White’s algorithm, White 1992, 1993, and generalizations, with another a slight generalization to Tree-Tensor-Networks (TTN), to find approximate solutions to the aforementioned QMB models efficiently. The first step in this algorithm is to obtain the factorization of your model into a tensor-network. For White, this was the 1-dimensional chain of the original Heisenberg-model, whose factorization is achieved via the Matrix-Product-Operator (MPO) as described above. The key property of this operator is that it has 2-physical-index surfaces, when each surface is reshaped into a single index, the result is the model’s Hamiltonian-matrix with two-indices. The Heisenberg-Hamiltonian is thus exactly factored, and will be represented by H , with each piece $W^{(\ell)}$ (corresponding to site ℓ), such that:

$$H^{\alpha\beta} = W_j^{(0) i \alpha_0 \beta_0} W_k^{(1) j \alpha_1 \beta_1} W_l^{(2) k \alpha_2 \beta_2} \dots W_z^{(\ell) y \alpha_n \beta_n} \quad .$$

with $\alpha = \{\alpha_0, \alpha_1, \dots, \alpha_n\}$ and $\beta = \{\beta_0, \beta_1, \dots, \beta_n\}$. And the ends of the MPO, are usually matched with “vacuum” states: $\langle 0|_i$ and $|0\rangle^z$.

4.8.1 State

Our objective will be a similar-shaped state-tensor-network with a single physical-index-surface. Unlike the Hamiltonian-operator, the state's *factorization* is not in-general exact and corresponds to the correlations/entanglement within the many-body wavefunction, represented symbolically by Ψ .

4.8.2 The Energy

If Ψ is representing the ground-state, then the full trace:

$$\mathcal{E} = \text{tr}(\Psi^\dagger H \Psi)$$

represents the ground-state energy. Our objective is therefore to optimize Ψ , such that \mathcal{E} is as low as possible

$$\frac{\delta \mathcal{E}}{\delta \Psi} = 0 \quad ,$$

given the degrees-of-freedom of Ψ . For this we need to introduce a new kind of calculus, that of tensor-networks. The integral is the aforementioned partial-trace, while the derivative is given next.

4.8.3 Tensor Network Derivative & Optimization

Let's define the Tensor-Network-Derivative (TND) of any tensor-network as:

$$\partial_{(i)} A \quad ,$$

this operation merely deletes the i th tensor, and the resulting external-indices created from this deletion creates a new tensor-network. Higher-order TNDs are possible via deleting more than 1 tensor:

$$\partial_{(ij)} \quad \partial_{(ijk)} \quad \partial_{(ijkl)} \quad \cdots \quad \partial_{(ij\dots z)} \quad .$$

However, of special interest are the 2nd-TND, or 2-site-optimization as discussed by White 1992. For the 2-site-optimization, given sites i and j , this corresponds to solving the eigenvalue-problem for ψ_{ij} (2-body/site wavefunction):

$$(\partial_{(ij)}\mathcal{E}) \psi_{ij} = \mathcal{E}_{ij} \psi_{ij} \quad .$$

In particular, we introduce the density-matrix (assuming quantum-equilibrium):

$$\rho = \Psi \otimes \Psi^\dagger \quad .$$

To obtain (the DMRG equation):

$$\underbrace{(\partial_{(ij)}(\rho H))}_{\text{enviroment}} \rho_{ij} = \mathcal{E}_{ij} \rho_{ij} \quad .$$

After diagonalizing to obtain the 2-body-density-matrix, ρ_{ij} , we use tensor-decomposition-techniques (e.g. SVD), to decompose this 2-body-density-matrix into connected 1-body pieces:

$$\rho_{ij} = \text{Tr}(\rho_i \rho_j) \quad .$$

4.8.4 Update & Sweep

Recall, our objective is to compute Ψ , while holding H fixed. Therefore, we perform a *sweep* over all adjacent site-pairs in our tensor-network. This would result

in a change in Ψ , and hopefully into a convergence to an unchanging state, the ground-state, given the constraints of the system. Excited states may be found in a similar way, by constraining the diagonalization into a particular excited state, *e.g.* via the Krylov-subspace methods explained before.

4.8.5 Generalized-DMRG

DMRG is commonly defined for 1-dimensional systems, as it was originally formulated for by White 1992. However, our definition above can be applied for any tensor-network, however, the issue becomes tracing the environment, $\text{Tr}(\partial_{(ij)}(\rho H))$, for higher dimension/more complicated (especially with loops) networks. This generalization (for dimensions higher than 1) is called Projected-Entangled-Pair-States (PEPS), see Verstraete and Cirac 2004.

4.9 Mean-Field Theory

After our definition of DMRG, we have a rather simple Mean-Field (MF) theory definition: we define Mean-Field Solution to be DMRG($m = 1$). That is we seek a MPS, that is a tensor-product state, *i.e.* the sparsest solution! For a homogeneous 1-dimensional system, with N state per site, with L sites the bond-dimension scales upto N^L , as shown below.

$$\text{Mean-Field, } 1 \xrightarrow{\quad m \quad} N^L, \text{ Exact}$$

Pure MF-theory for electrons is rather unnatural, but formally exists as Hartree Theory³¹, because of their fermionic mutual/pair exchange-interaction. Hence the wide-scale application of Hartree-Fock theory, a MF-theory including the essential fermionic 2-body exchange-interaction. Formally this requires us to consider MPS with at least $m = 2$. Therefore, we may extend this definition, a generalized-MF-theory, to be a MPS with bond dimension $m \sim N$, the degrees-of-freedom per site. Or even more generally, have a bond-dimension m independent on the system-size, *e.g.* $\partial_L m = 0$, a kind of *pseudo-mean-field-theory*.

4.10 Sparsity of Tensor-Network Pieces

4.10.1 Sparsity of H

For an arbitrary network/graph J_{ij} , the pair-spin-operator $S \cdot S$ has sparse-elements³² of $N^{L-2} \left(\frac{(3N+2)(N-1)}{2} \right)$. For $E = L - 1$ (1-dimensional path) number of edges in J_{ij} , and divided by the total number of elements we have:

$$\text{density} = N^{L-2} \left(\frac{(3N+2)(N-1)}{2} \right) \frac{1}{N^{2L}}$$

$$\text{sparsity} \sim N^L \quad .$$

³¹Also Time-Dependent-Hartree (TDH) or Random-Phase-Approximation (RPA), and even Kohn-Sham theory (without HF-exchange).

³²The number of parameters for $\mathfrak{su}(N)$ is $N(N-1) + \frac{N(N+1)}{2} - 1 = \frac{(3N+2)(N-1)}{2}$.

As $L \rightarrow \infty$, the sparsity becomes infinite and hence the matrix nonzero entry density becomes zero.

4.10.2 Sparsity of W

Without 1-body/on-site terms, in D dimensions (the 1-dimensional MPO uses $D = 2$ tensor-operators, 2-dimensional PEPS use $D = 4$), we have:

$$\begin{aligned} \# \text{ nonzero elements} &= D \left(\left(\frac{(3N+2)(N-1)}{2} \right) + N \right) \sim DN^2 \quad , \\ \# \text{ all elements} &= N^2 (N^2 + 1)^D \sim N^{2(D+1)} \quad , \\ \text{sparsity} &\sim \frac{N^{2D}}{D} \quad . \end{aligned}$$

Again sparsity increases exponentially with dimension with internal degrees of freedom N , albeit at a slower rate than the entire Hamiltonian-matrix. For a given-dimension, *e.g.* it decreases polynomially $\sim N^{-4}$ for $D = 2$ (for an MPO).

4.10.3 Sparsity of Ψ

Unlike the H which is undoubtedly very-sparse with a well-defined/obviously-factorizable structure, Ψ is trickier. The fact that it is given by a tensor-network already suggests sparsity. Within the MPS network, a larger system L (the number of sites or individual coherent systems), will require larger bond-dimension m . Suppose each 1-body-system has N states (for the regular Heisenberg model this is $N = 2$), in an MPS, we then may define the sparsity $= \frac{N^L}{m}$. An example of sparsity is the filled

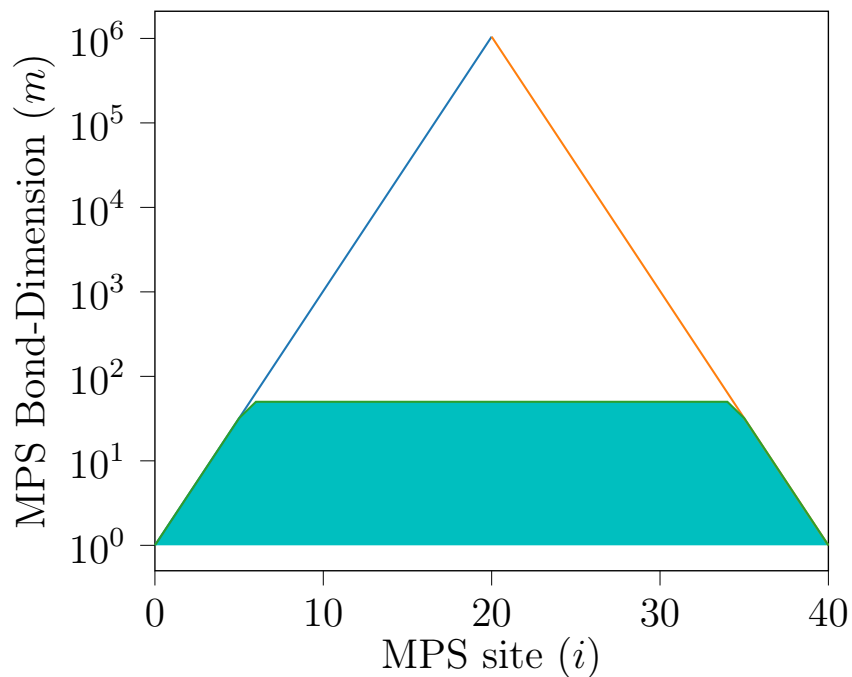


Figure 12. Shown is a Log-plot Showing the Bond-dimension, m , of an MPS. The Filled-in Portion, Shows the MPS is Modelling Compared to the Entire ED-wavefunction-tensor. This is Plotted For a Region of 40 Sites. Note Although the Non-filled in Region Appears Smaller, Its In-fact Exponentially Larger Than the Filled in Portion Due to the Vertical Axis-scale.

versus unfilled region in fig. 12. So to address the question of the sparsity of Ψ based on the spin-chain's system length L , we require $m(L)$, the required³³-bond-dimension given the system's length.

³³To achieve a certain level of convergence or accuracy.

4.11 Sparsity & Entanglement-Entropy

At the end of the last section, we discussed the sparsity of 1D MPS Heisenberg states to be $\frac{N^L}{m}$, with bond-dimension m , system-length L , and with generators belonging to the $\mathfrak{su}(N)$ Lie-algebra. However, this is unsatisfying on its own because N, L are intrinsic parameters of the system (they parameterize the Hamiltonian), while m does not. Therefore, we would like a functional form of m in terms of these parameters, here we achieve this numerically, and fit to an empirically motivated and conjectured function. In calculating our objective, we are motivated by a connection between: entanglement-entropy and sparsity. These are closely related, at least in the context of SVD: while entanglement-entropy is defined for quantum states (on a boundary) via the sum of the log of singular-values. Sparsity, by rank arguments before, is the number of singular-values.

When modelling our QMB-Hamiltonian, we would like our QMB-wavefunction to capture most of the entanglement-entropy set-forth by exactly-solvable analytical methods. The QMB-wavefunction that stays close to this value, despite compression, is said to *emulate* or represent the entire theory *faithfully*³⁴.

As we know the low-energy states of the Heisenberg-model resemble the WZW model; $[\mathfrak{su}(N)_1] \sim \text{WZW}(\mathfrak{su}(N)_1)$, as described in Führinger et al. 2008; Nataf and Mila 2018. Previously calculated WZW results such as the central-charges, c , which determine the entanglement-entropy function (shown later), and scaling-dimensions

³⁴The precise definition of *close* is arbitrarily defined.

Δ may be theoretically/analytically calculated, Fradkin 2013, to be:

$$\begin{aligned} c_{\mathfrak{su}(N)_k} &= \frac{k(N^2 - 1)}{k + N} & \rightarrow & c_{\mathfrak{su}(N)_1} = N - 1 & . \\ \Delta_{\mathfrak{su}(N)_k} &= \frac{N^2 - 1}{N(N + k)} & \rightarrow & \Delta_{\mathfrak{su}(N)_1} = \frac{N - 1}{N} & . \end{aligned}$$

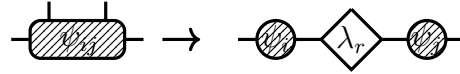
With k representing the integer *level* of the WZW theory, and the spin-representation $k = 2s$ of the $\mathfrak{su}(N)_k$ Lie-algebra in the Heisenberg-model case.

4.12 Entanglement-Entropy

Given a quantum-state, e.g. QMB-state Ψ , Entanglement-Entropy, as developed and shown by **Taddia2013**; Bennett et al. 1996; Calabrese and Cardy 2004; Rangamani and Takayanagi 2017 concerns the factorizability and connectivity of Ψ . This value may be theoretically and numerically calculated, and thus may be used to determine the amount of total connectivity considered by the numerical calculation. Theoretically as $m \rightarrow N^L$ the full-connectivity is obtained, and the central-charges should match: $c_{\text{CFT}} = c_{\text{DMRG}}$. However, even then there is a slight difference between the CFT-theory and the intrinsically discrete MPS, and thus the central-charge c , for larger $\mathfrak{su}(N)$ off the theoretical-value for increasing N , as shown in Nataf and Mila 2018 the precise reason for this effect is still unknown.

4.12.1 Bond-Entropy

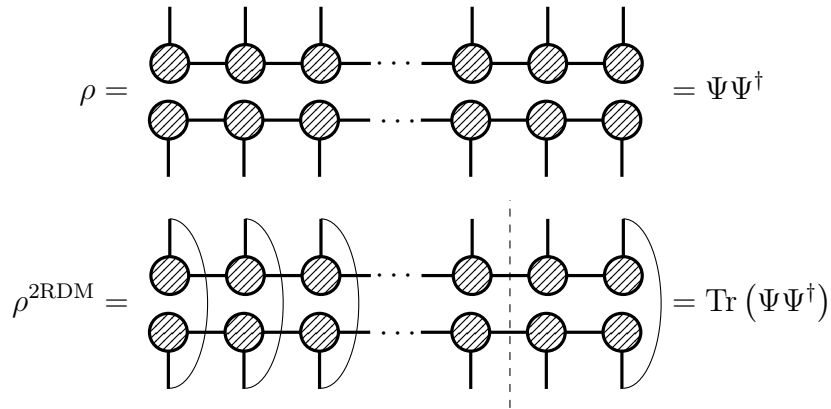
In the DMRG-algorithm, when we decompose the 2-site ψ_{ij} wavefunction, via SVD, into two 1-site wavefunctions ψ_i and ψ_j (with singular-values λ_r), we have an option for a bond-dimension cutoff m . The von-Neumann formula applied to these singular-values yields the Bond-Entropy. Throughout the MPS, this value should be constant (the *Area-law*).



$$S = - \sum_r \lambda_r \log \lambda_r$$

4.12.2 Block-Entropy

To compute the block-entropy, we compute a partial-trace over the overlap, in order to obtain Reduced-Density-Matrices (RDMs) on either side of the partition.



$$\rho = \Psi \Psi^\dagger$$

$$\rho^{2\text{RDM}} = \text{Tr}(\Psi \Psi^\dagger)$$

Therefore the singular-values of the $\rho^{2\text{RDM}}$ given the cut ℓ , yield the block-entropy. These can be obtained if the MPS can be canonicalized with respect to one of the

boundary sites, and traced and decomposed with the other boundary site. A similar algorithm may be used to compute Renyi-Entropies, as seen in Rolandi and Wilming 2020.

4.12.3 Central-Charge

The relationship between the central-charge c , along ℓ (the partition) a finite L -site spin-chain (with site spacing a , usually in units of $a = 1$, and Open-Boundary-Conditions (OBC)), and the Entanglement-entropy S is given by the Cardy-Calabrese formula, derived in Calabrese and Cardy 2004, (eq. 4.8):

$$S(\ell; c, C_1) = \frac{c}{6} \log_2 \left(\frac{2L}{a\pi} \sin \left(\frac{\pi\ell}{L} \right) \right) + C_1 \quad . \quad (4.8)$$

When used with DMRG for an Ψ_{MPS} , we may compute a numerical c_{DMRG} , along (ℓ) parts of the tensor-network (*i.e.* the MPS). This then may be compared to CFT results listed above. Above C_1 is a constant with contributions with the boundary entropy of Affleck-Ludwig, see Affleck and Ludwig 1991; Laflorencie et al. 2006, along with other contributions which are lesser understood.

4.13 Procedure

Our objective is to compute the required bond-dimension m of a converged Ψ_{MPS} on L sites. Therefore we scan over the L dimension (simulate many systems with

different number of L sites), and determine the required m , based on the convergence³⁵ c_{DMRG} . Because finite-sized systems seem to not match, as shown in Nataf and Mila 2018, the exact theoretical c_{CFT} , we opt for convergence with-in the c gradient with-respect-to m (for a given system of size L) for some tolerance (arbitrarily chosen as 0.001); $\frac{dc}{dm} < \delta c = 0.001$. Popular tensor-network software, Roberts et al. 2019; Gray 2018, are used in conjunction to the aforementioned algorithms, for computational efficiency. In summary the following simulation algorithm is followed (for each L -system size):

1. We solve L -site $[\mathfrak{su}(2)_1]$ DMRG upto m MPS-bond-dimension Ψ_{MPS}
2. Calculate the block Entanglement-Entropy, $S(\ell) = -\rho_\ell \log_2(\rho_\ell)$
3. least-squares-fit to the Cardy-Calabrese Formula (eq. 4.8) to find c (and C_1).
4. if $c - c' \leq 0.001$ the simulation is converged (take the m value which achieves this convergence)

4.14 Results

Let's first compute the block-entanglement entropy over a sample system³⁶ of 120 sites for $[\mathfrak{su}(2)_1]$ and $[\mathfrak{su}(3)_1]$, these results are plotted in fig. 14. Notice these plots follow the Cardy-Calabrese formula, they contain ripples/oscillations about this value

³⁵Or alternatively the ground-state-energy \mathcal{E}

³⁶The model used is $H_{\alpha\beta} = \frac{1}{N} J_{ij} S_{\alpha\gamma}^{ia} S_{\gamma\beta}^{ja}$, *ie.* for only the fundamental-representation.

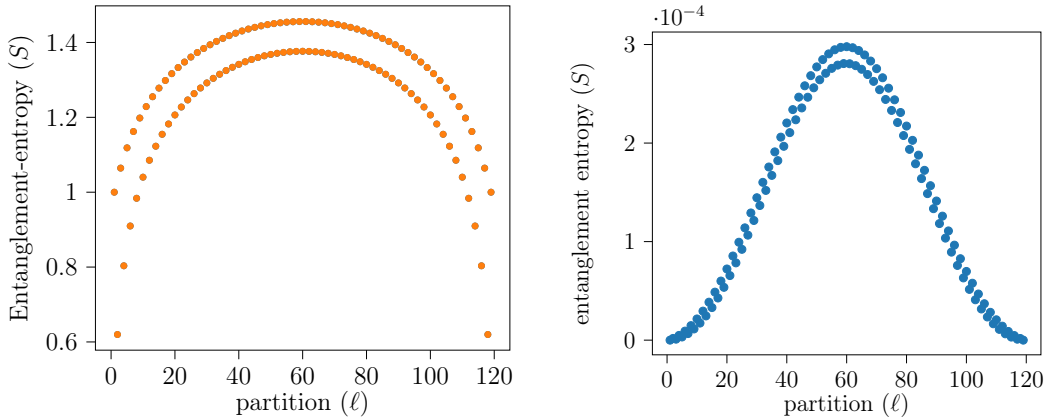


Figure 13. Examples of the Entanglement-Entropy of the $\mathfrak{su}(2)_1$ Model (Left Plot) for $m = 60$ (Lower Curve) and $m = 200$ (Upper Curve) Respectively, and Their Difference (Right Plot).

in the entanglement-entropy. The reason for these ripples, which exist only for the OBC formulation is not precisely known, but can be fitted as is done in Laflorencie et al. 2006. The correct fit is necessary to determine the proper entanglement-entropy and thus central-charge c . This is not our main focus in order to compute the sparsity, and thus the central-charge obtained from the Cardy-Celebrese formula is sufficient for our purposes.

4.14.1 $c(m)$ Calculation

Next, lets determine the central-charge (c , fig. 15) and entropy-constant (C_1 , fig. 16) with different MPS-bond-dimensions, given the same system $L = 60$ for $\mathfrak{su}(2)_1$, $\mathfrak{su}(3)_1$, and $\mathfrak{su}(4)_1$.

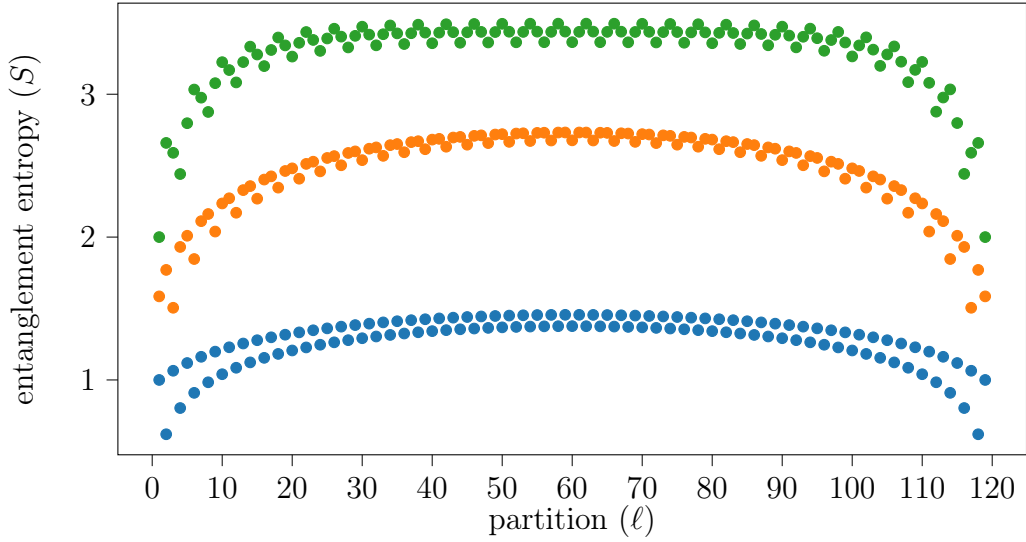


Figure 14. Examples of the Entanglement-entropy as a Function of the Bipartite Divider ℓ on a 1-dimensional $L = 120$ Site Lattice Spin-chain. The Upper Curve Shows the Results for $[\mathfrak{su}(4)_1]$, the Middle Curve Shows Results for $[\mathfrak{su}(3)_1]$, and the Lower Curve Shows the Result For $[\mathfrak{su}(2)_1]$.

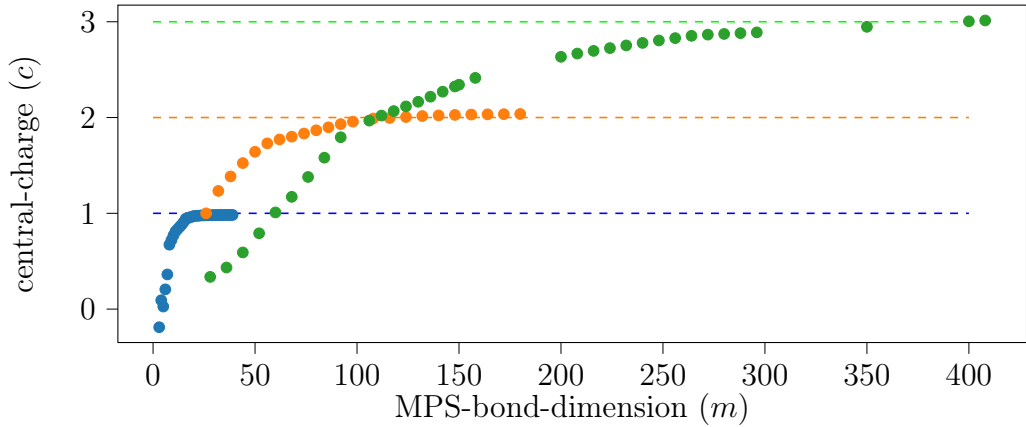


Figure 15. This Plot Shows the Central-charge as Fitted to DMRG Data, the MPS Over $L = 60$ Spin-sites. For $\mathfrak{su}(2)_1$ (Blue) Which Converges to $c = 1$ as Expected With Bond-dimension $m \approx 20$. $\mathfrak{su}(3)_1$ (Orange) Approaching $c = 2$, Also as Expected, with Bond-dimension Roughly $m \approx 100$. And $\mathfrak{su}(4)_1$ (Green), With the Central-charge Increasing Not Converging to the Expected $c = 3$ Value Yet.

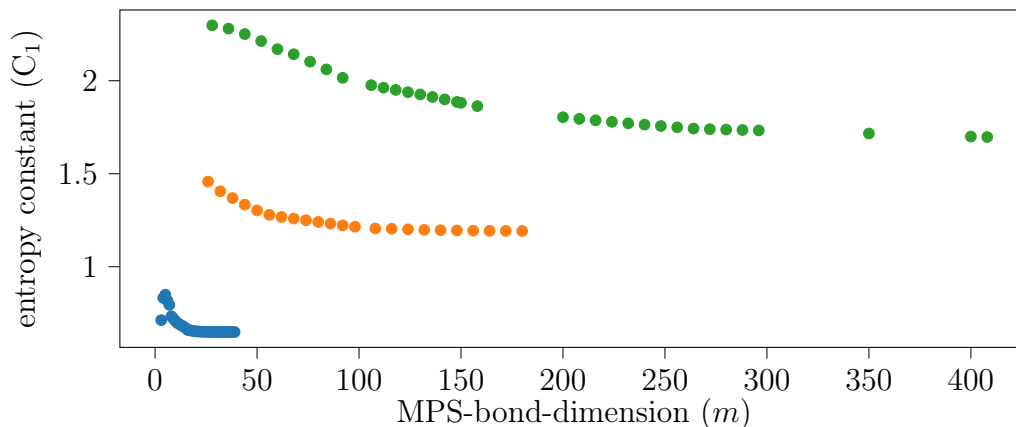


Figure 16. This Plot Shows the Constant-entropy Term, C_1 , in Eq. 4.8 Least-squares-fitted to Numerical DMRG Data (the MPS). For Systems With $L = 60$ Spin-site Chains in $\mathfrak{su}(2)_1$, $\mathfrak{su}(3)_1$, and $\mathfrak{su}(4)_1$ Theories.

4.14.2 $m(L)$ Calculation

Now we give an example in $SU(2)$ following the procedure above. We find the following result after running 120 DMRG calculations to compute the central-charge c :

$$\text{sparsity}(\Psi_{\mathfrak{su}(2)}^{\text{MPS}}) \sim \frac{2^L}{L^{1/2}} \quad .$$

The data is plotted on fig. 17, and every data-point has an uncertainty of $\delta m \pm 2$ (note m and δm are integers). Furthermore, the energy $\mathcal{E}(L)$ along the same parameters are plotted in fig. 18.

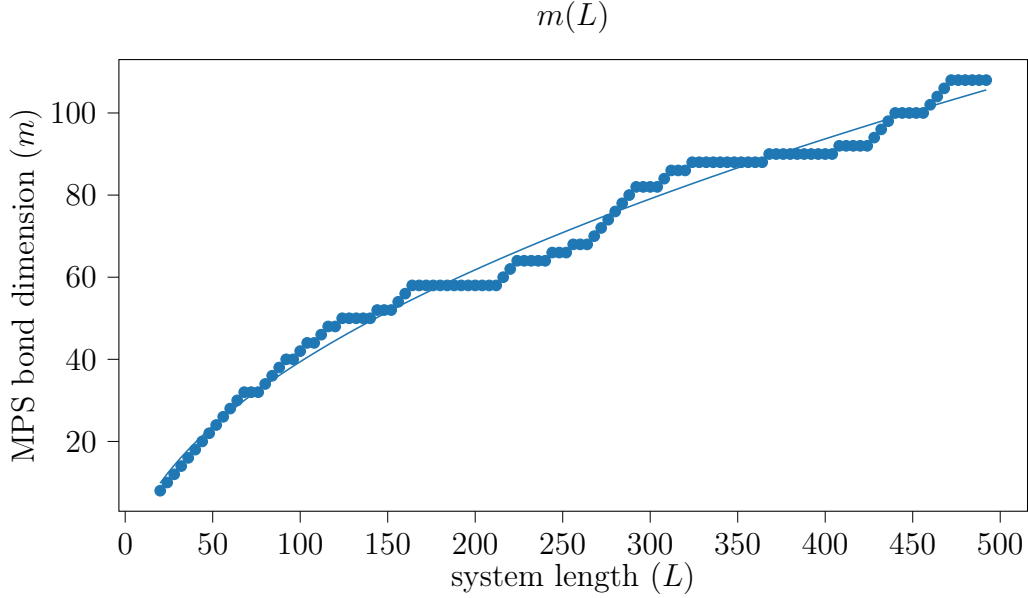


Figure 17. The “Converged” (Such That the Central-charge Does Not Change *Much*) Bond-dimension m (MPS in DMRG) For A XXX Spin-chain of Length L . DMRG Data is Fitted With Function $m = b * L^a + c$, With Least-squares Parameters: $a = 0.51001911$, $b = 5.0420591$, $c = -13.39164775$.

4.14.3 Conjecture

Note that $m \sim 5L^{1/2}$, so the total computational-complexity is $\mathcal{O}_{\text{DMRG}} \sim m^3 \sim L^{3/2}$, based on system’s size/length. Now we conjecture that the the sparsity of $[\mathfrak{su}(N)_1]$ Heisenberg is:

$$\text{sparsity}(\Psi_{[\mathfrak{su}(N)_1]}^{\text{MPS}}) \sim \frac{N^L}{L^{1/N}} \quad .$$

As it would follow that $N \rightarrow \infty$, the MPS-bond-dimension $m \rightarrow C$, *i.e.* m should approach some constant value C , reaffirming the Mean-Field-theoretic, large- N , concept. A major limitation to verifying this claim is the bond-dimension m quickly

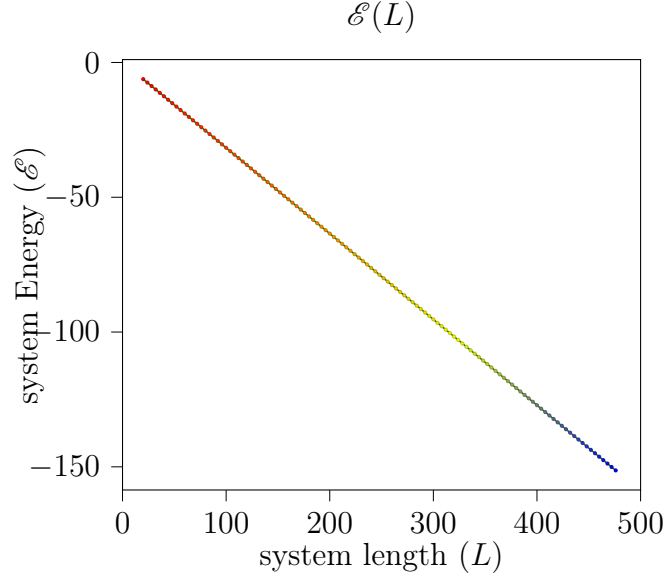


Figure 18. The Converged Energy \mathcal{E} (MPS in DMRG) For A XXX Spin-chain of Length L . DMRG Data is Fitted With Function $m = a * L + b$, With Least-squares Parameters: $a = -0.3183050050730525$, $b = 0.17962704375639882$.

becomes $m \approx 2000$ for moderate system sizes ($L = 120$ as shown in Nataf and Mila 2018) upto $\mathfrak{su}(8)$. As mentioned in earlier sections, the sparsity of the MPO is not much help here, instead we must constrain the symmetries of the MPS, this yields the blocksparse structure of Ψ_{MPS} .

4.14.4 Symmetry imposed Ψ_{MPS}

Taking into account symmetries induces a blocksparse structure on the Ψ_{MPS} , this is very convenient because DMRG-algorithm is largely limited by the MPS bond-dimension m , by $\sim m^3$. The block-sparse structure induced by symmetries, allow for linear-matrix-algebra on the smaller blocks. Symmetries alone do not reduce the $\sim m^3$, cubed, time-complexity, but do allow for larger systems to be considered (because of the suppression of $m \rightarrow m/f$, for some $f > 1$).

$$\begin{array}{c}
 N \\
 | \\
 m \text{ --- } \textcircled{\text{---}} \text{ --- } m
 \end{array}
 = N \left\{ \left(\begin{array}{c} m \\ m \\ \vdots \\ m \end{array} \right) \left\{ \left(\begin{array}{c} \\ \\ \vdots \\ \end{array} \right) \right\} \right\}$$

Figure 19. When Constraining For Symmetry, Within Each m^2 -sub-matrix May Be Reduced to Block-diagonal Form.

We may break the $SU(N)$ symmetry with-respect-to the $N - 1$ Cartan-generators, hence $SU(N) \rightarrow U(1)^{N-1}$. However, the best approach is using the full $SU(N)$ symmetry (all Lie-algebra generators), as done in Nataf and Mila 2014, this has been shown in earlier work. The importance and first implementation of this idea was motivated by earlier work by McCulloch and Gulácsi 2001.

4.15 Error Fidelity

The standard algorithm to compute the similarity between two quantum-states, or there *fidelity* (faithfulness to each other), is usually given by:

$$F(\sigma, \rho) = \left| \text{tr} \left(\sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right) \right|^2,$$

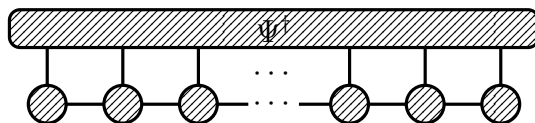
This applies for general mixed density matrices σ & ρ , Nielsen and Chuang 2010. If both states are pure-states we instead have the overlap:

$$F = \left| \text{tr} \left(\Psi_{\sigma}^{\dagger} \Psi_{\rho} \right) \right|^2 .$$

This measure, quantum-state-fidelity F , has two important attributes: $F(\sigma, \rho) = F(\rho, \sigma)$ and $F \in [0, 1]$, with $F(\sigma, \rho) = 1$ iff (if and only if) $\sigma = \rho$.

In the recent literature, interesting work in determining if two different quantum-states, from two different systems, have length-scale equivalence, IR-equivalence, using the same measure, Liu et al. 2017; Zou, Milsted, and Vidal 2018; Hauru and Vidal 2018.

In this section we compare the fidelity of the MPS when compared to the ED-State (EDS). This is done by computing the square of the overlap between the two quantum-states: $F = \left| \text{tr} \left(\Psi_{\text{EDS}}^{\dagger} \Psi_{\text{MPS}(m)} \right) \right|^2$, note $\text{tr} \left(\Psi_{\text{EDS}}^{\dagger} \Psi_{\text{EDS}} \right) = 1 = \text{tr} \left(\Psi_{\text{MPS}(m)}^{\dagger} \Psi_{\text{MPS}(m)} \right)$.



The results are shown in figs 20 and 21. It is noteworthy that with $m \approx 20$, the MPS essentially identical to the EDS state for all $L = 10, 14, 22$ lattice sites systems for $[\mathfrak{su}(2)_1]$. The MPS sparsity is calculated by:

$$\text{sparsity}(\Psi_{\text{MPS}}) = \frac{2^L}{\#\text{elements in MPS}} .$$

Notice that fractional sparsity, implies the MPS has more parameters than the EDS, in the aforementioned figures.

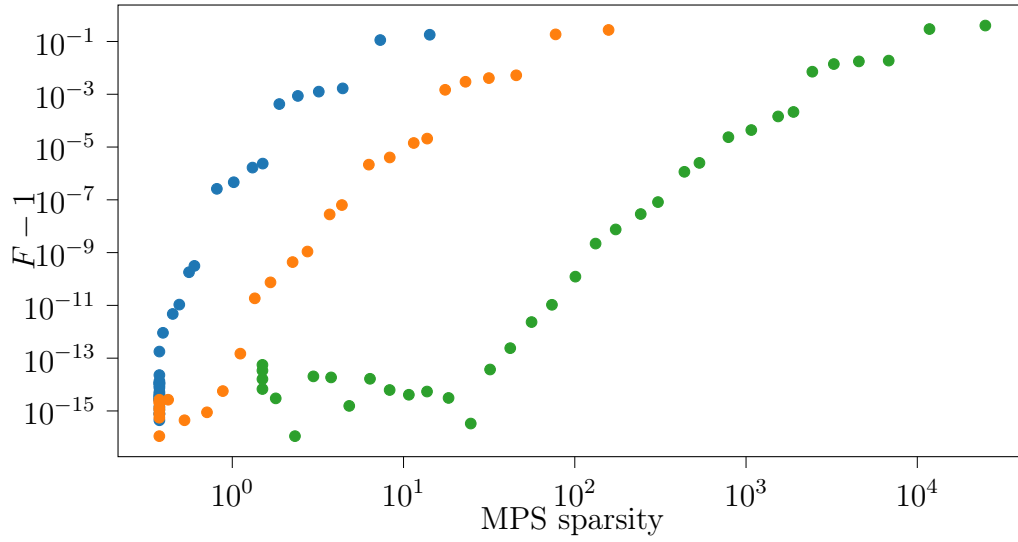


Figure 20. Shown is the Fidelity of the MPS State (for Sparsities) when Compared to the EDS, for $L = 10, 14, 22$ Lattice Sites, with Blue, Orange, and Green Dots Respectively.

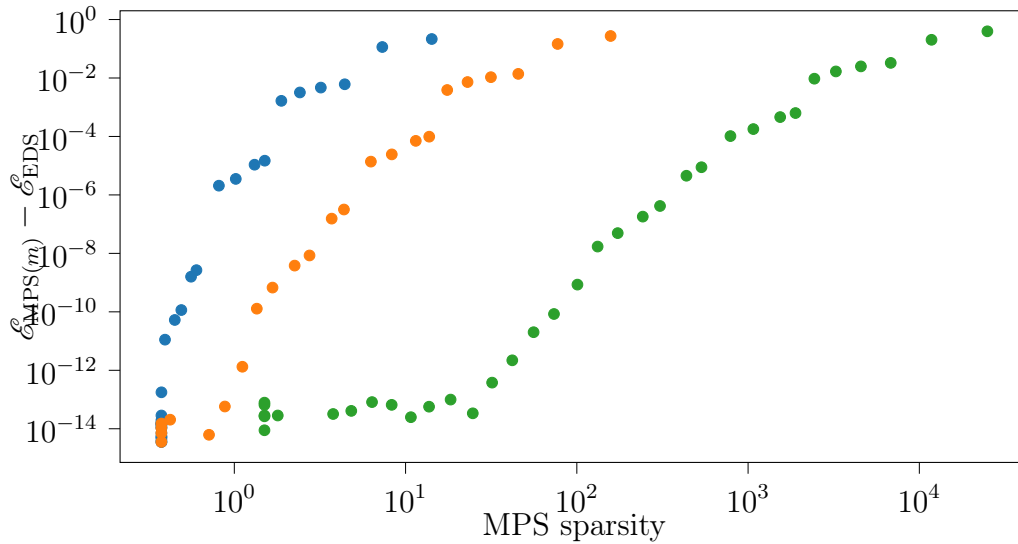


Figure 21. Shown is the energy difference of the EDS and MPS (for Various Sparsities) states, for $L = 10, 14, 22$ Lattice Sites, with Blue, Orange, and Green Dots Respectively.

Chapter 5

SPARSITY IN CHEMICAL SYSTEMS

Since the QMB Schrödinger-equation has been established as the *defacto* equation of microscopic scale chemical theory, quantum-chemistry has been *effectively solved*. Within the Born-Oppenheimer approximation, this QMB is represented by the Full-Configuration-Interaction Hamiltonian (FCI), whose Exact-Diagonalization (ED) yields the desired wavefunctions. The remaining problem is the actual wavefunction solution. Resulting problem is exponentially-complex, and directly infeasible to actually be solved on classical-computers. As we have explained before wavefunctions exist, within our selected basis. Therefore the lowest level structure for us are the basis-functions³⁷, when orthogonalized these are called *orbitals*, where wavefunctions may be constructed on to form *configurations* or Slater-determinants. The selection of many configurations are then called an *active-space*. Interestingly, sparsity actually plays a role in all three levels of theory, and this will be shown here. This chapter is based off, and serves as an introduction to Candanedo 2023a. This work allows for the construction of an arbitrary Selected-Configuration-Interaction (SCI) systems as an alternative to the exponentially-costly FCI problem.

³⁷Historically and currently the Gaussian-Basis-Functions (GBF) serve as an important representation of these basis-functions.

5.1 Orbital Interaction

The orbital-space is determined using atomic-centric orbitals, while their orbital configuration solution is optimized through a Mean-Field (MF) method known as Self-Consistent Field (SCF). This yields the orthogonal-basis Molecular-Orbitals (MO): SCF-coefficients (p index represents MO basis, while α represents atomic-orbital basis):

$$\psi_p = C_p^\alpha \phi_\alpha(r) \quad .$$

To obtain the SCF-coefficients, C_p^α , we need to diagonalize the 1-body Fock-operator, constructed by:

$$F_{\alpha\beta} = T_{\alpha\beta} + V_{\alpha\beta} + (\alpha\beta|\gamma\delta)C_p^\gamma C_q^\delta O_{pq} \quad .$$

With: $T_{\alpha\beta}$ the kinetic-energy-integral, $V_{\alpha\beta}$ 1-body potential-integral (*e.g.* nuclear-electronic interaction, dipole-moment, etc...), $(\alpha\beta|\gamma\delta)$ the 2-body-integral (*e.g.* Electronic-Repulsion-Integral (ERI)), and O_{pq} Hartree-Fock ground state Slater-determinant occupation. This algorithm has scales at $\sim N^4$, due to the 2-body term, fortunately this term is often sparse (especially for sufficiently large systems). This sparsity is due to two kinds: via selection rules (due to the Wigner-Eckart Theorem) and approximately (due to sufficiently small overlaps $\sim \epsilon_{\text{machine}}$). Once this sparsity is exploited the algorithm is $\sim N^3$ due to the full-diagonalization of $F_{\alpha\beta}$.

5.2 Full Configuration-Interaction

In the previous section, we have only optimized a special determinant: the ground-state determinant. In reality there are many possible, in-fact exponentially many of these determinants/QMB-states. The QMB-electronic-states, Ψ , are defined as (with CI-coefficient C_I^p):

$$\Psi_I = C_I^p \psi_p \quad .$$

Although, the FCI-Hamiltonian is incredibly sparse, the number of possible states is enormous:

$$\# \text{ FCI states}(N, n^\uparrow + n^\downarrow) = \binom{N}{n^\uparrow} \binom{N}{n^\downarrow} \quad ,$$

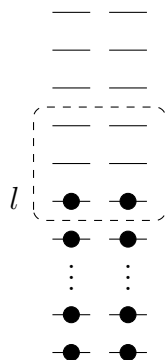
with N being the orthogonal-orbital count, n^\uparrow and n^\downarrow being the number of up and down electrons considered. Therefore we desire approximate solutions to this formulation.

5.3 Active-Space Configuration-Interaction

Interestingly not all determinants are required to capture the physics of interest in most problems, e.g. bond-breaking. Therefore, it has become increasingly clear that *sparse-methods*, as I like to call it, come to the rescue. Often, only a very small subset of states are selected, this may be from a truncated-CI (CIS, CISD, etc...) or active-space CI. The Complete-Active-Space (CAS) notation will be defined as such:

$$\text{CAS}(\text{orbitals}, \text{electrons}) = \text{CAS}(N, n^\uparrow + n^\downarrow) \quad ,$$

this is standard notation in chemical-physics. We give an example for CAS(3, 2):



Note this definition, CAS, is irrespective of the choice of active-orbitals, instead we may define a vector $\xi_i = (l, l + 1, l + 2)$, with l representing the location of the 1st active orbital in the figure above, to select the active-orbitals, to avoid ambiguity.

For CAS($M, m^\uparrow + m^\downarrow$), we have the following:

$$\text{sparsity} = \frac{\binom{N}{n^\uparrow} \binom{N}{n^\downarrow}}{\binom{M}{m^\uparrow} \binom{M}{m^\downarrow}} .$$

With each new QMB-state/determinant, we should re-optimize the SCF-orbital-coefficients C_p^α , this is also done, in addition to configuration-optimization, we obtain Multi-Configuration-Self-Consistent-Field (MCSCF).

5.3.1 Multi-Configuration-SCF (MCSCF)

The FCI problem may be solved for CAS(18,18), and CAS(22,22) on supercomputers. These active-spaces are often not sufficient for many interesting chemical systems. Instead approximate FCI-solvers allow for much larger active-spaces:

- DMRG, Ma, Schollwöck, and Shuai 2022b, (CAS(108,30) with $m = 6000$ and 2800 CPUs as shown in Zhai and Chan 2021)
- FCIQMC, Booth, Thom, and Alavi 2009, (CAS(24,24) as shown in Halson, Anderson, and Booth 2020)
- Selected-CI (SCI)
 - Adaptive-Sampling-CI (ASCI, CAS(50,50)), shown in Tubman et al. 2016; Levine et al. 2020
 - Heat-bath-CI (HCI, CAS(190, 12)), shown in Holmes, Tubman, and Umrigar 2016
 - incremental-FCI (iFCI, CAS(118,32)), developed by Zimmerman 2017

My preprint, follows the SCI path and allows for this theory to be computed for arbitrary Slater-determinants without a mechanism of choosing these determinants.

Chapter 6

MOLECULAR DYNAMICS

In this chapter, we explore real-time and real-space molecular dynamics with classical nuclei-centric dynamics. These algorithms are valuable for understanding condensed matter dynamics in both inorganic (Thompson et al. 2022) and organic chemistry (Van Der Spoel et al. 2005; Phillips et al. 2005). We describe these algorithms by first examining atomic dynamics. Then we develop an ab initio understanding of these dynamics. Next, we delve into the more general setting involving strong-correlation regimes and the potential energy surface. We then explore the algorithm’s sparsification, which includes neighbor lists and nearest-neighbor interactions. In conclusion, we bring these concepts together to create an alternative molecular dynamics algorithm aimed at classically modeling excited molecular states. Similar algorithms to molecular dynamics exist in other domains, such as nuclear physics and galaxy/universe dynamics. These large-scale simulations bridge the gap between atomic theory and fluid theory.

6.1 Freshman’s Dream

The prototypical example of molecular-dynamics is the simulation of rare-gases, as developed by Verlet 1967. This is because their inert nature, their interactions may

be reduced by a single function, called the Lennard-Jones function. Unfortunately, these simulations, although as a nice education tool or as a starting-point for a larger project, is otherwise considered non-interesting. However, as mentioned, let's use this as a starting point. The Lennard-Jones function is a analytic function given by³⁸:

$$V_{ij}(r) = \mathcal{E} \left(\left(\frac{r_0}{r} \right)^{12} - 2 \left(\frac{r_0}{r} \right)^6 \right) .$$

This function is empirically-fitted, however for theorists this empirically determined parameters are unsatisfying, and lacks the flexibly and subtle features we may want to capture. Therefore instead we may use a tabulated-potential, which is determined via *ab initio* methods (i.e. quantum-chemistry calculations). The tabulated-potential is a vector, i.e. $V(r) \rightarrow V_r$ over some discrete values of r . Intermediate values, between discrete values, may be obtained via interpolation (e.g. linear or spline). The interpolated-tabulated potential is also useful for capturing multidimensional potential functions, via multidimensional-interpolation methods.

We would like to generalize the atomic description to incorporate more complicated situations. For this we need to explore the inter-molecular forces, there are plenty of excellent books on the topic, including: Dill and Bromberg 2010; Israelachvili 2011; Stone 2013, however our perspectives are *ab initio*.

³⁸with r_0 being the minimum interaction distance (where $V = 0$), and \mathcal{E} being the max cohesion energy, r being the interatomic distance

6.2 Atomic Dynamics

6.2.1 Weak Molecular Interactions

Suppose we have two *atoms* (or molecules/particles), with isolated Hamiltonians H_1 and H_2 with eigenstates ψ_1 and ψ_2 respectively (*e.g.* Lennard-Jones particles). The combined isolated systems could have Hamiltonian $H = H_1 \oplus H_2$. The essential idea of weak/dispersion/induction-interactions is to have their eigenstates interact to yield modified eigenstates $\tilde{\psi}_1$ and $\tilde{\psi}_2$ that describe the (weakly)-interacting system³⁹. This is achieved via perturbation theory on H , as we tactically assume that their cohesion interaction is much weaker than each system internally. As long as these two systems do not appreciably exchange electrons⁴⁰, the systems can be said to be isolated/local.

To first-order, we have Coulomb interactions of any net monopole (electric) charges. The following orders successively include higher and higher any permanent multipole moments. However, the fun begins when we consider non-permanent multipole-moments, and these contributions first appear to second-order perturbation-theory (London's theory). Because, we assumed each atom to be a composite system, with internal structure, this structure itself may interact with the other system's

³⁹For example have a potential-energy as a function of the isolated eigenstates, $V(\psi_1, \psi_2)$, *e.g.* $\propto \psi_1 \otimes \psi_2$.

⁴⁰Note electrons will always be coherently-exchanged, due to quantum-tunneling, but this effect is exponentially suppressed (with $\exp(-\sigma x)$ with σ being the quantum conductivity). This is in-fact is related to the cohesive energies of the systems of interest.

structure. More concretely, their fluctuations interact with each other, and eventually in equilibrium, they synchronously fluctuate leading to a decrease in their combined correlation energy (and a slight alteration of their eigenstates).

Concretely, these contributions may be calculated via the aforementioned method, use the eigenstates to generate an interaction which mediates this weak-correlation contribution. This interaction is mediated by Linear-Response (LR) theory between the atoms (systems), this yields a system centric character: the polarizability or Green's-function (internal)⁴¹.. There exists polarizabilities for all multipole-moments, but to lowest order the most well-known is the dipole-polarizability, usually denoted by α (x, y denote Cartesian indices):

$$\alpha^{xy} \propto \langle \psi^\dagger d^x d^y \psi \rangle \quad .$$

Their frequency-dependent generalization allows for direct computation of the correlation energy:

$$\mathcal{E} = \frac{6}{\pi} \int_{\mathbb{R}} \alpha(i\omega) \alpha(i\omega) d\omega \quad .$$

This may be converted into the famous London formula, which intern can be converted into Lennard-Jones parameters, Tkatchenko et al. 2012.

However, unlike permanent multipole moments, inducible dispersion or weak-interactions (from non-permanent moments) are infamously non-additive. That is there exists intrinsic 3-atom/body contributions which may not be divided into pair-interactions. This also holds for a generic n-atom/body interaction, however,

⁴¹Potentially anisotropic (and multi-polar) and frequency-dependent, especially for interactions with close proximity.

for these interactions they are suppressed by a factor r^{-3n} (for the n -body dipole, leading term). Ultimately the inclusion of many other bodies incoherently, leads to the inclusion of a dielectric function, Lifshitz 1956; Dzyaloshinskii, Lifshitz, and Pitaevskii 1961.

A few notes on these weak/dispersion-interactions are in order. Although, these are “weak” interactions, they are ‘strong’ on the scale of thermal temperatures, and hence play an important role in soft-matter-physics, e.g. biophysics. There is a notable mathematical artifact of the serendipitous cancellation of many-body dispersion-forces with their higher-multipole contributions. Yielding a superficially accurate pair-wise dipole-dipole contributions to the complete dispersion-force.

6.2.2 Strong Molecular Interaction

When these two systems come into direct contact they intrinsically alter each other to form one merged system instead. And thus it would be inappropriate to describe this new system by the previously separate, but interacting pieces. Therefore, we must brute-force diagonalize the new system’s Hamiltonian’s altogether. This is known as the strong-correlation regime. This resulting system, then may then be weakly interacted with another system as described before.

This intuition follows from topological arguments, analogous to a cobordism between two manifolds. There exists a cobordism between two molecular PES joined together via these topological artifacts, known as *conical-intersections* (need not be point-like conical, but also higher dimensional seems, Martinez 2010). These

conical-intersections fuse two potential-energy-surfaces along a degeneracy-seam. And are not only ubiquitous in polyatomic-molecules, but mediate radiationless-transitions (radiation-matter coupling) and chemical-bond formation.

A characteristic feature of many chemicals is the tendency for molecules to settle in non-perfectly-symmetric configuration, even at 0 K. Instead a symmetry-breaking phenomena is at play is called the *Jahn-Teller Effect*, this effect is usually mediated by a conical-intersection, which usually occupies the symmetric point (usually yielding a potential-energy degeneracy), leaving the molecule with a choice of nearby minima.

Therefore, if we wish to model a change in chemistry (chemical reactions, or chemistry), strong-interaction/multi-reference methods are a necessity.

6.2.3 Computational Methods

The holy-grail is the exact-diagonalization of the whole chemical-Hamiltonian. Of course we have discussed this is essentially infeasible. Therefore, many approximate solutions exist. In nature, most molecules are near their equilibrium configuration, in local energy minima. In these portions of the PES, so-called *single-reference* methods do an excellent job capturing this weak-correlation, e.g. Hartree-Fock (exchange only), Coupled-Cluster, truncated-excitation-CI (e.g. CISD), and Kohn-Sham Theories. For situations in the strong-regime, *multi-reference* methods are required. Their are multi-reference generalizations of the aforementioned methods, however the gold-standard in this regime, in quantum-chemistry, is the Complete-Active-Space-2nd-Order-Perturbation-Theory (CASPT2), as developed by Roos et

al. 1982; Anderson, Malmqvist, and Roos 1990. Inside the active space, tensor-network, Gray and Chan 2022, or Density-Matrix-Renormalization-Group (DMRG), Ma, Schollwöck, and Shuai 2022a, algorithms can be used for efficient diagonalization. These multi-reference methods are still an active area of research, and are notoriously known for *a priori* chemical-knowledge for best results.

6.3 Potential Energy Surface

The Potential-Energy-Surface (PES) is a scalar function on the nuclei' configuration space, Teufel 2003. The PES is strictly different for all possible combinations of nucleons and electron count. However, the PES may have many layers, one for each electronic excited state, e.g. fig. 22, and therefore it may also be called a *spectra*. The PES is largely continuous, with possible *cusp*/non-differentiable points of a conical-intersection's seam. A generic PES, is not represented by some analytic function for all points. However, one can coarse grain the PES, by sampling it, *e.g.* by a equally spaced lattice, only to implement interpolation schemes to get the points in-between.

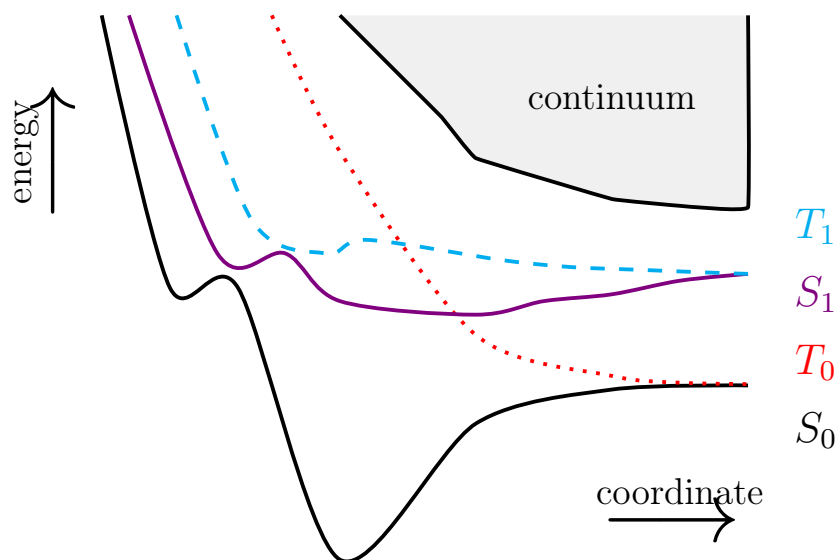


Figure 22. Above is a Cartoon Depiction of a Hypothetical Potential Energy Surface (PES), the Vertical-axis Being Energy, While the Horizontal-axis Being a Reaction Coordinate (Defined by Nuclei Distances). In this Figure, Four Curves are Shown: Ground-State Singlet (Solid Curve, S_0), its Associated Triplet State (Dotted Curve, T_0), An Excited Singlet State (S_1), and another triplet state (dashed curve, T_1). The Gray Shading Region, Bounded By the Uppermost Curve is the Ionization/Continuum Threshold.

6.3.1 Black-Box Decomposition of the Potential Energy Surface

The interatomic forces in real-systems are quite complicated to understand from an *ab initio* level. Therefore, beyond rare-gas-solids, the concept of Force-Fields (FF) arose to describe a “black-box” algorithm calibrated to empirical-data (or *en masse ab initio* Kohn-Sham data). This was done in roughly 3 generations: traditionally-fitted potentials (AMBER, J. Wang et al. 2004, CHARMM, Vanommeslaeghe et al. 2010), bond-order/reactive potentials (Tersoff, Tersoff 1988, ReaxFF, Senftle et al. 2016), machine-learning-potentials (Behler-Parrinello, Behler and Parrinello 2007). This kind

of has the unsatisfying “black-box” feature, which does not make it easily amendable (e.g. excited states) nor can their accuracy be systematically improved.

6.3.2 Atomic Cluster Expansion & Simplicial-Decomposition

Interestingly, the larger intrinsic n -body dispersion/weak interactions contribute a smaller and smaller amount of the cohesive energy. However, this is the opposite for strong interactions which contribute more, i.e. are excited by higher-energy photons. Therefore, we conclude that many-body cutoff is 2 more than the valency of the neutral-nuclei involved (e.g. carbon’s valency is 4, so consider 6-body interactions).

Suppose we have 1 million carbon atoms, with 6 electrons each, in a poor orbital basis (2 orbitals per electron), then our Chemical-Hamiltonian will have: $\approx 2.04 \times 10^{3612356}$ terms. Again this problem is intractable. Suppose only consider weak-interactions, then we may use the atomic approximation, and if we include all N atoms, this system would like an intrinsic N -fold interaction, along with all the other many-body interactions yielding 2^N terms. Hence the classical equations when taking into account all intrinsic many-body interactions also exponentiates, similar to the quantum-systems.

For MD simulations, we consider a potential-energy with pair-wise and greater interactions (that is we relegate this to the chemical-potential and the kinetic-energy). As total energy is conserved, the flow of kinetic energy into potential energy yields the dynamics we desire. We may expand and partition the system’s total energy in

terms of many-body contributions:

$$\mathcal{E} = \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 + \mathcal{E}_4 + \cdots + \mathcal{E}_n \quad (6.1)$$

$$= \frac{1}{1!} V_i(R_i^x) + \frac{1}{2!} V_{ij}(R_{ij}^x) + \frac{1}{3!} V_{ijk}(R_{ijk}^x) + \cdots + \frac{1}{n!} V_{ijk\dots z}(R_{ijk\dots z}^x) \quad (6.2)$$

Above $R_{ijk\dots z}^x$ are displacement-*vectors* (or tensors), in the multi-dimensional limit they represent simplices. Therefore the n -dimensional interaction is tabulated by a n -dimensional dense-array. And the n -body connectivity-structure is given by a n -dimensional sparse-array. My original idea, not implemented, was to tabulate the interactions of all possible atoms truncated at a given n -body level. For instance for hydrogen-peroxide (H_2O_2), we would calculate the following interactions in vacuum: $V_{\text{H}}, V_{\text{O}}, V_{\text{HH}}, V_{\text{HO}}, V_{\text{OO}}, V_{\text{HHO}}, V_{\text{HOO}}, V_{\text{HHOO}}$ (order does not matter). The issue with this native method, is the immediate atomic environment are extremely important. Atoms are polarizable in the weak-limit, and change behavior dramatically in the strong-limit with covalent-bonds.

Interestingly, parallel to my early work, Drautz 2019; Bachmayr et al. 2019 developed a new formalism, as of 2019, called the Atomic-Cluster-Expansion (ACE). His work was a far reaching and more developed form of my idea, originally building on the work of tensor-potentials, developed by Shapeev 2016, a few years prior to Drautz's work. Instead Drautz/ACE considers pseudo-hydrogenic (1-body) basis functions for each atom i and its immediate neighborhood atoms j :

$$A_{n\ell\mu}^i(R_{ijx}) = \sum_j \overbrace{R_{B_i B_j}^{n\ell}(R_{ij})}^{\text{radial}} \overbrace{Y_\ell(R_{ijx})}^{\text{angular}} = A_{B\alpha}^{(2)i}$$

A is essentially a 3-index/axis tensor⁴², as a function of pair-displacement R_{ijx} , consisting of an atomic-index i (lowercase Latin letters), atomic-state-index B (the ‘uncoupled’/hyperedge index, script-capital Latin letters), and α (the ‘coupled’ index, Greek-letters). After this tensor is found for-all atoms i , above $\alpha = \{n\ell\}$, a reshape for organization of the indices. Using this 2-body function we may obtain the many-body state for all atoms i by the following tensor-hyper-contraction:

$$A_{A\alpha\beta\dots\omega}^{(n)i} = \prod_{\alpha}^{n-1} A_{A\alpha}^{(2)i} \quad .$$

In order to obtain the energy or gradient, we require the tensor-hypercontraction of the n -body state-function ($n - 1$ 2-body state-functions) to the n -body expansion-coefficients, $c^{(n)}$:

$$\mathcal{E}_i = \overbrace{c_{\alpha}^{(1)B} A_{B\alpha}^i}^{\text{2-body}} + \overbrace{c_{\alpha\beta}^{(2)B} A_{B\alpha}^i A_{B\beta}^i}^{\text{3-body}} + \overbrace{c_{\alpha\beta\gamma}^{(3)B} A_{B\alpha}^i A_{B\beta}^i A_{B\gamma}^i}^{\text{4-body}} + \dots + \overbrace{c_{\alpha\beta\dots\omega}^{(n)B} \prod_{\alpha}^{n-1} A_{B\alpha}^i}^{\text{n-body}} \quad .$$

The expansion-coefficients ($c^{(1)}, c^{(2)}, \dots, c^{(n)}$) are universal given the basis-set (larger basis sets yield larger expansion coefficients), and act as the force-field parameters. As of this writing, in 2023, this is still an active area of research, specifically on the connection of machine-learning potential into the ACE, Lysogorskiy et al. 2023. The primary idea is to use ACE formalism to constrain the machine-learning space, Batzner et al. 2022; Batatia et al. 2022.

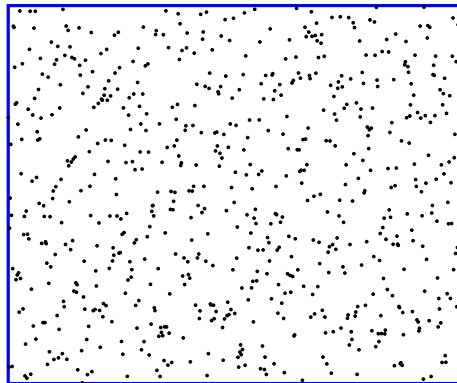
⁴²Quite reminiscent to the MPS tensors.

6.3.3 PES Hopping

Classical dynamics flows on a single PES layer. However, one can also conceive of pseudo-classical dynamics which hops between layers. The PES hopping can occur via excitations or spontaneously, as done in Tully 1990. The precise PES Hopping amplitude is given by the off-diagonal terms of the Non-Adiabatic-Coupling (NAC) vector⁴³. Approximations to this amplitude based on the diagonal components and the nuclei's kinetic-energy are given by Landau-Zener model, which has been modified by Nakamura and coworkers, Nakamura 2012.

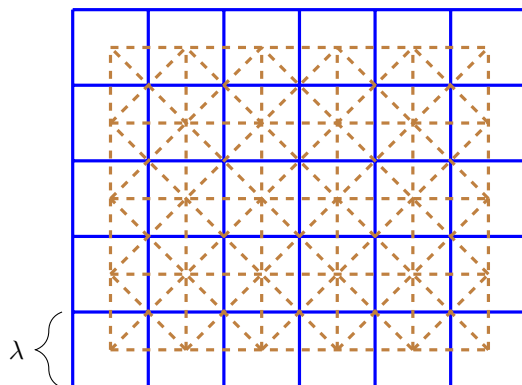
6.4 Cell-list and NN-interactions

Suppose we have a 2-dimensional (can be arbitrary dimension) system, with all particles located in a region given by the blue-box.



⁴³Also known as the derivative-coupling.

Now let's partition our system into smaller pieces:



Within each box, i.e. intra-interactions, all possible interactions are computed (hence dense). However, inter-box, i.e. inter-interactions, are sparse and these are given by the following triangular-lattice (dashed-brown connections above) that includes the typical cubic Nearest-Neighbor (NN) and Next-Nearest-Neighbors (NNN), these interactions form a network/graph, whose structure is captured by a sparse-tensor Λ_{IJ} (symmetric with interchange $I \leftrightarrow J$). It's worthy to note that any graph/network, defined by Λ_{IJ} , of interactions may be computed in this manner (including those with periodic-boundary-conditions).

This partition is computed by projecting the re-scaled 1-body-coordinates R_{ix} to integers⁴⁴. In order to determine the box in which the atom belongs we sparse-reshape (tuple-to-integer, discussed in a previous chapter, shape being the shape of the box stack):

$$B^i = \text{reshape}([\lambda R^{ix}]) \quad .$$

⁴⁴Such that the cell-list is in units of the scale.

This vector B^i , yields the parent-box for every atom, however, the inverse can be easily found (by the inverse-argument-sort, and domain-search as given in sparse-partial-tracing). This ultimately leads to the irregular sparse-tensor⁴⁵: C_i^I , a yielding given the box-enumeration and belonging atoms. Hence initializing a cell-list scales pseudo-linearly $\sim N \log N$, and should be re-initialized based on anticipated drift.

Now let's consider the effect of the cutoff/scale-choice λ . Suppose we have N atoms with M boxes (that are evenly divided by λ) then for bulk-atoms our interaction reduces the number of interactions by $\frac{9}{M^2}$ (more boxes, less of the total interaction is considered). The cutoff/scale is due to approximate-zero London interactions and is approximately: $\lambda \sim 15 \text{ \AA}$ for real-biological systems. This algorithm may be compared to other neighbor lists like Verlet-Neighbor-List. Suppose we have a Verlet-Neighbor-List (radial-cutoff) cutoff of λ_{Verlet} , then our cell-list scale is $\lambda_{\text{cell-list}} = \frac{2}{3} \lambda_{\text{Verlet}}$.

For many-body interactions, the cutoff may be made much smaller because as these interactions have a higher power-law decay in their interaction (go to zero much faster). Therefore, we may use another cell-list, with smaller cutoff $\lambda \sim \ell_{\text{atomic bond}}$, to generate all possible pair interactions. Because we have already considered all pair-interactions, these can be used to determine which many-body-interactions we wish to include. In the spirit of the simplicial-complex paradigm, all many-body interactions considered are those which are *born* from those pair-interactions.

⁴⁵Precise data-structure is that of PyTree, jagged/ragged-tensor.

6.5 Classical Molecular Dynamics Algorithm

Now let's describe our combined algorithm to compute the molecular-dynamics for potentially excited state molecules. We abbreviate this algorithm Fitted-Interpolated-Tabulated-Molecular-Dynamics (FIT-MD). Let's begin by describing the straight-forward pair-interactions. The system over consideration has the following characteristics:

- State: $\{R^{ix}, A^i\}$: Coordinates & Unary-atom-types,
- Cell-list: $\{\Lambda_{IJ}, B_{Jj}\}$: cell-list & cell-domain,
- Potential: $\{\mathcal{E}^{naR}, u\}$: tabulated-potential, potential-units.

From these we wish to obtain \mathcal{E} , f_{ix} , and g^{ar} : Energy (scalar), forces (3-vector for every atom), and pair-correlation function respectively. We will demonstrate how FIT-MD, once again, reduces to sparse-dense tensor-contractions.

6.5.1 $\sim N^2$ Pair-force Calculation

We begin by computing the displacement-matrix (R_{ijx}), and from it the distance matrix (R_{ij}) decomposed into its integer part N_{ij} and remainder decimal part dx_{ij} :

$$R_{ijx} = R_{ix} - R_{jx} = (R_{ix} \otimes 1_j - 1_i \otimes R_{jx})_{ijx}$$

$$N_{ij}, dx_{ij} = R_{ij} = |R_{ijx}|_{ij} = \left| \sum_x (R_{ijx})^2 \right|_{ij} .$$

Therefore we need N_{ij} , dx_{ij} , R_{ijx} . Next, we need to compute the pair-type a_{ij} from the two unary atom-types $\{A_i, A_j\}$. This is done by the sparse-reshape (symmetric

including diagonal) of the Cartesian-product (\parallel denotes the concatenation operation):

$$A_{ij\sigma} = (A_i \times A_j) = ((A_i \otimes 1_j) \parallel (1_i \otimes A_j))_{ij\sigma} \quad \rightarrow \quad A_{ij\sigma} \rightarrow a_{ij} \quad .$$

All together, we require: $\mathcal{O}_{\text{space}} \sim 6N^2$ in memory. Using these we may compute⁴⁶ the energy \mathcal{E} and force f_{ix} (with linear interpolation):

$$\mathcal{E} = \frac{1}{2} \sum_{ij} \left(\mathcal{E}[0, a_{ij}, N_{ij}] - \mathcal{E}[1, a_{ij}, N_{ij}] \frac{dx_{ij}}{u} \right)_{ij}$$

$$f_{ij} = (\mathcal{E}_{nAR}[1, a_{ij}, N_{ij}] + (\mathcal{E}_{nAR}[1, a_{ij}, N_{ij} + 1] - \mathcal{E}_{nAR}[1, a_{ij}, N_{ij}]) * dx_{ij})$$

$$f_{ix} = f_{ij} R_{ijx} \quad .$$

6.5.2 Sparse Pair-force Calculation

We follow a similar procedure as the $\sim N^2$ algorithm, but within each cell pair I and J :

$$R_{ijx}^{IJ} = \Lambda^{IJ} (B_I^i R^{ix} - B_J^j R^{jx})$$

$$N_{ij}^{IJ}, dx_{ij}^{IJ} = R_{ij}^{IJ} = |R_{ijx}^{IJ}|_{ij}^{IJ} \quad .$$

Next, we compute the pair-atomic-types:

$$A_{ij\sigma}^{IJ} = (A_i^I \times A_j^J) = ((A_i^I \otimes 1_j^J) \parallel (1_i^I \otimes A_j^J))_{ij\sigma} \quad \rightarrow \quad A_{ij\sigma}^{IJ} \rightarrow a_{ij}^{IJ} \quad .$$

⁴⁶Note the composition: $(\mathcal{E}[0, a_{ij}, N_{ij}])_{ij}$, yields a tensor of the same shape of the nested tensor.

Similarly, using these we may compute the energy \mathcal{E} and force f_{ix} :

$$\begin{aligned}\mathcal{E} &= \frac{1}{2} \sum_{IJij} \left(\mathcal{E}[0, a_{ij}^{IJ}, N_{ij}^{IJ}] - \mathcal{E}[1, a_{ij}^{IJ}, N_{ij}^{IJ}] \frac{dx_{ij}^{IJ}}{u} \right) \\ f_{ij}^{IJ} &= \left(\mathcal{E}_{nAR}[1, a_{ij}^{IJ}, N_{ij}^{IJ}] + (\mathcal{E}_{nAR}[1, a_{ij}^{IJ}, N_{ij}^{IJ} + 1] - \mathcal{E}_{nAR}[1, a_{ij}^{IJ}, N_{ij}^{IJ}]) * dx_{ij}^{IJ} \right)_{ij}^{IJ} \\ f_{ix} &= f_{ij}^{IJ} R_{ijx}^{IJ} \quad .\end{aligned}$$

Each cell-pair $\{I, J\}$ is parallelizable (achieved over a parallel-for-loop over IJ). Therefore in a distributed computing setting, the potential is known \mathcal{E}^{naR} along all nodes along with the relevant atomic-parameters, e.g. coordinates.

6.5.3 For Many-body Force Calculation

The objective, although not explicitly implemented was to include 3-body, 4-body, and perhaps higher order atomic terms, in order to model molecular covalent bonding. For Lennard-Jones particles, we may define this pair-wise interaction easily, however the potential function such as $\mathcal{E}^{naR_1R_2R_3}$ (3-body, 3-body-type a) could also be treated. Determining the proper form of $\mathcal{E}^{naR_1R_2R_3}$, and higher order terms, is still a challenge and active area of research.

6.5.4 An Example Run

We consider a $18 \times 12 \times 10$ FCC unit-cell Argon crystal dynamics using the aforementioned algorithm. We compute the energy as a function of time to demonstrate stability, fig. 23, and the correlation function, fig. 24, to demonstrate the dynamics.

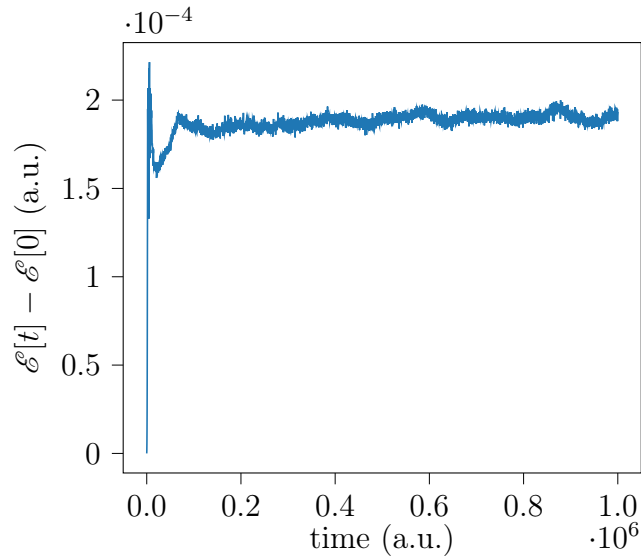


Figure 23. Plotted is the Energy-difference (When Compared to the First Time-step) in Time. Ideal Situation is a Perfectly Horizontal Line. This is On 1 CPU For A 10 Minute Simulation of Approximately 4300 Argon Atoms for 5000 Time-steps. The Algorithm is the Aforementioned FIT-MD, With Linear-interpolation.

6.6 Application in Radiation-Damage

Let's consider the Lennard-Jones solid (a solid whose large-scale cohesion is dominated by weak-correlation effects, *e.g.* rare-gas-solids, with experimental data shown in Klein and Venables 1977) irradiated by electron beams. When irradiated by electronic xor x-ray beams, we obtain excitations whose relative strength is captured by the inelastic spectra. Therefore, the experimental Electron-Energy-Loss-Spectrum (EELS), Egerton 2011, gives us hints on the excitation pathways, these are given

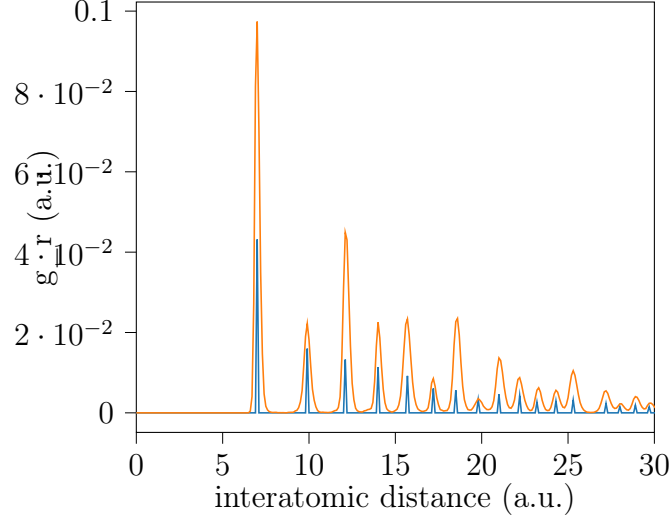


Figure 24. Plotted is the Radial-Distribution-Function (RDF) of the 1st Time-step (Blue) and Last (Orange), of the Same Simulation in Fig. 23. The Initial Stage is a Perfect Crystal, and the Last Instance is a Slightly Thermalized Sample.

in fig. 25. We notice two kinds of peaks: excitonic (of Frenkel type, long-lived ~ 1 ns) and an overdamped-*plasmonic* (density-oscillations with multiple peaks of broad and short-lived, ~ 1 fs, type). The local excitonic peaks are the highest narrow peaks in all spectra, in fig. 25, they do a singlet-triplet split with greater atomic number. The excitonic-peaks are electronic peaks to the left of the largest-width peak (the *plasmon*). The delocalized/large-impact-parameter *plasmon* peaks have the dominant cross-section for electron beams, the integral of these Lorentzian-like peaks. The precise mechanism of the overdamped-*plasmonic* decay is an interesting problem. However, we have simplified the discussion to consider two *plasmonic* pathways/resultants: excitons or phonons (which ultimately yielded to an incoherent temperature increase).

The smallest accessible electronic state is the 1st excitonic state, this results in a new PES between itself and the other atoms. This is straight-forward PES scan on *the excited state*. However, this *excited state* was a non-gas phase state, rather the atom submersed in a macroscopic dielectric. Two approaches may be tried to obtained this state: incorporation of a Screening Model, Klamt 2011 or solving a modified Hartree-Fock algorithm.

Although, the diffraction pattern is captured in Fourier-space, the Real-Space correlation is sufficient to extract results (à la Plancherel's theorem). Therefore we conduct Real-Space FIT Molecular-Dynamics simulations to include the effect of stochastic excitonic-excitations and temperature-increase, to determine this changing correlation-function, similar to those on fig.24. The result was in determining the *sublimation-point* with-respect-to radiation exposure $\left(\frac{e}{\text{\AA}^2}\right)$ (this sublimation-point is duration dependent), work is in Appendix A. This work was suppose to be a stepping stone for future work in covalently-bonded matter, *i.e.* molecules. In protein/polymer EELS, the EELS contain similar features: excitons (charge-transfer, those with larger dielectric function) and the familiar overdamped-plasmons.

6.6.1 The Exciton & Excimer Model

Now let's return to our exciton-model. We model, excitons to be excited states of molecules while submersed in a uniform dielectric. With respect to *ab initio* methods, this may be accomplished in a variety of ways. Here we introduce one of these. This

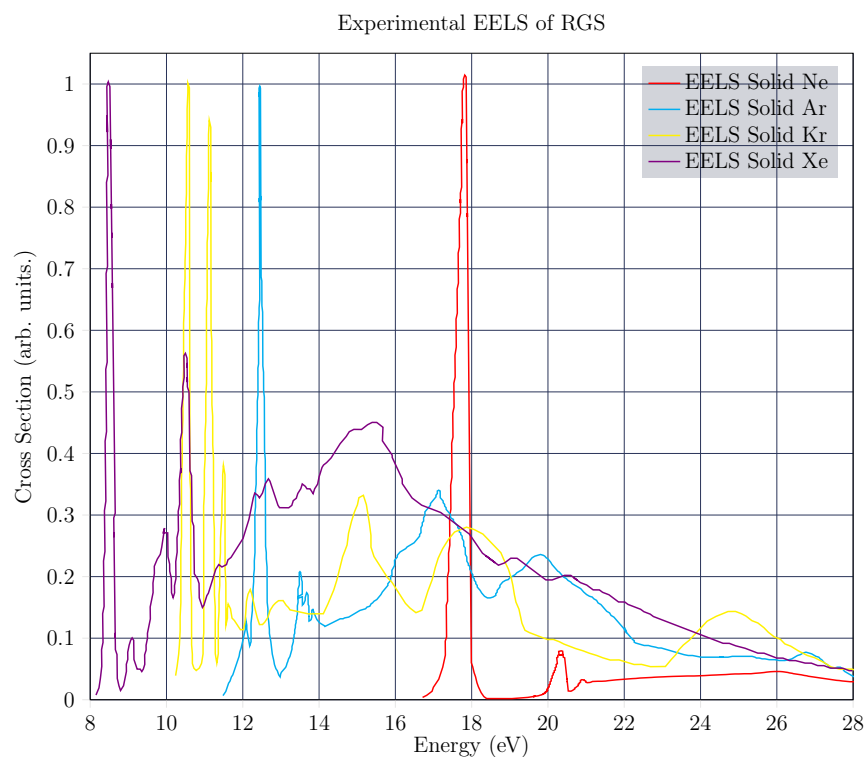


Figure 25. This Figure Shows the EELS spectrum of RGSs as Published in Klein & Venebles, for Neon (Red), Argon (Cyan), Krypton (Yellow), and Xenon (Violet). Vertical-axis is Re-scaled With-respect-to Each RGS Spectra (Highest Peak of Each is Rescaled to ‘1’).

may easily be achieved in Hartree-Fock (HF) theory⁴⁷ using the domain-decomposition COnductor-like Screening MOdel (ddCOSMO) *solvation-model*, introduced by Klamt 2011; Cancès, Maday, and Stamm 2013. The solution to this theory ψ is then used as a reference in Coupled-Cluster-Singles-Doubles (CCSD), a well-known method to capture weak-correlation effects of separated systems (*e.g.* of two separated Argon

⁴⁷With a psuedo-potential/Effective-Core-Potentials (ECP) in order to constrain later post-HF methods

atoms). However, this yields the ground state, to obtain excited states, Equation-of-Motion-CCSD (EoM-CCSD) is used (in addition to the ground-state provided by CCSD). The ground-state, 1st excited-state, and the lowest double *excimer* excited state (denoting a double single excitation, 1 excitation on either atom) is used for the PES.

6.7 Real-Time Molecular-Dynamics

Next, we release the electronic degrees-of-freedom, only to freeze the nuclei, to directly interact with a real-time electromagnetic field (only for nuclei to be influenced by a changing electronic field). This is an active area of research in electronic-structure theory, largely called Real-Time (RT) methods, Li et al. 2020, and are useful for studying strong fields as produced by bright lasers or concentrated electron beams, in order to study nonlinear light-matter effects.

In RT, we can define the driving Hamiltonian-operator $H(t)$, and take its matrix-exponential, to form unitary evolution operators $U(t, t') = \exp(-i\hbar H(t)dt)$ (upto truncation of the Magnus series)⁴⁸. As we've mentioned before the Hamiltonian-operator is very sparse, however their matrix-exponential is not, while both (before and after) state-vector are assumed to be dense. Therefore the trick of RT theory is to use the sparse operator's information to interpolate between both dense state-vectors.

⁴⁸Such that $\psi(t') = U(t', t)\psi(t)$

6.7.1 Real-Time H_2^+ Coherent Excitation

The simplest example of RT-HF dynamics is the coherent excitation of a 2-state model, in this case H_2^+ in the minimal basis sto-3g, provides this model. This model was previously calculated by Li, Smith, et al. 2005. The ground state being the bonding σ -orbital and the other (excited) state being the anti-bonding σ^* -orbital. The differences between the two Potential Energy Surfaces, given by the interatomic separation is shown below, fig. 26.

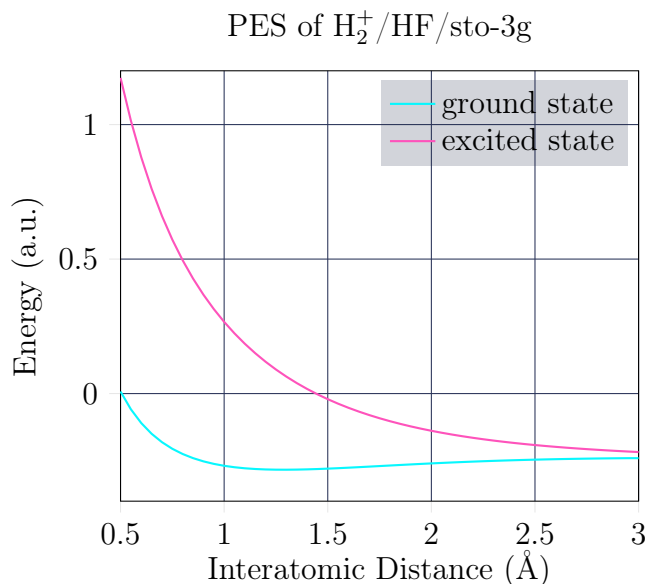


Figure 26. Above is the PES of H_2^+ /HF/STO-3G Basis, Showing Both Electronic States, σ and σ^* .

At rest (the minimum PES value), the energy difference is $\delta E = 0.4764$. Therefore by tuning to this frequency interesting dynamics can occur. So given the \mathcal{E} -field

($\omega_0 = 0.4764$ a.u. = 12.9635 eV = 95.64 nm light, $|\mathcal{E}| = 0.05$ a.u.), we intend to drive the cation's electronic dynamics (while holding the nuclei still). The results are shown in figure 27, initially the ground state is fully occupied, as irradiation continues the excited state gets populated (while the ground state gets de-populated). Eventually, oscillations occurs at the Rabi frequency $\omega = 0.4764$.

$$P_1(t) = 1 - P_2(t) \tag{6.3}$$

$$P_2(t) = \left(\frac{\Gamma^2}{\Gamma^2 + \frac{(\omega - \omega_0)^2}{4}} \right) \sin^2(\omega_{12}t) \tag{6.4}$$

the occupation of the σ and σ^* orbitals follow the $\cos^2(\omega t)$ and $\sin^2(\omega t)$ shape as suggested by equations 6.3 and 6.4. This is shown in fig. 27, which shows the time-varying electric field and a Runge-Kutta integration schemes for the RT dynamics with time-step $dt = 0.2$ a.u.

6.7.2 Ehrenfest Dynamics

Now lets finally release the nuclei, along with the electrons, to obtain the full molecular-dynamics (light, electronic, and nuclei dynamics). As mentioned in an earlier section, there are two paths of achieving this with semi-classical nuclei: Tully (surface-hopping) xor Ehrenfest (mean-field) Dynamics, Li, Tully, et al. 2005. Here we consider the latter, which defines the classical potential from the incoherent sum of the potentials weighted by the electronic density-matrix $f_{\text{classical}}(r, t) = f_{pp}(r)D_{pq}(t)$. Applying this to the aforementioned H_2^+ /sto-3g system we obtain fig. 28(a) for a

variety of electric-field-strengths. As is easily noticed the period oscillation slowly grows with increasing field-strength until an asymptotic point corresponding to an oscillation with infinite period, indicating a broken bond. The next figure 28(b) shows the period of oscillation as a function of the electric-field strength, clearly showing this asymptote.

6.8 Real-Time Radiation-Damage

Radiation damage is an unavoidable outcome of experimental probing of real systems. It manifests as intrinsic noise in experimental data, stemming from undesired information, which, in our case, is inelastic scattering (For spectroscopists, the situation is reversed, with elastic scattering considered noise, e.g., in phononic spectroscopy). Unlike the RGS simulation, we consider molecular scale effects of irradiation. The real-time formalism above is suitable for single-reference, or weakly, correlated systems, however as we've mentioned bond-breaking requires the strong correlation formalism solved statically by CASPT2 or DMRG/Tensor-Networks, as developed by Chan and Sharma 2011; Ma, Schollwöck, and Shuai 2022b. These methods can be implemented for small-scale molecules/systems, this is different from the earlier Lennard-Jones analysis. Ultimately, if sufficiently bright-beams, in the long-time limit, we expect the system to be completely transition into the Warm-Dense-Matter (WDM) state, as described in Dornheim, Groth, and Bonitz 2018;

Dornheim et al. 2023. Although interesting for nuclear and astro-physical systems, by the time the WDM state is reached, the original condensed system is nonexistent (and any additional probe only characterizes this state). Either way, sparsity⁴⁹ undoubtedly plays an important and necessary role in untangling the general problem of nonequilibrium and nonlinear dynamics associated with Radiation-Damage.

⁴⁹In: atomic distribution, electronic Hilbert-space, PES-interpolation, real-time propagation, and etc... as shown in this work.

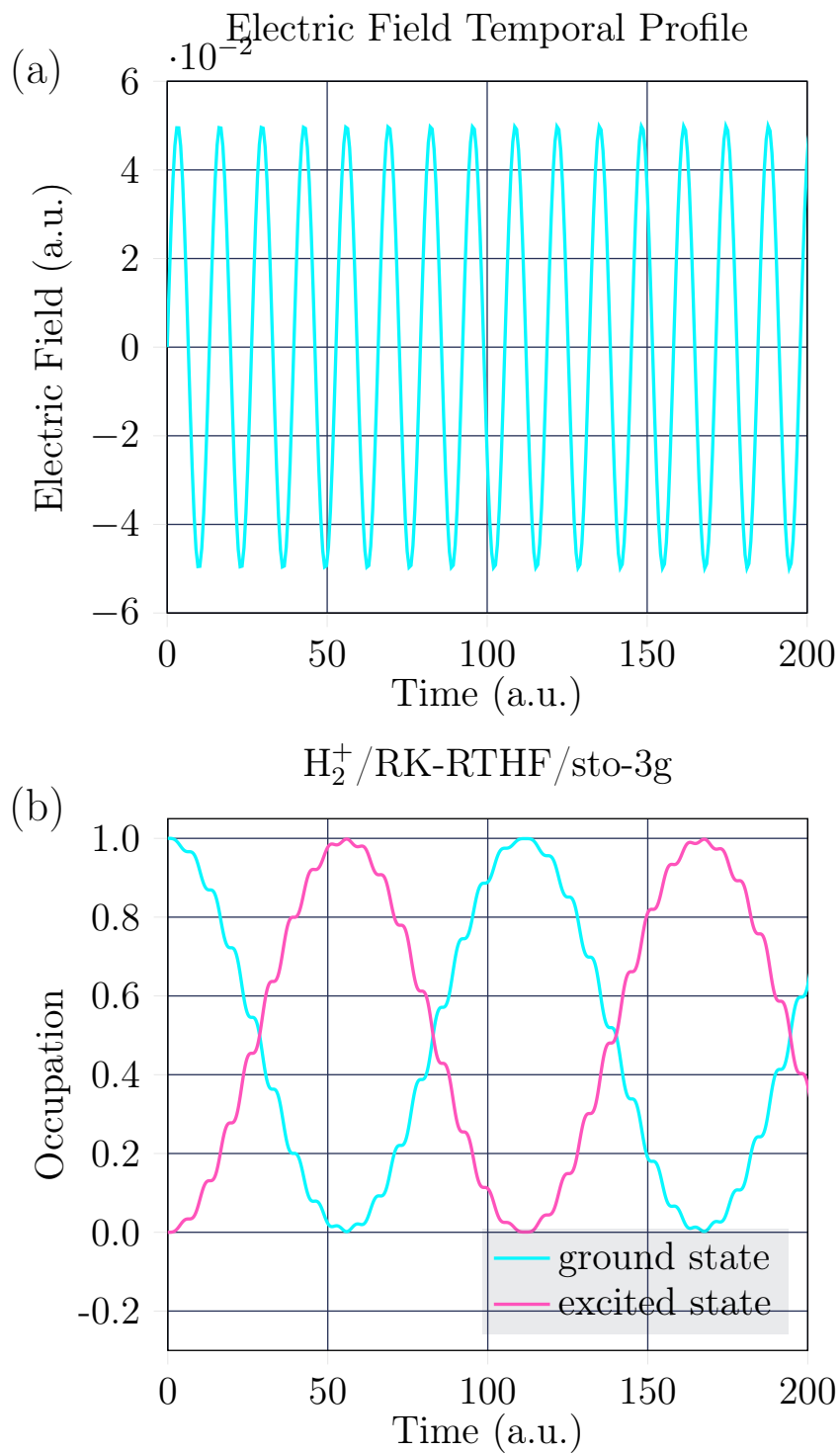


Figure 27. Above are Shown the Irradiation By the Electric-field Shown (Top) of the 2-state H_2^+ , By Real-Time Runge-Kutta (An Approximation for the Unitary Integration) Integration (bottom).

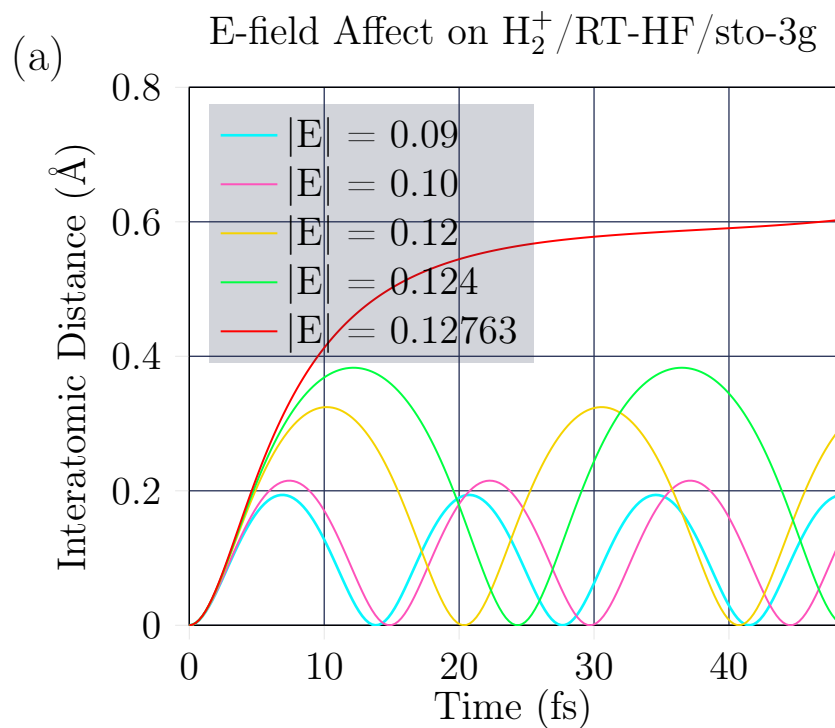


Figure 28. The Top Plot Shows the Interatomic Distance of the Same Cation, Displaying the Dynamics Under Various Pulse Intensities. This Clearly Shows an Increase to the Molecules Vibrational Period, Until an Infinite Period is Achieved at the Asymptotic Limit Leading to Dissociation. While the Bottom Plot Displays a Function of Vibrational Period to the Exposure Electric Field Strength.

Chapter 7

CONCLUSION

The original motivation for developing and researching these algorithms was radiation damage, where bright sources illuminate molecules, leading to nonlinear interactions with the sample. Understanding these nonlinear interactions is the core of radiation damage study, in both Ultrafast-Xray-Diffraction (UXD) and Ultrafast-Electron-Diffraction (UED)⁵⁰ The motivating idea is *diffraction-before-destruction* method was suggested by the Janos group, Neutze et al. 2000, (and others earlier) for x-ray lasers (UXD). These experiments use the diffraction pattern to capture structural (also dynamic, *i.e.* spectra) information. Understanding these mechanisms required a quantum understanding of electron motion, involving multiple disciplines. It became apparent that tensor contraction methods and sparsity are widespread in chemistry, physics, and computer science, all crucial for this research. The primary goal of this thesis is to emphasize the importance of sparse tensor methods for practical computational physics. We have illustrated three applications in quantum chemistry, tensor networks, and molecular dynamics.

Although sparse-tracing algorithms presented here give a performance boost for very sparse arrays on a single CPU, the primary difficulty with unstructured-sparse-tensor contractions is the *Parallelization-issue*, as noticed by Kanakagiri and Solomonik 2023. Unlike dense-array contractions which are essentially trivially

⁵⁰Also for their ‘Microscope’ mode.

parallelizable, sparse-array contractions involve irregular, potentially of a large extent, pieces which are difficult to distribute evenly along a *distributed computing* setting (itself a sparse-tensor, a network). This is the same difficulty for dense-sparse array-contractions. One limitation of current sparse-sparse tensor/array algorithms is the apparent lack of suitable Application-Specific Integrated Circuit (ASIC) development. These algorithms rely on a branched programming architecture, like binary search, which makes them suitable for CPU-like tasks rather than GPU tasks. It's worth noting that the entire CPU core functionality might not be required for the entire sparse-sparse algorithm.

Pioneered by White 1992, the use of tensor decompositions for QMB (Quantum Many-Body) operators and states provides a systematic method to solve exponentially difficult QMB problems approximately while increasing accuracy. This forms the core of the tensor network field. In their compressed form, these operators still exhibit extreme unstructured array sparsity, primarily filled with zeros. Compressed forms of QMB-states typically consist of a tensor-network of dense-tensors. Symmetries are crucial in transforming these dense tensors into block-sparse structures. However, it's possible to discover even sparser tensor networks (with a higher contraction-edge/node ratio), further compressing the QMB state. Future work may involve finding such sparser tensor networks, like MERA, beyond the one explored here, which is the MPS. Electronic structure in quantum chemistry, especially in its strong-correlation (multi-reference) form, is an exponentially difficult task, as it's a Quantum Many-Body (QMB) problem. We've developed a method to construct a Selected Configuration Interaction (SCI) method, a specific type of active-space method, to accurately choose

relevant configurations for future work. This active space is analogous to the MPS bond dimension but offers more precise selection.

Sparsity is observed in molecular dynamics where the potential energy surface can be sparsely tabulated and interpolated, atom interactions are limited to a cutoff neighborhood, and force calculations can be represented using sparse tensor hypercontractions. The main challenge lies in connecting strongly correlated *ab initio* chemical information with atomic cluster expansion coefficients.

These methods are however completely general for many other uses. It is interesting that although, the universe has a strictly “dense” description⁵¹, it is somehow approximately described by a *sparse* description.

⁵¹Every interaction is included, e.g. an apple on Earth interacts with M87.

REFERENCES

- Affleck, Ian. 1985. “Large- n Limit of $SU(n)$ Quantum “Spin” Chains.” *Phys. Rev. Lett.* 54 (10): 966–969. <https://doi.org/10.1103/PhysRevLett.54.966>.
- Affleck, Ian, and Andreas W. W. Ludwig. 1991. “Universal noninteger “ground-state degeneracy” in critical quantum systems.” *Phys. Rev. Lett.* 67 (2): 161–164. <https://doi.org/10.1103/PhysRevLett.67.161>.
- Ananthanarayanan, Rajagopal, Steven K Esser, Horst D Simon, and Dharmendra S Modha. 2009. *The Cat is out of the Bag: Cortical Simulations with 10^9 neurons, 10^{13} synapses.*
- Anderson, K, Per Aake Malmqvist, and B Roos. 1990. “Second-order perturbation theory with a CASSCF reference function.” *J. Phys. Chem. C* 94:5483.
- Arnold, Douglas N, Richard S Falk, and Ragnar Winther. 2006. “Finite Element Exterior Calculus, Homological Techniques, and Applications.” *Acta numerica* 15:1–155.
- Azevedo, Frederico AC, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. 2009. *J. Comp. Neurol.* 513 (5): 532–541.
- Bachmayr, M, G Csanyi, R Drautz, G Dusson, S Etter, C van der Oord, and C Ortner. 2019. “Atomic Cluster Expansion: Completeness, Efficiency and Stability.” arXiv:1911.03550, *arXiv*.
- Barabási, Albert-László, and Réka Albert. 1999. “Emergence of Scaling in Random Networks.” *Science* 286 (5439): 509–512.
- Batatia, Ilyes, Simon Batzner, Dávid Péter Kovács, Albert Musaelian, Gregor NC Simm, Ralf Drautz, Christoph Ortner, Boris Kozinsky, and Gábor Csányi. 2022. arXiv:2205.06643, *arXiv*.
- Batzner, Simon, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. 2022. *Nature Communications* 13 (1): 2453.

- Beach, K. S. D., Fabien Alet, Matthieu Mambrini, and Sylvain Capponi. 2009. “SU(N) Heisenberg model on the square lattice: A continuous- N quantum Monte Carlo study.” *Phys. Rev. B* 80 (18): 184401. <https://doi.org/10.1103/PhysRevB.80.184401> {10.1103/PhysRevB.80.184401}.
- Behler, Jörg, and Michele Parrinello. 2007. “Generalized neural-network representation of high-dimensional potential-energy surfaces.” *Phys. Rev. Lett.* 98 (14): 146401.
- Bennett, Charles H., Herbert J. Bernstein, Sandu Popescu, and Benjamin Schumacher. 1996. “Concentrating partial entanglement by local operations.” arXiv:quant-ph/9604024, *Phys. Rev. A* 53 (4): 2046–2052. <https://doi.org/10.1103/PhysRevA.53.2046>.
- Bethe, Hans. 1931. “Zur theorie der Metalle: I. Eigenwerte und eigenfunktionen der linearen atomkette.” *Zeitschrift für Physik* 71 (3-4): 205–226.
- Bollobás, Béla. 1998. *Modern Graph Theory*. Vol. 184. Springer Science & Business Media.
- Booth, George H, Alex JW Thom, and Ali Alavi. 2009. *J. Chem. Phys.* 131 (5).
- Born, Max. 1955. “Statistical Interpretation of Quantum Mechanics.” *Science* 122 (3172): 675–679.
- Born, Max, Werner Heisenberg, and Pasqual Jordan. 1926. “On Quantum Mechanics II.” *Z. Phys* 35 (8-9): 557–615.
- Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, et al. 2018. *JAX: composable transformations of Python+NumPy programs*. V. 0.3.13. <http://github.com/google/jax>.
- Bretto, Alain. 2013. *Hypergraph Theory: An Introduction*. Springer.
- Bridgeman, Jacob C, and Christopher T Chubb. 2017. “Hand-waving and Interpretive Dance.” *J. Phys. A* 50 (22): 223001.
- Brower, Richard C, George Fleming, Andrew Gasbarro, Timothy Raben, Chung-I Tan, and Evan Weinberg. 2016. “Quantum Finite Elements for Lattice Field Theory.” *arXiv preprint arXiv:1601.01367*.

- Brower, Richard C, George T Fleming, Andrew D Gasbarro, Dean Howarth, Timothy G Raben, Chung-I Tan, and Evan S Weinberg. 2021. “Radial Lattice Quantization of 3D ϕ^4 field theory.” *Phys. Rev. D* 104 (9): 094502.
- Buczyński, Jarosław, and Joseph M Landsberg. 2013. “Ranks of Tensors and a Generalization of Secant Varieties.” *Linear Algebra and its Applications* 438 (2): 668–689.
- Calabrese, Pasquale, and John Cardy. 2004. “Entanglement Entropy and Quantum Field Theory.” arXiv:hep-th/0405152, *JSTAT* 2004 (06): P06002.
- Cancès, Eric, Yvon Maday, and Benjamin Stamm. 2013. “Domain Decomposition for Implicit Solvation Models.” *J. Chem. Phys.* 139 (5).
- Candanedo, Julio. 2023a. “notes on Generalized Configuration-Interaction in python.” chemrxiv-2023-9gch7, *chemRxiv* 32.
- . 2023b. “Sparse Partial-Tracing.” *arXiv preprint arXiv:2303.10784*.
- Candès, Emmanuel J, Justin Romberg, and Terence Tao. 2006. “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information.” *IEEE Transactions on information theory* 52 (2): 489–509.
- Chan, Garnet Kin-Lic, and Sandeep Sharma. 2011. “The Density Matrix Renormalization Group in Quantum Chemistry.” *Annual review of physical chemistry* 62:465–481.
- Choy, Christopher, JunYoung Gwak, and Silvio Savarese. 2019. *4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks*.
- Davidson, Ernest R, and William J Thompson. 1993. “Monster Matrices: their eigenvalues and eigenvectors.” *Computers in Physics* 7 (5): 519–522.
- Dill, Ken, and Sarina Bromberg. 2010. *Molecular Driving Forces: statistical thermodynamics in biology, chemistry, physics, and nanoscience*. Garland Science.
- Dornheim, Tobias, Simon Groth, and Michael Bonitz. 2018. “The uniform electron gas at warm dense matter conditions.” *Physics Reports* 744:1–86.

- Dornheim, Tobias, Zhandos A Moldabekov, Kushal Ramakrishna, Panagiotis Tolias, Andrew D Baczewski, Dominik Kraus, Thomas R Preston, David A Chapman, Maximilian P Böhme, Tilo Döppner, et al. 2023. “Electronic Density Response of Warm Dense Matter.” *Physics of Plasmas* 30 (3).
- Drautz, Ralf. 2019. *Phys. Rev. B* 99 (1): 014104.
- Dummit, David Steven, and Richard M Foote. 2004. *Abstract algebra*. Vol. 3. Wiley Hoboken.
- Dzyaloshinskii, Igor E, Evgenii M Lifshitz, and Lev P Pitaevskii. 1961. “The General Theory of van der Waals Forces.” *Advances in Physics* 10 (38): 165–209.
- Egerton, Ray F. 2011. *Electron energy-loss spectroscopy in the Electron Microscope*. Springer Science & Business Media.
- Einstein, Albert. 1916. “The Foundation of the General Theory of Relativity.” <https://einsteinpapers.press.princeton.edu/vol6-trans/7>, on page 158. *Annalen der Physik* 49:769–822.
- El Gamal, Abbas, and Young-Han Kim. 2011. *Network Information Theory*. Cambridge University Press.
- Epifanovsky, Evgeny, Michael Wormit, Tomasz Kuś, Arie Landau, Dmitry Zuev, Kirill Khistyayev, Prashant Manohar, Ilya Kaliman, Andreas Dreuw, and Anna I Krylov. 2013. “New implementation of high-level correlated methods using a general block tensor library for high-performance electronic structure calculations.” *J. Comput. Chem.* 34,
- Fawzi, Alhussein, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. 2022. “Discovering faster matrix multiplication algorithms with reinforcement learning.” *Nature* 610 (7930): 47–53.
- Fradkin, Eduardo. 2013. *Field Theories of Condensed Matter Physics*. Cambridge University Press.
- Führinger, Max, Stephan Rachel, Ronny Thomale, Martin Greiter, and Peter Schmitteckert. 2008. “DMRG studies of critical $SU(N)$ spin chains.” arXiv:0806.2563, *Annalen der Physik* 520 (12): 922–936.

- Ganahl, Martin, Jackson Beall, Markus Hauru, Adam GM Lewis, Tomasz Wojno, Jae Hyeon Yoo, Yijian Zou, and Guifre Vidal. 2023. “Density Matrix Renormalization Group with Tensor Processing Units.” *PRX Quantum* 4 (1): 010317.
- Gauthé, Olivier. 2019. “Tensor Network Methods for SU(N) Spin Systems.” PhD diss., Université Paul Sabatier-Toulouse III.
- Georgi, Howard. 1999. *Lie algebras in particle physics*. Westview Press.
- Gray, Johnnie. 2018. “quimb: A python package for quantum information and many-body calculations.” *JOSS* 3 (29): 819.
- Gray, Johnnie, and Garnet Kin Chan. 2022. “Hyper-optimized compressed contraction of tensor networks with arbitrary geometry.” arXiv:2206.07044, *arXiv*.
- Gustavson, Fred G. 1978. “Two fast algorithms for sparse matrices: Multiplication and permuted transposition.” *ACM TOMS* 4 (3): 250–269.
- Haldane, F Duncan M. 1983. “Continuum dynamics of the 1-D Heisenberg antiferromagnet.” *Phys. Lett. A* 93 (9): 464–468.
- Halson, James J, Robert J Anderson, and George H Booth. 2020. arXiv:2007.11939v1, *Molecular Physics* 118 (19-20): e1802072.
- Hatcher, Allen. 2005. *Algebraic Topology*. Cambridge University Press.
- Hauru, Markus, and Guifre Vidal. 2018. “Uhlmann Fidelities from Tensor Networks.” arXiv:1710.05397v2, *Phys. Rev. A* 98 (4): 042316.
- Heisenberg, Werner. 1928. “Zur Theorie des Ferromagnetismus.” *Zeitschrift für Physik* 49:619–636.
- Herculano-Houzel, Suzana, and Roberto Lent. 2005. “Isotropic fractionator: a simple, rapid method for the quantification of total cell and neuron numbers in the brain.” *Journal of Neuroscience* 25 (10): 2518–2521.
- Herculano-Houzel, Suzana, Bruno Mota, and Roberto Lent. 2006. “Cellular scaling rules for rodent brains.” *PNAS* 103 (32): 12138–12143.

- Holmes, Adam A, Norm M Tubman, and CJ Umrigar. 2016. arXiv:1606.07453, *JCTC* 12 (8): 3674–3680.
- Israelachvili, Jacob N. 2011. *Intermolecular and Surface Forces*. Academic Press.
- Jouppi, Norman P, Doe Hyun Yoon, Matthew Ashcraft, et al. 2021. “Ten Lessons From Three Generations Shaped Google’s tpuv4i: Industrial Product.” In *Proceedings of the 48th annual ISCA*, 1–14. IEEE.
- Jouppi, Norman P, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, et al. 2017. “In-Datacenter Performance Analysis of a Tensor Processing Unit.” In *Proceedings of the 44th annual ISCA*, 1–12.
- Kanakagiri, Raghavendra, and Edgar Solomonik. 2023. “Distributed-memory DMRG via sparse and dense parallel tensor contractions.” arXiv:2307.05740, *arXiv*.
- Klamt, Andreas. 2011. “The COSMO and COSMO-RS solvation models.” *Wiley Interdiscip. Rev. Comput. Mol. Sci.* 1 (5): 699–709.
- Klein, Michael L., and John A. Venables. 1977. “Rare Gas Solids.” *Academic Press Inc. London*.
- Laflorie, Nicolas, Erik S Sørensen, Ming-Shyang Chang, and Ian Affleck. 2006. “Boundary Effects in the critical scaling of entanglement entropy in 1D systems.” arXiv:cond-mat/0512475, *Phys. Rev. Lett.* 96 (10): 100603.
- Lee, John M. 2012. *Introduction to Smooth Manifolds*. Springer.
- Levine, Daniel S, Diptarka Hait, Norm M Tubman, Susi Lehtola, K Birgitta Whaley, and Martin Head-Gordon. 2020. arXiv:1912.08379, *JCTC* 16 (4): 2340–2354.
- Li, Xiaosong, Niranjana Govind, Christine Isborn, A Eugene DePrince III, and Kenneth Lopata. 2020. “Real-time time-dependent electronic structure theory.” *Chem. Rev.* 120 (18): 9951–9993.
- Li, Xiaosong, Stanley M Smith, Alexei N Markevitch, Dmitri A Romanov, Robert J Levis, and H Bernhard Schlegel. 2005. “A time-dependent Hartree–Fock approach for studying the electronic optical response of molecules in intense fields.” *PCCP* 7 (2): 233–239.

- Li, Xiaosong, John C Tully, H Bernhard Schlegel, and Michael J Frisch. 2005. “Ab initio Ehrenfest dynamics.” *J. Chem. Phys.* 123 (8).
- Lifshitz, Evgenii M. 1956. “The Theory of Molecular Attractive Forces between Solids.” *J. Exp. Theor. Phys.* 2 (1): 73–83.
- Liu, Jin-Guo, Zhao-Long Gu, Jian-Xin Li, and Qiang-Hua Wang. 2017. arXiv:1609.09309, *New J. Phys.* 19.
- Lysogorskiy, Yury, Anton Bochkarev, Matous Mrovec, and Ralf Drautz. 2023. *Phys. Rev. Mater.* 7 (4): 043801.
- Ma, Haibo, Ulrich Schollwöck, and Zhigang Shuai. 2022a. *Density Matrix Renormalization Group (DMRG)-Based Approaches in Computational Chemistry*. Elsevier.
- . 2022b. *DMRG-Based Approaches in Computational Chemistry*. Elsevier.
- Martinez, Todd J. 2010. “Seaming is Believing.” *Nature* 467 (7314): 412–413.
- McCulloch, Ian P. 2007. arXiv:cond-mat/0701428, *J. Stat. Mech.* 2007 (10): P10014.
- McCulloch, Ian P, and Miklós Gulácsi. 2001. “Total spin in the Density Matrix Renormalization Group algorithm.” *Philosophical Magazine Letters* 81 (6): 447–453.
- Menzel, Randolph, and Martin Giurfa. 2001. *Trends in Cognitive Sciences* 5 (2): 62–71.
- Munkres, James R. 2000. *Topology*. Vol. 2. Prentice Hall Upper Saddle River.
- Nakamura, Hiroki. 2012. *Nonadiabatic Transition: concepts, basic theories and applications*. World Scientific.
- Nataf, Pierre, and Frédéric Mila. 2014. “Exact Diagonalization of Heisenberg SU(N) models.” *Phys. Rev. Lett.* 113 (12): 127204.
- . 2018. arXiv:1802.05482, *Phys. Rev. B* 97 (13): 134420.
- Needham, Tristan. 2021. *Visual differential geometry and forms: a mathematical drama in five acts*. Princeton University Press.

- Neutze, Richard, Remco Wouts, David Van der Spoel, Edgar Weckert, and Janos Hajdu. 2000. "Potential for biomolecular imaging with femtosecond X-ray pulses." *Nature* 406 (6797): 752–757.
- Nielsen, Michael A, and Isaac L Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge university press.
- NumPy, Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, et al. 2020. "Array programming with NumPy." *Nature* 585, no. 7825 (September): 357–362. <https://doi.org/10.1038/s41586-020-2649-2>.
- Paszke, Adam, Sam Gross, and et al. 2019. "PyTorch." In *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, 8024–8035. Curran Associates, Inc.
- Pauli, Wolfgang. 1927. "Zur quantenmechanik des Magnetischen Elektrons." 43. http://neo-classical-physics.info/uploads/3/4/3/6/34363841/pauli_-_the_magnetic_electron.pdf.
- Pederson, Ryan, John Kozlowski, Ruyi Song, Jackson Beall, et al. 2022. "Large Scale Quantum Chemistry with Tensor Processing Units." *J. Chem. Theory Comput.*
- Penrose, Roger, and Wolfgang Rindler. 1984. *Spinors and Space-Time: Volume 1, Two-Spinor Calculus and Relativistic Fields*. Vol. 1. Cambridge University Press.
- Pfeifer, Robert NC, Glen Evenbly, Sukhwinder Singh, and Guifre Vidal. 2014. "NCON: A tensor network contractor for MATLAB." *arXiv preprint arXiv:1402.0939*.
- Phillips, James C, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D Skeel, Laxmikant Kale, and Klaus Schulten. 2005. "Scalable molecular dynamics with NAMD." *J. Comput. Chem.* 26 (16): 1781–1802.
- Raji, Joshua I, and Christopher J Potter. 2021. *PLoS One* 16 (5): e0250381.
- Rangamani, Mukund, and Tadashi Takayanagi. 2017. *Holographic Entanglement Entropy*. Springer.

- Roberts, Chase, Ashley Milsted, Martin Ganahl, Adam Zalcman, Bruce Fontaine, Yijian Zou, Jack Hidary, Guifre Vidal, and Stefan Leichenauer. 2019. *TensorNetwork: A Library for Physics and Machine Learning*. arXiv: 1905.01330 [physics.comp-ph].
- Rocktäschel, Tim. 2018. “Einsum is All you Need - Einstein Summation in Deep Learning.” <https://rockt.github.io/2018/04/30/einsum>.
- Rolandi, Alberto, and Henrik Wilming. 2020. “Extensive Renyi entropies in Matrix Product States.” arXiv:2008.11764, *arXiv*.
- Roos, Björn O, Per Linse, Per EM Siegbahn, and Margareta RA Blomberg. 1982. “A simple method for the evaluation of the second-order-perturbation energy from external double-excitations with a CASSCF reference wavefunction.” *Chemical Physics* 66 (1-2): 197–207.
- Ryan, Kerriane, Zhiyuan Lu, and Ian A Meinertzhagen. 2018. “The peripheral nervous system of the ascidian tadpole larva: Types of neurons and their synaptic networks.” *Journal of Comparative Neurology* 526 (4): 583–608.
- Saad, Yousef. 2003. *Iterative Methods for Sparse Linear Systems*. SIAM.
- Schlegel, Philipp, Yijie Yin, Alexander S Bates, Sven Dorkenwald, et al. 2023. 2023.06.27.546055, *bioRxiv*.
- Senftle, Thomas P, Sungwook Hong, Md Mahbulul Islam, Sudhir B Kylasa, Yuanxia Zheng, Yun Kyung Shin, Chad Junkermeier, Roman Engel-Herbert, Michael J Janik, Hasan Metin Aktulga, et al. 2016. “The ReaxFF reactive force-field: development, applications and future directions.” *npj Computational Materials* 2 (1): 1–14.
- Shapeev, Alexander V. 2016. arXiv:1512.06054, *Multiscale Modeling & Simulation* 14 (3): 1153–1173.
- Solomonik, Edgar, and Torsten Hoefler. 2015. “Sparse Tensor Algebra as a Parallel Programming Model.” arXiv:1512.00066, *arXiv*.
- Solomonik, Edgar, Devin Matthews, Jeff R Hammond, John F Stanton, and James Demmel. 2014. “A massively parallel tensor contraction framework for coupled-

- cluster computations.” *Journal of Parallel and Distributed Computing* 74 (12): 3176–3190.
- Stone, Anthony. 2013. *The Theory of Intermolecular Forces*. Oxford University Press.
- Strassen, Volker. 1969. “Gaussian Elimination is not Optimal.” *Numerische Mathematik* 13 (4): 354–356.
- TensorFlow, Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, et al. 2015. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.” Software available from [tensorflow.org](https://www.tensorflow.org/), <https://www.tensorflow.org/>.
- Tersoff, Jerry. 1988. “New empirical approach for the structure and energy of covalent systems.” *Phys. Rev. B* 37 (12): 6991.
- Teufel, Stefan. 2003. *Adiabatic Perturbation Theory in Quantum Dynamics*. Springer Science & Business Media.
- Thompson, Aidan P, H Metin Aktulga, Richard Berger, Dan S Bolintineanu, W Michael Brown, Paul S Crozier, Pieter J in’t Veld, Axel Kohlmeyer, Stan G Moore, Trung Dac Nguyen, et al. 2022. “LAMMPS-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales.” *Comput. Phys. Commun.* 271:108171.
- Thorne, Kip S, John Archibald Wheeler, and Charles W Misner. 2000. *Gravitation*. Princeton University Press.
- Tkatchenko, Alexandre, Robert A DiStasio Jr, Roberto Car, and Matthias Scheffler. 2012. “Accurate and efficient method for many-body van der Waals interactions.” *Phys. Rev. Lett.* 108 (23): 236402.
- Tomonaga, Sin-Itiro. 1997. *The Story of Spin*. University of Chicago Press.
- Tubman, Norm M, Joonho Lee, Tyler Y Takeshita, Martin Head-Gordon, and K Birgitta Whaley. 2016. *J. Chem. Phys.* 145 (4).
- Tully, John C. 1990. “Molecular Dynamics with Electronic Transitions.” *J. Chem. Phys.* 93 (2): 1061–1071.

- Turner, M Jon, Ray W Clough, Harold C Martin, and LJ Topp. 1956. “Stiffness and Deflection Analysis of Complex Structures.” *J. Aeronaut. Sci.* 23 (9): 805–823.
- Van Der Spoel, David, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E Mark, and Herman JC Berendsen. 2005. “GROMACS: fast, flexible, and free.” *J. Comput. Chem.* 26 (16): 1701–1718.
- Vanommeslaeghe, Kenno, Elizabeth Hatcher, Chayan Acharya, Sibsankar Kundu, Shijun Zhong, Jihyun Shim, Eva Darian, Olgun Guvench, P Lopes, Igor Vorobyov, et al. 2010. “CHARMM general force field: A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields.” *J. Comput. Chem.* 31 (4): 671–690.
- Verlet, Loup. 1967. “Computer “experiments” on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules.” *Phys. Rev.* 159 (1): 98.
- Verstraete, Frank, and J Ignacio Cirac. 2004. “Renormalization Algorithms for Quantum-Many-Body systems in two and higher dimensions.” arXiv:cond-mat/0407066, *arXiv*.
- Verstraete, Frank, Valentin Murg, and J Ignacio Cirac. 2008. “Matrix Product States, projected entangled pair states, and variational renormalization group methods for quantum spin systems.” arXiv:0907.2796, *Advances in Physics* 57 (2): 143–224.
- Wang, Junmei, Romain M Wolf, James W Caldwell, Peter A Kollman, and David A Case. 2004. “Development and Testing of a General Amber Force Field.” *J. Comput. Chem.* 25 (9): 1157–1174.
- Wang, Qing, Matthias Ihme, Yi-Fan Chen, and John Anderson. 2022. “A Tensor-Flow Simulation framework for Scientific Computing of Fluid Flows on Tensor Processing Units.” *Computer Physics Communications* 274:108292.
- Watts, Duncan J, and Steven H Strogatz. 1998. “Collective Dynamics of ‘small-world’ Networks.” *Nature* 393 (6684): 440–442.
- White, John G, Eileen Southgate, J Nichol Thomson, Sydney Brenner, et al. 1986. “The structure of the nervous system of the nematode *Caenorhabditis elegans*.” *Philos Trans R Soc Lond B Biol Sci* 314 (1165): 1–340.

- White, Steven R. 1992. “Density Matrix Formulation for Quantum Renormalization Groups.” *Phys. Rev. Lett.* 69 (19): 2863.
- . 1993. “Density-Matrix Algorithms for Quantum Renormalization Groups.” *Phys. Rev. B* 48 (14): 10345.
- Yu, Li-Wei, and Mo-Lin Ge. 2016. “ \mathbb{Z}_3 parafermionic chain emerging from Yang-Baxter equation.” arXiv:1507.05269, *Scientific Reports* 6 (1): 21497.
- Zhai, Huanchen, and Garnet Kin Chan. 2021. *J. Chem. Phys.* 154 (22).
- Zimmerman, Paul M. 2017. *J. Chem. Phys.* 146 (10).
- Zou, Yijian, Ashley Milsted, and Guifre Vidal. 2018. “Conformal data and renormalization group flow in critical quantum spin chains using periodic uniform matrix product states.” arXiv:1710.05397v2, *Phys. Rev. Lett.* 121 (23): 230402.

APPENDIX A
SPARSE-EINSUM BISUM†

[†]This was a paper submitted to the Journal of Open Source Software (JOSS) and arXiv.

A.0.1 Summary

In this work we introduce sparse-tensor contraction method in PyTorch analogous to `einsum` in NUMPY.

A.0.2 Statement of need

Among the many needs in high-performance scientific computing, two major problems arise: we must leverage sparse-data-structures and work with multidimensional-arrays (a parallelizable data-structure). When working with multidimensional-arrays, a clear-and-concise and universal manipulation is the `einsum` function of NumPy et al. 2020. However, there is not much work in the intersection of these needs. I.e. those which manipulate sparse-tensors/arrays as `einsum` does. Therefore this work remedies this need.

A.0.3 Overview of Functionality

As `einsum` stands for Einstein-Summation, `bisum` stands for Binary-Summation. The primary function of this package traces/contractions two tensors at a time (pair sequential-contraction is usually required for efficient contraction in multi-tensor traces) for types: sparse-sparse, sparse-dense, dense-sparse, and dense-dense (the original ‘`einsum`’ function). This function intakes a string, list-of-tensors, xor tensor; to describe the partial-tracing procedure. Key features include:

1. Efficient Tensor Operations: `bisum` excels in performing a variety of tensor operations, including summation, contraction, and element-wise multiplication, on large dense data structures. While minimizing the memory usage.
2. Sparse Data Focus: the program capitalizes on the idea that many real-world data-sets contain numerous zero values. `bisum` optimizes computations by ignoring these zero values, significantly reducing the computational load and improving execution speed. By eliminating calculations involving zero values, `bisum` reduces memory usage and speeds up computation times, making it

a valuable tool for applications involving massive data-sets. This work was originally motivated by earlier work Candanedo 2023b.

3. Streamlined Syntax: `bisum` introduces a user-friendly syntax that simplifies the representation of tensor operations. This enables users to express complex mathematical operations concisely and intuitively, contributing to improved code readability and maintainability.
4. Application Flexibility `bisum` finds applications in a wide range of fields, including scientific research, engineering, machine learning, signal processing, and more. Its efficiency and ease of use make it a versatile choice for various computational tasks. This involves uses in Machine-Learning, Scientific-Simulations (e.g. physics, chemistry, engineering, and etc...), and Signal-Processing.
5. Optimized for Real Data: While many computations involve zero-padding, `bisum` focuses solely on real data values, eliminating the need to iterate over zero entries. This targeted approach ensures that the program's performance is optimized for sparse/dense, real-valued data-sets. Much real-world data can be made sparse with adequate transformations.
6. Compatibility: `bisum` can be easily integrated into existing code-bases and workflows, complementing other computational libraries and tools. It is integrated with the popular Machine-Learning library PyTorch (Paszke, Gross, and al. 2019).

`bisum` fills a crucial niche in the computational landscape by providing a specialized solution for efficient tensor operations on large sparse/dense/mixed data structures. Its focus on sparse data values that significantly boosts performance, making it an indispensable tool for tackling complex calculations in various domains. Whether it's accelerating machine learning tasks or enhancing scientific simulations, 'bisum' offers a practical approach to optimizing computations while maintaining code simplicity and readability.

A.0.4 Usage: how to install

`bisum` is on the python-index, and therefore may be easily installed via the following command:

```
pip install bisum
```

A.0.5 Usage: how to import

`bisum` relies on sparse-tensors from PyTorch and therefore we import both libraries as such:

```
import torch
from torch import einsum
from bisum import bisum
```

and on PyTorch we can determine where we would like the tensor to live (on CPU xor GPU)

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

A.0.6 Usage: einsum-like example

We create natively dense random tensors (and can cast from to be sparse, via the `.to_sparse()` command):

```
A = torch.rand(8**3, device=device).reshape(8,8,8)
B = torch.rand(8**3, device=device).reshape(8,8,8)

torch.allclose( bisum("ijk,kjl", A.to_sparse(), B ), einsum("ijk,kjl", A,
    B ) )
torch.allclose( bisum("ijk,kjl", A.to_sparse(), B.to_sparse()
    ).to_dense(), einsum("ijk,kjl", A, B ) )
```

A.0.7 other label types example

`bisum` traces 2 sparse-tensors (`torch.tensor` objects) via 3 Tracing-prescriptions:

1. `einsum`-string (like `numpy.str`, labelling each tensor axis)
2. `ncon` (used in the tensor-network community, list of `1d-int-torch.tensor`, labelling each tensor axis, as described by Pfeifer et al. 2014)
3. `adjacency-matrix` (as in `numpy.tensordot`, `(2,n)` `2d-int-torch.tensor`, with `n` being the number of indices identified between

the two tensors). Suppose we would like to compute the following partial-trace/tensor-contraction $C_{njwl} = A_{iksndj}B_{wklsdi}$:

```
C_einsum = bisum("iksndj, wklsdi -> njwl", A, B)
C_ncon   = bisum([-1,-2,-3,4,-5,6],[1,-2,3,-3,-5,-1]), A, B)
C_adjmat = bisum(torch.tensor([[0,1,2,4],[5,1,3,4]]), A, B)

print(torch.allclose(C_einsum, C_ncon) and torch.allclose(C_ncon,
    C_adjmat))
```

while the pure tensor-product, \otimes is:

```
import numpy as np

C_einsum = bisum("abcdef, ghijkl", A, B)
C_ncon   = bisum([], A, B)
C_adjmat = bisum(torch.tensor([]), A, B)

print(np.allclose(C_einsum, C_ncon) and np.allclose(C_ncon, C_adjmat))
```

A.0.8 brief results when compared to `torch.einsum`

We did a quick comparison of this function (in the sparse-sparse mode) to PyTorch's native `einsum` function. The results of this comparison of relatively sparse-tensors is shown in fig. 29.

A.0.9 Development Notes

Currently, `bisum` is in alpha-stage (0.2.0) with code on: [github/bisum](#), and posted on the Python Package Index, or PyPI/[bisum](#). On here `bisum` has a MIT License. Although, `bisum` is a useful extension of `einsum`-function, more improvements are desired. Sparse-sparse matrix products on GPUs (tailor-made for dense-dense contractions) are relatively slow, sparse-dense contraction should be much faster. Also functionality on block-sparse or jagged/ragged/PyTree (irregularity shaped) sparse-tensors is desired.

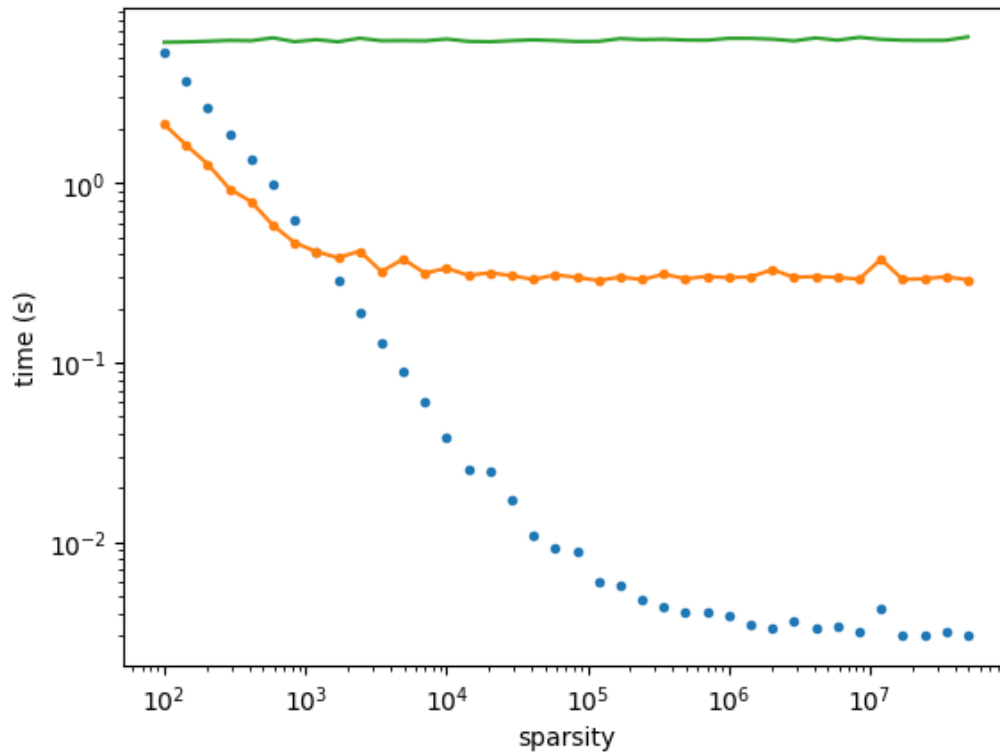


Figure 29. This plot shows a timing comparison between the `torch.einsum` (solid line, averaged over 2 samples) function, the sparse-dense `bisum` trace (connected dots, averaged over 5 samples) function, and finally the sparse-sparse `bisum` tensor contraction: $A_{qjwhkrjd}B_{krqljdmn}$ (each tensor of shape (14 14 14 14 14 14 14 14)) on a single CPU.

APPENDIX B
CHEMRXIV PAPER

notes on Generalized Configuration-Interaction in python

Julio Candanedo*

September 28, 2023

Abstract

In this work we construct a detailed understanding of the distribution of electrons as described by the Full-Configuration-Interaction (FCI). These results are presented at an introductory level for beginning practitioners or non-experts. Suitable background are a knowledge of linear-algebra and basic NumPy and Python principles of array manipulation.

Contents

1	Introduction	3
2	Many Body Theory	3
2.1	Solving the Many-Body Schrödinger Equation	4
2.2	Fock Space & Exterior Algebra Representation	5
2.3	Occupied Representation	5
2.4	Occupied Representation	6
2.5	Graphical SO Representation	6
2.6	Binary Representation	7
2.7	The Sign	7
2.8	Many-Body Hamiltonian	7
2.8.1	1-body, kinetic	8
2.8.2	1-body, external	8
2.8.3	2-body, electron-electron	8
2.8.4	Total CI Hamiltonian	8
3	Combinatorics	9
3.1	Unrestricted-FCI	9
3.2	Python <code>itertools</code>	9
3.3	The Cartesian Product	9
4	Active Space	9
5	SO-Matrix Elements	10
6	CI-Matrix Elements	11
6.1	Slater-Condon Information	11
6.2	Excitation/Deexcitation Operators	12
6.2.1	single excitations	13
6.2.2	double excitations	13
6.2.3	Orbitals in Common	13
6.3	CI Matrix Construction	13
7	Solving the CI Equation	14
7.1	direct diagonalization	14
7.2	Krylov-subspace method	14

*Department of Physics, Arizona State University, Tempe, AZ 85287, USA
jcandane@asu.edu / juliojcandanedo@gmail.com

8 Optimization Considerations	15
8.1 Slater-Condon Searching	15
8.2 CI Hamiltonian Diagonalization	15
9 Comparison and Examples	15
9.1 Sparsity of the CI Hamiltonian Matrix	15
9.2 Potential Energy Surfaces	15
10 Conclusion	16
11 Acknowledgements	17
A Python Subindexing	18
A.1 Subindexing/Subscripting	18
A.1.1 sparse-partial-tracing	18
B Raw Code	19
B.1 useful definitions	19
B.1.1 Determinant Manipulation	19
B.1.2 Slater-Condon Rules	20
B.1.3 SO Dressing	21
B.1.4 CI Dressing	22
B.2 <code>fci.py</code>	22
C Mean Field Theory	23
C.1 AO & MO Basis	23
C.2 Hartree-Fock	23
C.2.1 Objective	24
C.2.2 SCF Setup	24
C.2.3 SCF procedure	24
C.3 Adiabatic Kohn-Sham	25
C.3.1 Response-KS	25
C.4 Multicomponent	25

1 Introduction

The electronic-structure problem is amongst the most important problems in science, especially in chemistry, to determine the chemical properties *ab initio*, without the *a priori* need for empirical information. As an agreement between *ab initio* and empirical results constitute a complete understanding, here we desire to understand the *ab initio* theory.

In chemistry, we have two kinds of fundamental particles: electrons and nuclei. These have very different mass scales yet similar charge, therefore in this paradigm we may invoke the Born-Oppenheimer approximation, to separate nuclear and electronic degrees of freedom. This is not always an excellent approximation in all circumstances, but is often a suitable starting point in understanding the more general-setting. Because electrons must be treated quantum mechanically, the Schrödinger equation is required. Beyond hydrogenic atoms, molecules have multiple electrons, and hence the Many-Body Schrödinger-equation (MBSE) is the fundamental equation describing their distribution. Then this problem transposes into solving the nuclei-clamped static Schrödinger equation for the electrons. It is this problem we wish to understand at a basic level.

In order to compute this distribution in the computer, discretization is compulsory. For this we consider a Basis-Set-*Ansatz* (BSA), particularly we consider a Gaussian-Basis-Set. This theory is broadly known as the Configuration Interaction (CI). And like the full Schrödinger equation the theory is linear and is solved via Exact-Diagonalization (ED) of the Full-CI (FCI) Hamiltonian. The full-CI is an approximation of the full-SE (SE), for a finite basis set, whereas the SE is the idealistic equation describing the structure on the continuum (an infinite basis set).

The FCI Hamiltonian is very sparse and maybe constructed matrix-element by matrix-element for non-zero elements described by the well-known *Slater-Condon* rules, described in §6.1. These rules exist because physical Hamiltonians at the microscopic-level contain 1-particle (1-body) and 2-particle/pair (2-body) interactions. These 1-particle interactions are the kinetic energy or effective potential (interactions with particles that are not electrons, e.g. nuclei), while the 2-particle interactions are the inter-electron interactions. The equations implementing the Slater-Condon rules for explicitly restricted-systems are found in both [Szabo and Ostlund, 2012a] and [Helgaker *et al.*, 2014].

Because the FCI space is very large, i.e. scaling exponentially with the number of electrons, it is computationally intractable for most interesting systems (in chemistry and biochemistry). It is often truncated by excitations or a choice of an Active-Space descriptions, this is beyond the scope of this paper. If the theory is contracted to a single state, this is known as Mean-Field Theory, for the ground state this is the famous Hartree-Fock Theory. Solving the Mean-Field/Hartree-Fock Equation is the first step to solving the FCI equations here.

Although, the full-CI algorithm is not directly useful beyond academic interest, for its aforementioned exponential complexity, it is directly useful to calibrate new hypotheses/approximations to the Many-Body Schrödinger Equation, that may be useful. Our goal is to reduce CI theory into a game, combinatorics followed by a set of rules, and with it to construct a Hamiltonian from the subset of Slater-Determinants $\Lambda_{\text{CI}} \subset \Lambda_{\text{FCI}}$.

Additionally, this paper describes the implementation of the CI equations in NUMPY with the help of [PySCF *et al.*, 2018], to give Atomic Orbital Integrals, to hopefully connect and render CI-theory more transparent. The FCI tool of [Psi4NumPy *et al.*, 2018] was initially consulted, but proved to be unhelpful for the author's purposes. Note that Einstein-summation-notation, i.e. sum-of-repeated-indices, is used through-out the text.

2 Many Body Theory

Our objective is to understand the Many-Body Schrödinger-equation for electrons, romantically put:

$$H\Psi = \mathcal{E}\Psi$$

with H being the Many-body Hamiltonian, Ψ being the many-body *wavefunction* (a complex vector), and \mathcal{E} being the many-body excitation energies.

2.1 Solving the Many-Body Schrödinger Equation

Now let's attempt to solve the static Schrödinger equation for the electronic distribution. The most straight-forward solution of the Schrödinger equation for the molecular Hamiltonian is on a cubic grid. Therefore, suppose we have a cubic grid with N nodes on a side, i.e. N^3 grid-points total, representing a grid of side ℓ with resolution dx with n -electrons, then we require

$$\text{grid-points}(\Psi^{\text{Schrödinger}}) = (N^3)^n = N^{3n} = \left(\frac{\ell}{dx}\right)^{3n},$$

to represent Ψ ! If the grid-points are represented by a complex NUMPY array¹. Let's solve for the neutral Boron atom ($n = 5$ electrons) with $\ell = 2.2 \text{ \AA}$ and $dx = 0.1 \text{ \AA}$ (very compact and coarse)².

$$\text{memory}(\Psi^{\text{Schrödinger}}) = 16 (22)^{3 \cdot 5} \approx 2 \times 10^{21} \text{ Bytes} \sim 2 \text{ ZB},$$

$$\text{memory}(\Psi^{\text{Schrödinger}}) = \frac{1}{2} \text{ internet-traffic in 2018!}$$

In more practical terms a Becke (non-uniform) grid might be more useful (this is also used in DFT calculations), but this exponential need for memory still remains an issue. The first major innovation to solving the Schrödinger equation is the Mean-Field approximation on 1-body wavefunctions, for electrons (Fermions more generally), the Hartree-Fock approximation (when including *exchange*), as is explained in §C.2. This reduces the memory substantially to:

$$\text{grid-points}(\Psi^{\text{Hartree-Fock}}) = nN^3 = n \left(\frac{\ell}{dx}\right)^3,$$

$$\text{memory}(\Psi^{\text{Hartree-Fock}}) \approx 852 \text{ kB}.$$

Three more innovations (DFT, LCAO, and Gaussian-orbitals) were introduced in the mid-20th century to alleviate this extreme cost. Two pioneers (of many) Walter Kohn and John Pople were awarded the Nobel Prize in Chemistry in 1998 for their contributions. Walter Kohn was awarded for this work on developing DFT, using functionals on 1-body densities to capture many-body correlations. This in principle leads to a reduction in memory to:

$$\rho_{\text{grid-points}}^{\text{DFT}} = N^3 = \left(\frac{\ell}{dx}\right)^3,$$

$$\text{memory}(\rho^{\text{DFT}}) \approx 170.4 \text{ kB}.$$

It was clear early-on that grid based techniques, run into a computational bottleneck. Because electrons clump close to the point charge nuclei, they are usually described by central-distribution analytic functions, e.g. Slater-functions (e.g. $\exp(-|r|)$). For multi-atom (multi-centered) molecules, Roothaan suggested using Linear Combinations of Atomic Orbitals (LCAO or Molecular Orbitals MO), to model this kind of electron amplitude distribution. This is the basis-function ansatz. Pople (and others) developed and promoted Gaussian-function Basis Sets (GBS) for rapid integration and manipulation. He popularized this approach with his Quantum Chemistry code GAUSSIAN. Gaussian-functions are used because they are popular well-behaved central-distribution, which have analytic integrals (unlike the analytic Slater functions). Within the given basis-set, the Schrödinger equation becomes the Configuration Interaction (CI) equation. Despite the challenge of solving even the Boron atom above, these innovations have allowed for routine large-scale *ab initio*³ simulations of proteins, with 10,000+ basis-functions. For a recent implementation see [Seritan *et al.*, 2021]. Appendix A discusses a few different kinds of basis-sets and how to obtain them from online databases. Before we discuss details of an Self-Consistent-Field (SCF) calculations lets describe these atomic/molecular integrals.

¹With 8 Bytes/double-float (64-bit float), and 16 Bytes/double-complex (64-bit complex). With KB= 2^{10} B, MB= 2^{20} B, GB= 2^{30} B, etc...

²See Cisco's news-release <https://newsroom.cisco.com/press-release-content?articleId=1955935>.

³KS-DFT without physically motivated functionals can hardly be called *ab initio*.

2.2 Fock Space & Exterior Algebra Representation

This subsection may seem more abstract, but serves an essential foundation. Let's rewrite the MBSE in a more useable form:

$$\begin{aligned} H(\mathbb{X})\Psi(\mathbb{X}) &= \mathcal{E}\Psi(\mathbb{X}) \\ H(\mathbb{X})(\mathbb{X}|\Psi) &= \mathcal{E}(\mathbb{X}|\Psi). \end{aligned}$$

Above \mathbb{X} denotes $3N$ coordinates and spin, and let \mathbf{x} denote 3-dimensional coordinates and spin, and let x denote the Cartesian direction. What is the structure of Ψ ? As introduced by Slater and Fock in the 1920s, recognizing the Fermionic character of electrons, many-body electronic-wavefunctions must be wholly anti-symmetric under particle-swapping. This is done by the n -electron Slater Determinant of 1-electron wavefunctions ψ :

$$\Lambda(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = \frac{1}{n!} \begin{vmatrix} \psi_1(\mathbf{x}_1) & \psi_2(\mathbf{x}_1) & \cdots & \psi_n(\mathbf{x}_1) \\ \psi_1(\mathbf{x}_2) & \psi_2(\mathbf{x}_2) & \cdots & \psi_n(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\mathbf{x}_n) & \psi_2(\mathbf{x}_n) & \cdots & \psi_n(\mathbf{x}_n) \end{vmatrix}.$$

For those whom are familiar with the exterior-algebra, this is repeated use of the exterior/anti-symmetric product:

$$\Lambda(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = \Lambda(\mathbb{X}) = (\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n)(\mathbb{X}) = \bigwedge_i^n \psi_i(\mathbb{X}).$$

The similarity of the symbols \wedge and Λ (Greek-letter), is the reason why it will be used to represent a single Slater-Determinant. More generally; our many-body wavefunction are an n -form on N dimensional complex vector space:

$$\Psi \in \bigwedge^n (\mathbb{C}^N) \quad .$$

With N being the number of grid points xor basis functions, and n the number of electrons/Fermions. This is indeed true for HF determinant (the Fermi-vacuum state, all lowest orbitals occupied first). However, Ψ being an arbitrary n -form is a linear combination of all possible n -forms, therefore:

$$\Psi = \sum \bigwedge^n \psi \quad .$$

Each determinant is a volume element in that dimension. The full-CI wavefunction therefore resembles finding a perimeter. For example, suppose we have a system of $N = 4$ orbitals (xor grid points), and $n = 2$ electrons, then the wavefunction is a linear combination:

$$\Psi = c_1 \psi_1 \wedge \psi_2 + c_2 \psi_1 \wedge \psi_3 + c_3 \psi_1 \wedge \psi_4 + c_4 \psi_2 \wedge \psi_3 + c_5 \psi_2 \wedge \psi_4 + c_6 \psi_3 \wedge \psi_4 \quad ,$$

and hence a 2-form, with coefficients c_i being the CI-coefficients we desire for a given state. Total finite dimensional Fock-space is the full exterior algebra of dimension of the number of SO orbitals, i.e. 2^N for N SOs, while the CI-space is the subset of occupied space, $\binom{N}{m}$.

2.3 Occupied Representation

Instead of writing the anti-symmetric product, the information can be kept track of using Dirac notation, along with the determinant sign, and the many-body anti-symmetric/wedge product may be written as:

$$|\psi_1 \psi_2 \cdots \psi_n\rangle.$$

Before we proceed we need to determine representations of these determinants, in particular how to represent them in the computer.

The occupied-representation is useful to determine the precise MOs to evaluate 1-body and 2-body Hamiltonians to the CI matrix element. For example, suppose we have the following occupied determinant, for a given CI-state \hat{I} over occupied MOs i :

$$\left| \psi_1 \psi_3 \psi_4 \cdots \psi_i \right\rangle \sim [1 \ 3 \ 4 \ \cdots \ i] = \Lambda_{\hat{I}}.$$

2.4 Occupied Representation

Instead of writing the anti-symmetric product, the information can be kept track of using Dirac notation, along with the determinant sign, and the many-body anti-symmetric/wedge product may be written as:

$$\bigwedge_{i \in \Lambda} \psi_i = |\psi_1 \psi_2 \cdots \psi_n\rangle.$$

Before we proceed we need to determine representations of these determinants, in particular how to represent them in the computer. The occupied-representation is useful to determine the precise MOs to evaluate 1-body and 2-body Hamiltonians to the CI matrix element. For example, suppose we have the following occupied determinant, for a given CI-state \hat{I} over occupied MOs i :

$$\left| \psi_1 \psi_3 \psi_4 \cdots \psi_i \right\rangle \sim [1 \ 3 \ 4 \ \cdots \ i] = \Lambda_{\hat{I}}.$$

As we treat the spins independently, two sets of determinants, $(\Lambda^\uparrow, \Lambda^\downarrow)$, describe the full configuration, for CI states I , and occupied MOs i (in general for open-shell cases Λ^\uparrow and Λ^\downarrow may have different amount of occupied orbitals. Also within each spin-type each determinant may have different amount of occupied orbitals, i.e. Spin-Flip):

$$\Lambda_{I_i}^\uparrow, \quad \Lambda_{I_i}^\downarrow.$$

2.5 Graphical SO Representation

Graphically, Slater-Determinants may be represented by MO or SO diagrams. With $\{\bullet, \uparrow, \downarrow\}$ representing an occupied MO/SO orbital, while a vacancy represents an unfilled virtual orbital, an example is shown in fig. 1.

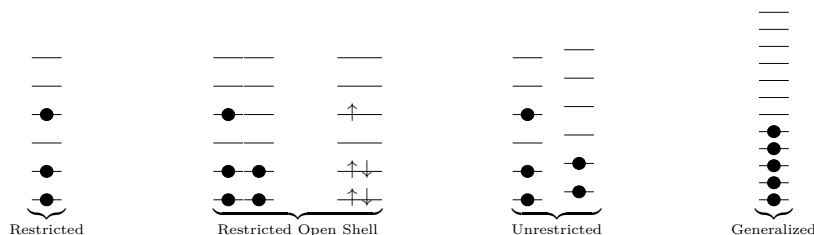


Figure 1: An example of a Restricted (R) Molecular-Orbital diagram (left). (center-left) For Restricted-Open (RO) Shell, two examples of MO and SO diagrams ($N = 6$ orbitals, and $n = 5$ electrons), showing occupation of various orbitals. For Unrestricted (U) states, the energies need-not be equal between the two different spins (center-right). If we have no restriction on spin, and desire to include spin-flip situations the Generalized (G) theory is given by the right diagram.

2.6 Binary Representation

Alternatively, we have the binary-representations of molecular-orbitals, which directly correspond to the occupation of each molecular orbital, and hence directly to the MO-diagram. Suppose we have an arbitrary CI-state I , denoted by \hat{I} , with the following representation:

$$\left| 0 \ 1 \ 0 \ 1 \ 1 \ \dots \ 0 \ 1 \ 0 \right\rangle \sim [0 \ 1 \ 0 \ 1 \ 1 \ \dots \ 0 \ 1 \ 0] = B_{\hat{I}p}.$$

The binary-representation will be useful for inter-determinant comparison. For each the spin, $\sigma = \{\uparrow, \downarrow\}$, case we have two binary-representations, representation the occupation of each Spin-Orbital (SO). Thus the full-CI-configuration is described by the following 3-dimensional array (for CI state I , spin σ , and MO orbital p):

$$B_{I\sigma p}.$$

2.7 The Sign

The determinant sign, is perhaps the trickiest part of computing CI-matrix elements. Fortunately, [Helgaker *et al.*, 2014], clears this issue on page 2-5, and uses Γ to denote the sign-convention of a determinant. Because annihilating (denoted by a , target MO is occupied) and creation (denoted by a^\dagger , target MO is vacant) operations are mutually exclusive they may be done with-in the same MO:

$$\begin{aligned} a_{\sigma m}^\dagger B_{I\sigma p} &= a_{\sigma m}^\dagger [p_0, p_1, \dots, 0_m, \dots, p_n] \\ &= \Gamma_{I\sigma m} [p_0, p_1, \dots, 1_m, \dots, p_n], \end{aligned}$$

while, for the annihilation operator:

$$\begin{aligned} a_{\sigma m} B_{I\sigma p} &= a_{\sigma m} [p_0, p_1, \dots, 0_m, \dots, p_n] = 0 \\ a_{\sigma m} B_{I\sigma p} &= a_{\sigma m} [p_0, p_1, \dots, 1_m, \dots, p_n] \\ &= +1 [p_0, p_1, \dots, 1_m, \dots, p_n]. \end{aligned}$$

Above $\Gamma_{I\sigma m} = +1$, “if there are an even number of electrons in the spin orbitals”⁴ left to the MO of interest. The method of determining this is a cumulative sum over the occupation.

In the code, we begin with the cumulative-sum over the MOs, $\left(\sum_p B_{I\sigma p}\right)_{I\sigma p}$, as previously shown this yields the correct sign for the the creation/vacant operations. Next, the annihilating/occupied parts of the MO (assuming $\Psi_{I\sigma i}$ are sorted in ascending order along i), must be replaced by a counting-up array.

$$\begin{aligned} C_{I\sigma p} &= \left(\sum_p B_{I\sigma p}\right)_{I\sigma p} \\ C_{I\sigma i} &\stackrel{!}{=} [0, 1, 2, \dots, n_i] \\ \Gamma_{I\sigma p} &= (-1)^{C_{I\sigma p}} = [I|p]^\sigma. \end{aligned}$$

The *sign*, for each determinant, is required because the information of the exact full anti-symmetric sequence was lost.

2.8 Many-Body Hamiltonian

The Many-Body Hamiltonian may be decomposed into 3 major parts: kinetic, external, and electron-electron interactions. The electron-electron interactions involve the interaction of two electrons.

⁴[Helgaker *et al.*, 2014]

2.8.1 1-body, kinetic

If the electrons are nonrelativistic, then they have a kinetic energy given by the Laplacian-operator⁵:

$$\begin{aligned}
H^{IJ} &= -\frac{\hbar^2}{2m} \int \Psi^{\dagger I}(\mathbf{x}) \Delta \Psi^J(\mathbf{x}) d\mathbf{x} \quad , \\
&= -\frac{\hbar^2}{2m} \gamma_{pq}^{IJ} \int \psi_p^{\dagger}(\mathbf{x}) \Delta \psi_q(\mathbf{x}) d\mathbf{x} \quad , \\
&= -\frac{\hbar^2}{2m} \gamma_{pq}^{IJ} C_{\alpha}^{\dagger p} C_{\beta}^q \underbrace{\int \phi_{\alpha}^{\dagger}(\mathbf{x}) \Delta \phi_{\beta}(\mathbf{x}) d\mathbf{x}}_{k_{\alpha\beta}} \quad .
\end{aligned}$$

2.8.2 1-body, external

Suppose we know the external distribution of particles. Then we may introduce their influence on a many-body electronic system as a 1-body interaction (as they involve interacting with 1-electron at a time). Most common types considered are nuclei-electron and light-electrons interactions. For nuclei-electron, the interaction is:

$$\begin{aligned}
H^{IJ} &= -\frac{Ze^2}{4\pi\epsilon_0} \int \frac{\Psi^{\dagger I}(\mathbf{x}) \Psi^J(\mathbf{x})}{|\mathbb{R}_i - \mathbf{x}|} d\mathbf{x} \\
&= -\frac{Ze^2}{4\pi\epsilon_0} \gamma_{pq}^{IJ} \int \frac{\psi_p^{\dagger}(\mathbf{x}) \psi_q(\mathbf{x})}{|\mathbb{R}_i - \mathbf{x}|} d\mathbf{x} \\
&= -\frac{Ze^2}{4\pi\epsilon_0} \gamma_{pq}^{IJ} \int \frac{(C_{\alpha}^{\dagger p} \phi_{\alpha}^{\dagger}(\mathbf{x})) (C_{\beta}^q \phi_{\beta}(\mathbf{x}))}{|\mathbb{R}_i - \mathbf{x}|} d\mathbf{x} \\
&= -\frac{Ze^2}{4\pi\epsilon_0} \gamma_{pq}^{IJ} C_{\alpha}^{\dagger p} C_{\beta}^q \underbrace{\int \frac{\phi_{\alpha}^{\dagger}(\mathbf{x}) \phi_{\beta}(\mathbf{x})}{|\mathbb{R}_i - \mathbf{x}|} d\mathbf{x}}_{h_{\alpha\beta}} \\
&= -\frac{Ze^2}{4\pi\epsilon_0} \gamma_{pq}^{IJ} C_{\alpha}^{\dagger p} C_{\beta}^q h_{\alpha\beta}
\end{aligned}$$

Above \mathbb{R}_i are the coordinates of the nuclei.

2.8.3 2-body, electron-electron

Because we are considering inter-electron contributions (as they are identical particles), this requires the 2-body electronic density. The explicit form is:

$$\begin{aligned}
H^{IJ} &= +\frac{e^2}{4\pi\epsilon_0} \iint \frac{\Psi^{\dagger}(\mathbf{x}, \mathbf{x}') \Psi(\mathbf{x}, \mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} d\mathbf{x} d\mathbf{x}' \quad , \\
&= +\frac{e^2}{4\pi\epsilon_0} \Gamma_{pqrs}^{IJ} \iint \frac{\psi_{pq}^{\dagger}(\mathbf{x}, \mathbf{x}') \psi_{rs}(\mathbf{x}, \mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} d\mathbf{x} d\mathbf{x}' \quad , \\
&= +\frac{e^2}{4\pi\epsilon_0} \Gamma_{pqrs}^{IJ} C_{\alpha}^{\dagger p} C_{\beta}^{\dagger q} C_{\gamma}^r C_{\delta}^s \underbrace{\iint \frac{\phi_{\alpha\beta}^{\dagger}(\mathbf{x}, \mathbf{x}') \psi_{rs}(\mathbf{x}, \mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} d\mathbf{x} d\mathbf{x}'}_{(\alpha\beta|\gamma\delta)} \quad ,
\end{aligned}$$

Above Γ_{pqrs}^{IJ} are the 2-body structure coefficients.

2.8.4 Total CI Hamiltonian

The total-CI Hamiltonian is the matrix:

$$H^{IJ} = \sum_{\sigma\tau pqrs} (\gamma_{pq}^{IJ\sigma} (p|q)^{\sigma} + \Gamma_{pqrs}^{IJ\sigma\tau} (pq||rs)^{\sigma\tau}) \quad .$$

⁵If relativistic, this is the Dirac-operator, which may be expanded to first order to the Laplace-operator, this is beyond the scope of this paper.

γ and Γ are the structural 1-body and 2-body coefficients respectively, they are independent of the molecular structure (all this information is captured in the 1- and 2- body AO Hamiltonians). These coefficients only depend on the combinatorics at hand. If the Mean-Field equations have been solved, then we can solve instead (by rotating the equations above in the AO basis):

$$H^{IJ} = \sum_{\sigma\tau\alpha\beta\gamma\delta} (\gamma_{\alpha\beta}^{IJ\sigma} (\alpha|\beta)^\sigma + \Gamma_{\alpha\beta\gamma\delta}^{IJ\sigma\tau} (\alpha\beta||\gamma\delta)^{\sigma\tau}) \quad .$$

3 Combinatorics

3.1 Unrestricted-FCI

The Combinatorics of FCI follow the *N-balls-in-M-slots* combinatorics (for identical balls with $N < M$). Except, the *slots* are called orbitals of the MO/SO/GO variety, and *balls* are called electrons. Here we consider SO-orbitals, this considers two independent *N-balls-in-M-slots* problems for \uparrow and \downarrow electrons separately. The number of combinations for the *N-balls-in-M-slots* problem is given by the *choose* expression:

$$\# \text{ states} = \binom{M}{N} = \frac{M!}{N!(M-N)!}.$$

Although it might be worthwhile to compute this for low M and N , the computer can do this for larger M and N . For unrestricted-FCI considered here we have:

$$\# \text{ states} = \binom{M}{N^\uparrow} \binom{M}{N^\downarrow} \quad .$$

3.2 Python itertools

Since both \uparrow and \downarrow systems are independent (we do not consider spin-flip situations here), we may consider without-loss-of-generality one of these systems. E.g. the \downarrow -system, in python using `itertools` we may create an array indexing all possible Molecular Orbitals (MO)⁶ and *choose* N^\downarrow elements, indicated the occupied MO orbitals. Python's `itertools` then generates all possible choices, this is exactly what we want for FCI. This yields the an array/matrix $\Lambda^{\downarrow Ip}$, and may be similarly done for $\Lambda^{\uparrow Ip}$.

3.3 The Cartesian Product

Suppose we have all the possible combinations for each spin type, \uparrow and \downarrow , i.e. $B^{\uparrow Ip}$ and $B^{\downarrow Ip}$ respectively. We would like to combine these two, to produce a complete list of determinants. In order to do this, it must be realized these systems independent, given any configuration, e.g. for determinant $\uparrow 7 B^{\uparrow Ip}$, we would like all the configurations for the \downarrow -type i.e. $B^{\downarrow Ip}$. Therefore, the approach is to concatenate (this operation is given by the two-vertical-lines) the following outer-product:

$$B^{\sigma Ip} = B^{\uparrow Ip} \times B^{\downarrow Ip} = \underbrace{B^{\uparrow Ip} \otimes 1^J}_{\sigma=0=\uparrow} \quad || \quad \underbrace{1^I \otimes B^{\downarrow Ip}}_{\sigma=1=\downarrow} \quad .$$

4 Active Space

Let's constrain the active-space, by introducing vector ξ_p (Boolean vector, `True` : part of active-space, `False` : not part of active-space) xor ξ_z (array enumerating active orbitals). The FCI combinatorics then may be applied within the occupied active-orbitals going to the vacant active-orbitals, this yields a set of $\{\Lambda_\xi^\uparrow, \Lambda_\xi^\downarrow\}$. These occupation values may be recast into the proper orbitals by the composition operation: $\Lambda^\uparrow = \xi[\Lambda_\xi^\uparrow]$ and $\Lambda^\downarrow = \xi[\Lambda_\xi^\downarrow]$. These then may be recombined with the reference $B^{\sigma p}$, they zeroing-out the active-orbitals and replacing with the new occupations (obtained via the composition operation).

⁶If there are 7 MOs, then this array is [0,1,2,3,4,5,6].

5 SO-Matrix Elements

Following our focus on the unrestricted-calculation, let's suppose we complete a unrestricted-SCF calculation, see the appendix, then we obtain the MO-coefficients $\{C^\uparrow, C^\downarrow\}$ which not only yield orthogonal states, but are minimized to a mean-field shielding of the electronic charges. This forms a foundation to build beyond the mean-field approximation, and these theories, including CI are called post-HF or post-SCF. Using these coefficients the 1-body and 2-body interactions may be formed by *dressing* the atomic-centric interactions given by the Atomic Orbital, i.e. AO-arrays. This dressed interaction is called the Spin-Orbit (SO) representation, and serves as the playground where CI games occur. That is to say, CI theory itself keeps the same level of SCF shielding throughout all electronic states⁷.

The dressing is implemented by matrix-products between the AO-representation and the Mean-Field shielding Orbital (MO) coefficients. Here we consider 1-body AO interaction represented by a 2-dimensional matrix/array $(\alpha|\beta)$ and a 2-body AO interaction represented by a 4-dimensional matrix/array $(\alpha\beta|\gamma\delta)$:

$$\begin{aligned} (\sigma p | \sigma' q) &= \sum_{\alpha\beta} (\alpha | \beta) C_{\sigma p}^\alpha C_{\sigma' q}^\beta \\ (\sigma p, \sigma' q || \tau r, \tau' s) &= \sum_{\alpha\beta\gamma\delta} (\alpha, \beta || \gamma, \delta) C_{\sigma p}^\alpha C_{\sigma' q}^\beta C_{\tau r}^\gamma C_{\tau' s}^\delta \\ &= \sum_{\alpha\beta\gamma\delta} (p, q || r, s)^{\sigma\sigma'\tau\tau'} \end{aligned}$$

Let's attempt to understand the 2-body interaction first. Here the complete SO-transformed MO inter-electron integrals are⁸:

$$\begin{aligned} (p, q | r, s)^{\sigma\sigma'\tau\tau'} &= \begin{pmatrix} (p, q || r, s)^{\uparrow\uparrow\uparrow\uparrow} & (p, q || r, s)^{\uparrow\uparrow\downarrow\downarrow} & (p, q || r, s)^{\uparrow\downarrow\uparrow\downarrow} & (p, q || r, s)^{\uparrow\downarrow\downarrow\uparrow} \\ (p, q || r, s)^{\downarrow\downarrow\uparrow\uparrow} & (p, q || r, s)^{\downarrow\downarrow\downarrow\downarrow} & (p, q || r, s)^{\downarrow\downarrow\uparrow\downarrow} & (p, q || r, s)^{\downarrow\downarrow\downarrow\uparrow} \\ (p, q || r, s)^{\uparrow\downarrow\uparrow\uparrow} & (p, q || r, s)^{\uparrow\downarrow\downarrow\downarrow} & (p, q || r, s)^{\uparrow\downarrow\downarrow\uparrow} & (p, q || r, s)^{\uparrow\downarrow\uparrow\downarrow} \\ (p, q || r, s)^{\downarrow\uparrow\uparrow\uparrow} & (p, q || r, s)^{\downarrow\uparrow\downarrow\downarrow} & (p, q || r, s)^{\downarrow\uparrow\downarrow\uparrow} & (p, q || r, s)^{\downarrow\uparrow\uparrow\downarrow} \end{pmatrix} \\ &= \begin{pmatrix} (p, q || r, s)^{\uparrow\uparrow\uparrow\uparrow} & (p, q || r, s)^{\uparrow\uparrow\downarrow\downarrow} & 0 & 0 \\ (p, q || r, s)^{\downarrow\downarrow\uparrow\uparrow} & (p, q || r, s)^{\downarrow\downarrow\downarrow\downarrow} & 0 & 0 \\ 0 & 0 & (p, q || r, s)^{\uparrow\downarrow\uparrow\downarrow} & (p, q || r, s)^{\uparrow\downarrow\downarrow\uparrow} \\ 0 & 0 & (p, q || r, s)^{\downarrow\uparrow\uparrow\downarrow} & (p, q || r, s)^{\downarrow\uparrow\downarrow\uparrow} \end{pmatrix}. \end{aligned}$$

However, those matrix-elements which have an odd-number of spin electrons are nonphysical and should be 0. Therefore the nonzero matrix-elements are:

$$\begin{aligned} (p, q | r, s)^{\sigma\tau} &= (\sigma p, \sigma q | \tau r, \tau s) \\ (p, q | r, s)_+^{\sigma\tau} &= (\sigma p, \tau q | \tau r, \sigma s) \\ (p, q | r, s)_-^{\sigma\tau} &= (\sigma p, \tau q | \sigma r, \tau s). \end{aligned}$$

A similar procedure is done for the 1-body AO matrices:

$$(p | q)^\sigma = (\sigma p | \sigma q) = (\alpha | \beta) C_{\sigma p}^\alpha C_{\sigma q}^\beta.$$

In unrestricted (Spin-Orbit) CI, we require only the spin-orbital transformed 1-electron and 2-electron operators, with (spin state given by $\sigma = \{\uparrow, \downarrow\}$):

$$\begin{aligned} (p | q)^\sigma &= \begin{pmatrix} h_{\alpha\beta} C_p^{\uparrow\alpha} C_q^{\uparrow\beta} \\ h_{\alpha\beta} C_p^{\downarrow\alpha} C_q^{\downarrow\beta} \end{pmatrix} \\ (pq || rs)^{\sigma\tau} &= \begin{pmatrix} (\alpha\beta || \gamma\delta) C_p^{\uparrow\alpha} C_q^{\uparrow\beta} C_r^{\uparrow\gamma} C_s^{\uparrow\delta} & (\alpha\beta || \gamma\delta) C_p^{\downarrow\alpha} C_q^{\downarrow\beta} C_r^{\uparrow\gamma} C_s^{\uparrow\delta} \\ (\alpha\beta || \gamma\delta) C_p^{\uparrow\alpha} C_q^{\uparrow\beta} C_r^{\downarrow\gamma} C_s^{\downarrow\delta} & (\alpha\beta || \gamma\delta) C_p^{\downarrow\alpha} C_q^{\downarrow\beta} C_r^{\downarrow\gamma} C_s^{\downarrow\delta} \end{pmatrix}. \end{aligned}$$

⁷Correcting for the excited state shielding and polarizability is done by Multiconfigurational Mean Field Methods, beyond standard CI.

⁸from the large blocks, we flip-spin from left-to-right on the 4th MO index, and flip-spin from up-to-down from the 2nd MO index.

6 CI-Matrix Elements

Next, we would like to take the SO-matrix elements (to keep our focus on the unrestricted case), of the previous section, and transform these into the CI-basis. This transformation needs to take into account the structure of the determinants. The structure of the determinants probes for nontrivial elements as suggested by the Slater-Condon rules, which probe the connectivity of the determinant space. In particular the 1-differences and 2-differences of the determinants. The Slater-Condon rules are entirely independent of the molecule involved, they only depend on the number of electrons and the basis space.

6.1 Slater-Condon Information

The Slater-Condon rules are dependent on the differences between determinants. In the unrestricted case, we may take tensor-differences of the binary representation of the Slater-determinants ($I \rightarrow J$):

$$\mathbb{B}_{IJ\sigma p} = B_{J\sigma p} - B_{I\sigma p}. \quad (1)$$

Given two CI states I and J , $\mathbb{B}_{IJ\sigma p}$ gives the difference of the binary MO occupations, p , for each spin σ . This matrix consists of 0 (no change), 1 (1 new occupation), xor -1 (1 less occupation). To get the difference-of-occupations for each spin we take the absolute-value for each element and sum (over MOs):

$$S_{IJ\sigma} = \frac{1}{2} \left(\sum_p |\mathbb{B}_{IJ\sigma p}| \right)_{IJ\sigma}.$$

This will be useful to determine the type of Slater-Condon rules required for each CI Hamiltonian matrix element I, J . For each spin-component the possible non-zero differences (a given I, J) in the the determinants are given (e.g. $[\delta\uparrow, \delta\downarrow]$):

rule 0	rule 1	rule 2	
		[2, 0]	(2)
	[1, 0]		
[0, 0]		[1, 1]	
	[0, 1]		
		[0, 2]	

For differences greater than $\delta\uparrow + \delta\downarrow \geq 3$, have a matrix-element is 0 (rule 3). This yields 7 Slater-Condon rules in U-CI. For each of the 6 terms above (plus rule 3) apply a *partition* of the entire CI matrix. This may be seen by the example on fig. 2, for 4 Slater-Condon rules (each column above, in eq. 2). The CI Hamiltonian matrix-elements may be partitioned into seven 1-dimensional arrays. The value of these matrix-elements are determined by a formula on the next page. As the 0-difference rule is always located on the matrix-diagonal It may be used to initialize the CI Hamiltonian with 0s in the off diagonal (covering rule 3). Next, for the remaining 5 rules, the indices of the CI Hamiltonian for their applicability are determined by (determined by the Spin-Difference Matrix):

$$\begin{aligned} I^\uparrow, J^\uparrow \text{ such that } S [I^\uparrow, J^\uparrow] &= [1, 0] \\ I^\downarrow, J^\downarrow \text{ such that } S [I^\downarrow, J^\downarrow] &= [0, 1] \\ I^{\uparrow\uparrow}, J^{\uparrow\uparrow} \text{ such that } S [I^{\uparrow\uparrow}, J^{\uparrow\uparrow}] &= [2, 0] \\ I^{\uparrow\downarrow}, J^{\uparrow\downarrow} \text{ such that } S [I^{\uparrow\downarrow}, J^{\uparrow\downarrow}] &= [1, 1] \\ I^{\downarrow\downarrow}, J^{\downarrow\downarrow} \text{ such that } S [I^{\downarrow\downarrow}, J^{\downarrow\downarrow}] &= [0, 2], \end{aligned}$$

note for a symmetric (in $I \leftrightarrow J$) spin-difference matrix $S_{IJ\sigma}$ this is double counting⁹. This algorithm scales as $\mathcal{O} \sim N^2$, with N being the number of CI states (which in-turn scales exponentially). In principle, the sparsity of the CI algorithm scales suggests a more optimum approach.

⁹E.g. for every pair $i \in I^\uparrow, j \in J^\uparrow$, there exists a pair $j \in I^\uparrow, i \in J^\uparrow$

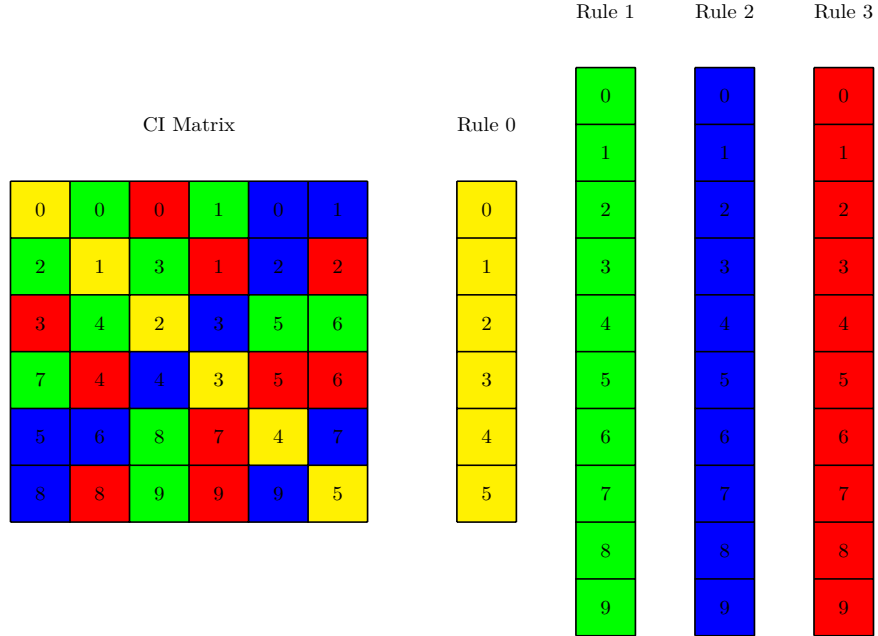


Figure 2: Above is an example of decomposing a 6×6 CI Matrix into 4 1-dimensional arrays, according to which Slater-Condon Rule is satisfied: rule 0 (yellow), rule 1 (green), rule 2 (blue), and rule 3 (red).

For an example, let's consider we have a system of 3 spin-orbitals (SO) per spin-type, and 2 \uparrow -electrons and 1 \downarrow -electron system. This system has 9 possible FCI states (let's enumerate them 1 thru 9). This system has the following connectivity given in figures 3 and 4.

The connectivity of these CI states is purely combinatorics, i.e. independent of the molecular structure. The sparsity¹⁰ of the CI matrix here is due to the neglecting of determinant-differences greater than 2. If differences of 3 or higher are considered then the Slater-Condon CI connectivity is just the complete graph matrix.

6.2 Excitation/Deexcitation Operators

The creation/annihilation operators are obtained directly from the difference matrix, let's denote it by symbol: \mathbb{B} . Suppose we suppose a particular matrix-element with indices: \mathcal{I} and \mathcal{J} . Then the pair-difference, δ , is just an array with "-1", "0", and "1" elements. The "-1" element indicates a loss of an electron from a given orbital, i.e. an annihilation, while "1" indicates a creation of an electron into a given orbital (and "0" indicates no change):

$$\mathbb{B}^{\mathcal{K}\sigma p} = B^{\mathcal{I}\sigma p} - B^{\mathcal{J}\sigma p}$$

where $\mathbb{B}^{\mathcal{K}\sigma p} = +1$ creation operator, rest matrix p elements set to 0,

where $\mathbb{B}^{\mathcal{K}\sigma p} = -1$ annihilation operator, rest matrix p elements set to 0,

$$\mathbb{B}_{\mathcal{K}\sigma p}^{\dagger} \longrightarrow \{a_{\mathcal{K}\sigma p}, a_{\mathcal{K}\sigma p}^{\dagger}\}$$

¹⁰This is compounded/*convoled* by any sparsity of the molecule specific MO/SO Hamiltonian arrays.

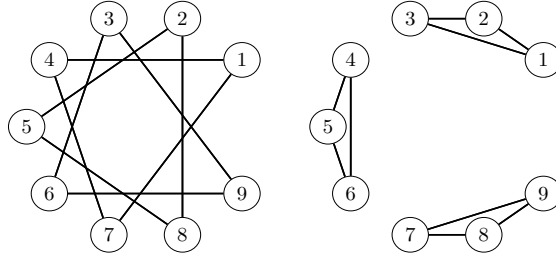


Figure 3: Slater-Condon Rules for 1-difference, in \uparrow and \downarrow spaces respectively, for the example.

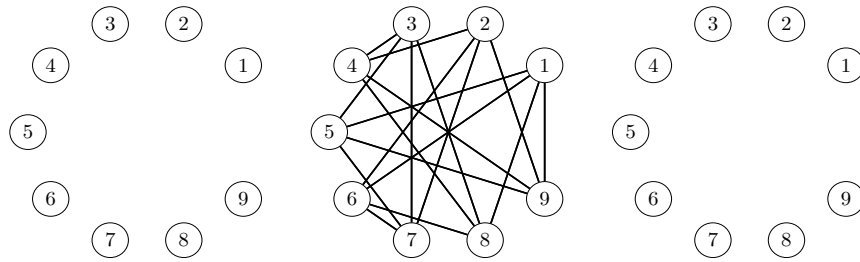


Figure 4: Slater-Condon Rules for 2-difference, in $\uparrow\uparrow$, $\uparrow\downarrow$, and $\downarrow\downarrow$ spaces respectively, for the example.

6.2.1 single excitations

For single excitations, the process is easy enough

6.2.2 double excitations

For double excitations, we have to consider sign changes due to the structure of the double excitation. This sign is computed as in §2.7.

6.2.3 Orbitals in Common

For the first Slater-Condon rule we require the orbitals which are initially occupied and do not change under excitation/deexcitation, these are named: $c_{\uparrow r}$ and $c_{\downarrow r}$. These are found by adding the two Binary representations:

$$c^{\mathcal{K}\sigma p} = \left\lfloor \frac{B^{\mathcal{I}\sigma p} + B^{\mathcal{J}\sigma p}}{2} \right\rfloor.$$

With the result of the division is projected into the integers, i.e. the floor function.

6.3 CI Matrix Construction

In UCI, should have 12 different excitation/deexcitation-operators¹¹ and 2 ‘common’ orbital-operators (for the 1st Slater-Condon rule), with the SO-matrix elements $(pq||rs)^{\sigma\tau}$, and along with the occupation

¹¹For Slater-Condon Rule 1, there are 4: $\begin{pmatrix} \uparrow \\ \downarrow \end{pmatrix} \times \begin{pmatrix} \text{excite} \\ \text{deexcite} \end{pmatrix}$.
While for Slater-Condon Rule 2, there are 8: $\begin{pmatrix} \uparrow\uparrow \\ \uparrow\downarrow \\ \downarrow\uparrow \\ \downarrow\downarrow \end{pmatrix} \times \begin{pmatrix} \text{excite} \\ \text{deexcite} \end{pmatrix}$.

$B_{I\sigma p}$ we may construct the explicit CI matrix construction may be implemented as:

$$\begin{aligned}
H &\stackrel{\dagger}{=} 0_{IJ} \\
H[I, I] &= (p|p)^\sigma B_{I\sigma p} + \frac{1}{2} (pp|qq)^{\sigma\tau} B^{I\sigma p} B^{I\tau q} \\
H[I^\uparrow, J^\uparrow] &= (p|q)^\uparrow a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} + (pq||rr)^\uparrow\uparrow a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} c_{\uparrow r}^\mathcal{K} + (pq|rr)^\uparrow\downarrow a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} c_{\downarrow r}^\mathcal{K} \\
H[I^\downarrow, J^\downarrow] &= (p|q)^\downarrow a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} + (pq||rr)^\downarrow\downarrow a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} c_{\downarrow r}^\mathcal{K} + (pq|rr)^\downarrow\uparrow a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} c_{\uparrow r}^\mathcal{K} \\
H[I^\uparrow\uparrow, J^\uparrow\uparrow] &= (pq||rs)^\uparrow\uparrow a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} a_{\uparrow r}^\mathcal{K} a_{\uparrow s}^{\dagger\mathcal{K}} \\
H[I^\uparrow\downarrow, J^\uparrow\downarrow] &= (pq|rs)^\uparrow\downarrow a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} a_{\downarrow r}^\mathcal{K} a_{\downarrow s}^{\dagger\mathcal{K}} \\
H[I^\downarrow\downarrow, J^\downarrow\downarrow] &= (pq||rs)^\downarrow\downarrow a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} a_{\downarrow r}^\mathcal{K} a_{\downarrow s}^{\dagger\mathcal{K}} \quad ,
\end{aligned}$$

note repeated indices are summed over¹². $H[I, J]$ indices fill the H matrix with indices I and J with the right-hand-side of the equation. Or succinctly put as:

$$\begin{aligned}
H &\stackrel{\dagger}{=} 0_{IJ} \\
H[I, I] &= (p|p)^\sigma B_{I\sigma p} + \frac{1}{2} (pp|qq)^{\sigma\tau} B^{I\sigma p} B^{I\tau q} \\
H[I^\sigma, J^\sigma] &= (p|q)^\sigma a_{\sigma p}^\mathcal{K} a_{\sigma q}^{\dagger\mathcal{K}} + (pq||rr)^{\sigma\tau} a_{\sigma p}^\mathcal{K} a_{\sigma q}^{\dagger\mathcal{K}} c_{\tau r}^\mathcal{K} \\
H[I^{\sigma\tau}, J^{\sigma\tau}] &= (pq||rs)^{\sigma\tau} a_{\sigma p}^\mathcal{K} a_{\sigma q}^{\dagger\mathcal{K}} a_{\tau r}^\mathcal{K} a_{\tau s}^{\dagger\mathcal{K}} \quad ,
\end{aligned}$$

this is possible by combining the excite/deexcite-operators.

7 Solving the CI Equation

7.1 direct diagonalization

After construction of the CI Hamiltonian Matrix (in the CI basis), a simple diagonalization is performed to get the eigenspectrum:

$$H_{IJ} X_{JK} = \mathcal{E}_K X_{IK}$$

where X_{JK} is the eigenvalue matrix and \mathcal{E}_K are the eigenvectors. If the list of determinants include the Hartree-Fock determinant, then this is the solution to the fully correlated ground state.

7.2 Krylov-subspace method

Often instead we seek the ground-state or low-lying excited states. In this case we may use iterative-diagonalization solvers on the Krylov-subspace, such as developed by [Davidson, 1975]. The python library [SciPy *et al.*, 2020] has an implementation of this in its `scipy.sparse.linalg.LinearOperator` module. The SciPy function solely asks for how to compute the matrix-vector product, *e.g.* $\Psi' = H\Psi$, this may be done without explicitly constructing H with all its zeros. So H_{CI} is saved as a dense diagonal vector (the diagonal part of the Hamiltonian above) and a sparse-matrix (given by the Slater-Condon rules, defined above). The resulting matrix-vector product is then (essentially a sparse-matrix-dense-vector product):

$$\begin{aligned}
\Psi'[I] &= D^{(0)}\Psi[I] \\
\Psi'[J^{\text{SC}}] &= D^{\text{SC}}\Psi[J^{\text{SC}}] \\
\Psi'[I] &= \Psi'[I] + \Psi'[J^{\text{SC}}] \quad .
\end{aligned}$$

One of the indices of the SC-rules selects the trial-wave-function's indices and these are multiplied by the sparse-matrix's data entry only to be segmented summed over the over index. Above $\Psi'[J^{\text{SC}}]$ sums over like indices, the segment-sum, this is also explicitly implemented by popular python libraries, *e.g.* in JAX, [Bradbury *et al.*, 2018], the desired function is `jax.ops.segment_sum`.

¹²For instance $(p|p)^\sigma B_{I\sigma p} + \frac{1}{2} (pp|qq)^{\sigma\tau} B^{I\sigma p} B^{I\tau q}$ is actually:
 $\sum_{p\sigma} (p|p)^\sigma B_{I\sigma p} + \frac{1}{2} \sum_{pq\sigma\tau} (pp|qq)^{\sigma\tau} B^{I\sigma p} B^{I\tau q}$

8 Optimization Considerations

As mentioned the number of CI-states grows exponentially, therefore the number of CI-states is potentially much larger than the number of molecular/spin-orbitals, or even the number of atom coordinates. Essentially the CI algorithm amounts to the diagonalization of this exponentially large many-body chemical Hamiltonian. There are two ways to remedy this: exploit sparsity/symmetry and reduce the wavefunction’s search-space (Hilbert Space, the active-space). In practice both should be used. Because it is sparse, due to the sparsity of the SO integrals (defined above) and the Slater-Condon rules, we may save these entries in a convenient format: as a sparse-matrix (for off-diagonal parts) and a dense-vector (for diagonal parts). The algorithm above only requires $B_{I\sigma p}$ (which scales exponentially due to I CI-state index), and the ladder operators $\{a, a^\dagger\}$ to compute the H_{CI} matrix data entries, and hence are only transiently needed. Ultimately, the workflow begins with selecting the theory (active-space and reference(s) states) and the Hartree-Fock solution. Then using these to compute intermediaries as needed to yield the sparse-matrix $\{I^K, J^K, d_K\}$ (K goes over all SC values, not a CI-state index) and dense-vector D^I :

$$\begin{array}{l} \text{Rule (e.g. AS \& reference)} \\ \text{Hartree-Fock } C_p^{\sigma\alpha} \end{array} \rightarrow \begin{array}{l} B_{I\sigma p} \\ (p|q)^{\sigma} \\ (pq||rs)^{\sigma\tau} \\ \{a, a^\dagger\} \end{array} \rightarrow \begin{array}{l} D^I \\ I^K \\ J^K \\ d^K \end{array} .$$

8.1 Slater-Condon Searching

In particular our current implementation of the FCI algorithm, requires an element-wise search which scales N^2 (N being the number of CI states), because it must search every determinant pair to discover its connectivity. If the set of determinants can be generated systematically instead of searched, then the connectivity is automatically known. This algorithm scales pseudolinearly $\mathcal{O} \sim N^1$. Note this still does not help the $\mathcal{O} \sim N^3$ diagonalization costs, with exponentiate with exponential $N \rightarrow \exp(N)$.

8.2 CI Hamiltonian Diagonalization

As mentioned many times, the exponential scaling of the FCI algorithm implies the FCI matrix is very large, and hence iterative diagonalization methods prove useful here that can obtain the n th smallest or largest eigenstates. These iterative algorithms may be used in conjunction with Sparse-Matrix Vector Multiplication, to prevent the explicit construction of the CI Hamiltonian.

9 Comparison and Examples

As we are limited because of the exponentiation of unrestricted-FCI, and our *inefficient* implementation, we consider small examples (as is usually the case, even in *efficient* implementations).

9.1 Sparsity of the CI Hamiltonian Matrix

The sparsity of the CI Hamiltonian Matrix is demonstrated for a few sample molecules in figure 5. The larger the CI Hamiltonian Matrix is the greater the sparsity, also many nonzero elements, although not exactly zero are approximately-zero. These approximately-zero elements are due to the sparsity of the SO-matrices (from the electron-repulsion). We notice however, that in all cases, the matrices are diagonally dominant, indicative that Slater-Condon Rule 0 is usually the dominant source of nonzero elements.

9.2 Potential Energy Surfaces

Here we consider the simplest neutral molecule H_2 /FCI/cc-pvdz to visualize its potential energy surface, shown in fig. 6. The lowest curve representing the ground state, and the rest are enumerated as their energy increases.

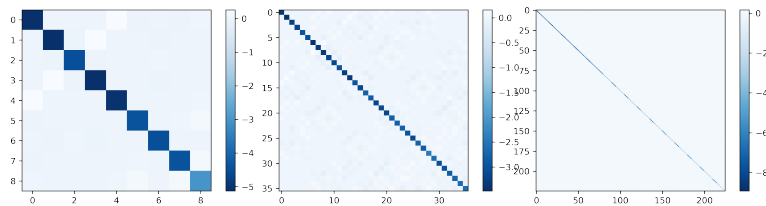


Figure 5: Above are examples of the FCI Hamiltonian for molecules $\text{H}_2\text{He}/\text{FCI}/\text{sto-3g}$, $\text{H}_4/\text{FCI}/\text{sto-3g}$, and $\text{LiH}/\text{FCI}/\text{sto-3g}$ respectively in a minimal basis, all atoms spaced 1.5 \AA from each other.

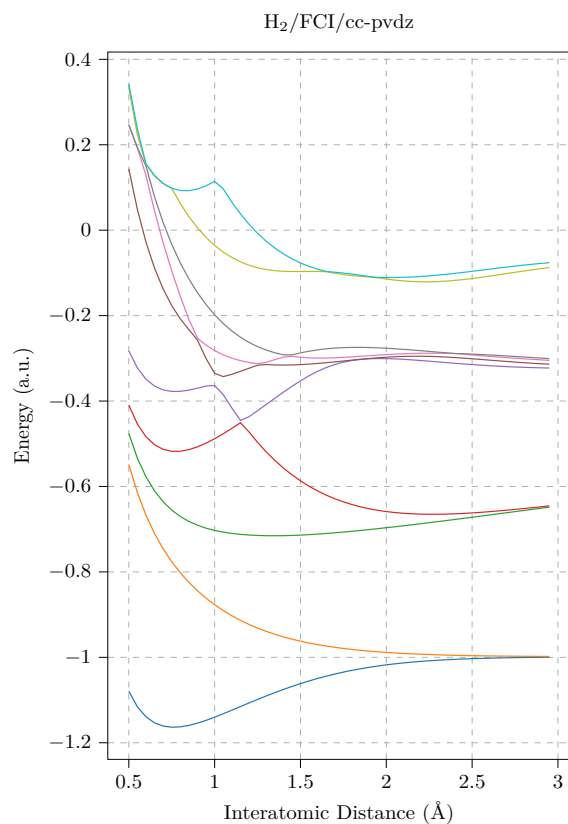


Figure 6: The Potential Energy Surface of $\text{H}_2/\text{FCI}/\text{cc-pvdz}$.

10 Conclusion

The code and description given here, allows for general considerations of CI theory, a variety of patterns of different determinants. Typically truncated CI methods such as CIS, CISD, CISDT, etc... include all singles, doubles or triples, while the discussion here allows to consider only certain excitations.

Ultimately, we are left with two major questions:

- How to generate all nontrivial CI matrix elements without $\mathcal{O} \sim n^2$ scaling? (with $n \equiv$ number-of-CI states, *i.e.* $n \sim \exp(M)$, with M being the number-of-orbitals).
- Is it possible to determine the important CI states (and their connectivity), from the molecular geometry alone (R_{ix} Cartesian-coordinates $x = \{\hat{x}, \hat{y}, \hat{z}\}$ for all nuclei i), atomic-number Z_i , and net charge Q)?

The former question has been so some degree answered by DetCI method introduced by [Handy, 1980] and [Olsen *et al.*, 1988], and commonly used by Physical Chemists to Partition or truncate the FCI wavefunction (*e.g.* Restricted-Active-Spaces, CIS, etc...), Further explanations may be provided by excellent notes: [Sherrill, 1995]. The latter question is more difficult, and has not been addressed as far as the author is concerned (as of 2023).

11 Acknowledgements

The author would like to thank Daniel Nascimento for many interesting discussions, including on this work's topic. The author would also like to thank the Department of Physics at Arizona State University for funding.

A Python Subindexing

The rôle of Slater-determinant MO occupation ($\Lambda^\uparrow, \Lambda^\downarrow$), excitation operators (A^\dagger, A, C , etc...), and indices (I^\uparrow, J^\downarrow , etc...), is to sub-index the 3 important arrays: 1-body SO Hamiltonian, 2-body SO Hamiltonian, and the SO sign-matrix. These sub-indexing arrays must be of data type `integers`. When sub-indexing an array A by another array B , the sub-indexed array inherits of the shape of its sub-indexer, e.g. B sub-indexes A , then $A[B]$ has the same dimension/shape as B . If the sub-indexed array A is sub-indexed by 2 or more, potentially different, arrays, e.g. B, C , then B and C must be of the same shape and dimension. If there is a shape mismatch (dimensional mismatch, e.g. a 2D and 3D array), then one array must be extended to accommodate the larger array. This is done by the outer-product with the ones array (an array of entirely ones), e.g. for extending Λ_{Ii} :

$$A_{Iij} = \Lambda_{Ii} \otimes 1_j.$$

Let's examine the case of:

$$\sum_i (A^\dagger, A \parallel C_i, C_i)^{\uparrow\uparrow},$$

here $A_{\mathbb{I}}$ and $A_{\mathbb{I}}^\dagger$ are 1-dimensional (\mathbb{I} is the 1D CI-matrix partition-index), yet $C_{\mathbb{I}i}$ is 2-dimensional. Therefore, the actual formula is:

$$\left(\sum_i \left((A_{\mathbb{I}}^\dagger \otimes 1_i, A_{\mathbb{I}} \otimes 1_i \parallel C_{\mathbb{I}i}, C_{\mathbb{I}i})^{\uparrow\uparrow} \right)_{\mathbb{I}i} \right)_{\mathbb{I}}.$$

Therefore this term before the sum is also 2-dimensional with indices $\{\mathbb{I}, i\}$, note \dagger and \uparrow are not indices. After the sum, this term is 1-dimensional, given by the outer-most parenthesis, and thus yields this term's contribution to the CI-matrix partition.

A.1 Subindexing/Subscripting

For regular CI-theory, the creation/annihilation operators are arrays with just a single non-zero entry, with value "1" xor "-1", and hence they merely select a certain matrix element from the 1- and 2-body SO Hamiltonian (up to a sign). In particular, for these theories, the tracing with 2-body SO Hamiltonian is very inefficient. Instead, the creation/annihilation can be stored as a single entry indexing the SO-orbital of interest, and be used to sub-index that particular orbital. In this formalism the CI Hamiltonian becomes:

$$\begin{aligned} H &\stackrel{\perp}{=} 0_{IJ} \\ H[I, I] &= (p|p)^\sigma B_{I\sigma p} + \frac{1}{2} (pp|qq)^{\sigma\tau} B^{I\sigma p} B^{I\tau q} \\ H[I^\uparrow, J^\uparrow] &= (p|q)^\uparrow a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} + (pq|rr)^{\uparrow\uparrow} a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} c_{\uparrow r}^\mathcal{K} + (pq|rr)^{\uparrow\downarrow} a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} c_{\downarrow r}^\mathcal{K} \\ H[I^\downarrow, J^\downarrow] &= (p|q)^\downarrow a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} + (pq|rr)^{\downarrow\downarrow} a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} c_{\downarrow r}^\mathcal{K} + (pq|rr)^{\downarrow\uparrow} a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} c_{\uparrow r}^\mathcal{K} \\ H[I^{\uparrow\uparrow}, J^{\uparrow\uparrow}] &= (a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} \parallel a_{\uparrow r}^\mathcal{K} a_{\uparrow s}^{\dagger\mathcal{K}})^{\uparrow\uparrow} \\ H[I^{\uparrow\downarrow}, J^{\uparrow\downarrow}] &= (a_{\uparrow p}^\mathcal{K} a_{\uparrow q}^{\dagger\mathcal{K}} \parallel a_{\downarrow r}^\mathcal{K} a_{\downarrow s}^{\dagger\mathcal{K}})^{\uparrow\downarrow} \\ H[I^{\downarrow\downarrow}, J^{\downarrow\downarrow}] &= (a_{\downarrow p}^\mathcal{K} a_{\downarrow q}^{\dagger\mathcal{K}} \parallel a_{\downarrow r}^\mathcal{K} a_{\downarrow s}^{\dagger\mathcal{K}})^{\downarrow\downarrow} \end{aligned}$$

Therefore if all excitation/deexcitation-operators had merely entries binary (0 xor 1), they provide a mask for obtaining the needed entry at $\mathcal{O} \sim 1$. These entries, however need to be supplemented by the sign as given in §2.7.

A.1.1 sparse-partial-tracing

Building on this, sparse-partial-tracing, instead of the dense-partial-tracing, can greatly speed up and save on memory for many of these very sparse methods.

B Raw Code

B.1 useful definitions

B.1.1 Determinant Manipulation

```
def givenAgetB(AA, AB, N_mo):
    "Given A(i occupied orbitals for each determinant) get B (binary rep.)"

    Binary = np.zeros((AA.shape[0], 2, N_mo), dtype=np.int8)
    for I in range(len(Binary)):
        Binary[I, 0, AA[I,:]] = 1
        Binary[I, 1, AB[I,:]] = 1

    return Binary

def SpinOuterProduct(A, B, stack=False):
    AA = np.einsum("Ii, J -> IJi", A, np.ones(B.shape[0], dtype=np.int8)).reshape(
        (A.shape[0]*B.shape[0], A.shape[1]))
    AB = np.einsum("Ii, J -> JIi", B, np.ones(A.shape[0], dtype=np.int8)).reshape(
        (A.shape[0]*B.shape[0], B.shape[1]))

    if stack:
        return np.array([AA,AB])
    else:
        return AA, AB

def get_fci_combos(mf):
    O_sp = np.asarray(mf.mo_occ, dtype=np.int8)
    N_s = np.einsum("sp -> s", O_sp)
    N = O_sp.shape[1]

    A_alpha = np.asarray( list(combinations( np.arange(0, N, 1, dtype=np.int8) , N_s[0] ) ) )
    A_beta = np.asarray( list(combinations( np.arange(0, N, 1, dtype=np.int8) , N_s[1] ) ) )
    AA, AB = SpinOuterProduct(A_alpha, A_beta)
    Binary = givenAgetB(AA, AB, N)

    return Binary
```

B.1.2 Slater-Condon Rules

```
def determinantsign(Binary):
    sign = np.cumsum( Binary, axis=2)
    for I in range(len(Binary)):
        iia = np.where( Binary[I,0] == 1)[0]
        iib = np.where( Binary[I,1] == 1)[0]
        sign[I, 0, iia] = np.arange(0, len(iia), 1)
        sign[I, 1, iib] = np.arange(0, len(iib), 1)

    return ( (-1)**(sign) ).astype(np.int8)

def get_excitation_op(i, j, binary, sign, spin=0):
    Difference = binary[i,spin] - binary[j, spin]
    a_t = (Difference + 0.5).astype(np.int8)
    a = -1*(Difference - 0.5).astype(np.int8)
    if np.sum(a[0]) > 1: ### this is a double excitation
        a_t = 1*a_t ## make copy
        a_t[ np.arange(len(a_t)),(a_t!=0).argmax(axis=1) ] = 0 ## zero first 1
        a_t = np.abs(a_t - a) ## absolute difference from original
        a_t = np.asarray([sign[j, spin]*a_t,sign[j, spin]*a_t]) ## stack

        a = 1*a ## make copy
        a[ np.arange(len(a)),(a!=0).argmax(axis=1) ] = 0 ## zero first 1
        a = np.abs(a - a) ## absolute difference from original
        a = np.asarray([sign[i, spin]*a,sign[i, spin]*a]) ## stack

    return a_t, a

def SlaterCondon(Binary):
    sign = determinantsign(Binary)
    SpinDifference = np.sum( np.abs(Binary[:, None, :, :] - Binary[None, :, :, :]), axis=3)//2

    ## indices for 1-difference
    I_A, J_A = np.where( np.all(SpinDifference==np.array([1,0], dtype=np.int8), axis=2) )
    I_B, J_B = np.where( np.all(SpinDifference==np.array([0,1], dtype=np.int8), axis=2) )
    ## indices for 2-differences
    I_AA, J_AA = np.where( np.all(SpinDifference==np.array([2,0], dtype=np.int8), axis=2) )
    I_BB, J_BB = np.where( np.all(SpinDifference==np.array([0,2], dtype=np.int8), axis=2) )
    I_AB, J_AB = np.where( np.all(SpinDifference==np.array([1,1], dtype=np.int8), axis=2) )

    ### get excitation operators
    a_t , a = get_excitation_op(I_A , J_A , Binary, sign, spin=0)
    b_t , b = get_excitation_op(I_B , J_B , Binary, sign, spin=1)
    ca = ((Binary[I_A,0,:] + Binary[J_A,0,:])/2).astype(np.int8)
    cb = ((Binary[I_B,1,:] + Binary[J_B,1,:])/2).astype(np.int8)

    aa_t, aa = get_excitation_op(I_AA, J_AA, Binary, sign, spin=0)
    bb_t, bb = get_excitation_op(I_BB, J_BB, Binary, sign, spin=1)
    ab_t, ab = get_excitation_op(I_AB, J_AB, Binary, sign, spin=0)
    ba_t, ba = get_excitation_op(I_AB, J_AB, Binary, sign, spin=1)

    SC1 = [I_A, J_A, a_t , a, I_B, J_B, b_t , b, ca, cb]
    SC2 = [I_AA, J_AA, aa_t, aa, I_BB, J_BB, bb_t, bb, I_AB, J_AB, ab_t, ab, ba_t, ba]

    return SC1, SC2
```

B.1.3 SO Dressing

```
def get_SO_matrix(uhf_pyscf, SF=False, H1=None, H2=None):
    """ Given a PySCF uhf object get SO Matrices """

    Ca, Cb = (uhf_pyscf).mo_coeff
    S = (uhf_pyscf.mol).intor("int1e_ovlp")
    eig, v = np.linalg.eigh(S)
    A = (v) @ np.diag(eig**(-0.5)) @ np.linalg.inv(v)
    H = uhf_pyscf.get_hcore()

    n = Ca.shape[1]
    eri_aa = (ao2mo.general( (uhf_pyscf)._eri , (Ca, Ca, Ca, Ca),
        compact=False)).reshape((n,n,n,n), order="C")
    eri_aa -= eri_aa.swapaxes(1,3)
    eri_bb = (ao2mo.general( (uhf_pyscf)._eri , (Cb, Cb, Cb, Cb),
        compact=False)).reshape((n,n,n,n), order="C")
    eri_bb -= eri_bb.swapaxes(1,3)
    eri_ab = (ao2mo.general( (uhf_pyscf)._eri , (Ca, Ca, Cb, Cb),
        compact=False)).reshape((n,n,n,n), order="C")
    #eri_ba = (1.*eri_ab).swapaxes(0,3).swapaxes(1,2) ## !! caution depends on symmetry
    eri_ba = (ao2mo.general( (uhf_pyscf)._eri , (Cb, Cb, Ca, Ca),
        compact=False)).reshape((n,n,n,n), order="C")
    H2 = np.stack(( np.stack((eri_aa, eri_ab)), np.stack((eri_ba, eri_bb)) ))

    H1 = np.asarray([np.einsum("AB, Ap, Bq -> pq", H, Ca, Ca), np.einsum("AB, Ap, Bq -> pq",
        H, Cb, Cb)])

    if SF:
        eri_abab = (ao2mo.general( (uhf_pyscf)._eri , (Ca, Cb, Ca, Cb),
            compact=False)).reshape((n,n,n,n), order="C")
        eri_abba = (ao2mo.general( (uhf_pyscf)._eri , (Ca, Cb, Cb, Ca),
            compact=False)).reshape((n,n,n,n), order="C")
        eri_baab = (ao2mo.general( (uhf_pyscf)._eri , (Cb, Ca, Ca, Cb),
            compact=False)).reshape((n,n,n,n), order="C")
        eri_baba = (ao2mo.general( (uhf_pyscf)._eri , (Cb, Ca, Cb, Ca),
            compact=False)).reshape((n,n,n,n), order="C")
        H2_SF = np.stack(( np.stack((eri_abab, eri_abba)), np.stack((eri_baab, eri_baba)) ))
        return H1, H2, H2_SF

    else:
        return H1, H2
```

B.1.4 CI Dressing

```
def CI_H(H1, H2, SC1, SC2):
    """
    Explicitly construct the CI Hamiltonian Matrix
    GIVEN: H1 (1-body Hamiltonian)
           H2 (2-body Hamiltonian)
           SC1 (1-body Slater-Condon Rules)
           SC2 (2-body Slater-Condon Rules)
    GET:   CI Hamiltonian
    """

    I_A, J_A, a_t , a, I_B, J_B, b_t , b, ca, cb = SC1
    I_AA, J_AA, aa_t, aa, I_BB, J_BB, bb_t, bb, I_AB, J_AB, ab_t, ab, ba_t, ba = SC2

    H_CI = np.einsum("Spp, ISp -> I", H1, Binary, optimize=True)
    H_CI += np.einsum("STppqq, ISp, ITq -> I", H2, Binary, Binary, optimize=True)/2
    H_CI = np.diag(H_CI)

    ## Rule 1
    H_CI[I_A , J_A ] -= np.einsum("pq, Kp, Kq -> K", H1[0], a_t, a, optimize=True)
    H_CI[I_A , J_A ] -= np.einsum("pqrr, Kp, Kq, Kr -> K", H2[0,0], a_t, a, ca, optimize=True)
    H_CI[I_A , J_A ] -= np.einsum("pqrr, Kp, Kq, Kr -> K", H2[0,1], a_t, a, Binary[I_A,1],
        optimize=True)

    H_CI[I_B , J_B ] -= np.einsum("pq, Kp, Kq -> K", H1[1], b_t, b, optimize=True)
    H_CI[I_B , J_B ] -= np.einsum("pqrr, Kp, Kq, Kr -> K", H2[1,1], b_t, b, cb, optimize=True)
    H_CI[I_B , J_B ] -= np.einsum("pqrr, Kp, Kq, Kr -> K", H2[1,0], b_t, b, Binary[I_B,0],
        optimize=True)

    ## Rule 2
    H_CI[I_AA, J_AA] = np.einsum("pqrs, Kp, Kq, Kr, Ks -> K", H2[0,0], aa_t[0], aa[0],
        aa_t[1], aa[1], optimize=True)
    H_CI[I_BB, J_BB] = np.einsum("pqrs, Kp, Kq, Kr, Ks -> K", H2[1,1], bb_t[0], bb[0],
        bb_t[1], bb[1], optimize=True)
    H_CI[I_AB, J_AB] = np.einsum("pqrs, Kp, Kq, Kr, Ks -> K", H2[0,1], ab_t, ab, ba_t, ba,
        optimize=True)

    return H_CI
```

B.2 fci.py

```
def fci(mf):
    """ Calculate the FCI of a PySCF Mean Field Object
    GIVEN: mf (PySCF Mean Field Object)
    GET:   E (Eigenvalues), X (Eigenstates) """

    Binary = get_fci_combos(mf)
    H1, H2 = get_SO_matrix(mf)
    SC1, SC2 = SlaterCondon(Binary)
    H_CI = CI_H(H1, H2, SC1, SC2)
    E, X = np.linalg.eigh(H_CI)

    return E, X
```

C Mean Field Theory

Unlike the CI theory, which is a linear theory, Mean-Field (MF) Theories are necessarily nonlinear. In particular their Hamiltonian depends on their eigenstates. To this end, these equations can be solved by a Self-Consistent Field (SCF) algorithm. In theory, MF approach to “electronic structure” attempts to solve the equilibrium electrostatic distribution, In practice, the MF equations also orthogonalize the atomic-centric Basis-Set-Ansatz.

C.1 AO & MO Basis

Solving the MF equations even for a single particle in a inhomogeneous setting (a single electron on some ionic background) is not analytically feasible. Instead [Roothaan, 1951] introduced the insight of using Linear Combination of Atomic Orbitals (LCAO). Atomic orbitals are spherically symmetric and analytic functions, Roothaan’s insight is that they may be added together to capture the inhomogeneous aspects of molecular-orbitals (lack of spherical or rotational symmetry). This may done by a rotation matrix A_p^α :

$$\psi_p^{\text{LCAO}} = A_p^\alpha \phi_\alpha$$

such that (orthonormality holds):

$$\psi_p^\dagger \psi_q = \delta_{pq} \quad .$$

The question is the functional form of A_p^α . Let’s begin by:

$$\begin{aligned} \delta_{pq} &= \psi_p^\dagger \psi_q \\ &= A_p^\alpha \underbrace{\phi_\alpha \phi_\beta}_{S_{\alpha\beta}} A_q^\beta \\ A_p^\alpha S_{\alpha\beta} A_q^\beta &= S_{pq} = \delta_{pq} \\ A_p^\alpha &= \left(S_{\alpha\beta}^{-1/2} \right)_p^\alpha \quad . \end{aligned}$$

This requires the matrix-inverse-square-root of the overlap-matrix.

C.2 Hartree-Fock

Hartree-Fock Theory developed by [Hartree, 1928], [Fock, 1930], and [Slater, 1929] is the basis of much of physical chemistry. It succeeds in obtaining most of the binding energy between electrons and nuclei. Being a mean-field-theory it cannot account for the total binding energy, and the energy deficit from the true binding energy (full-CI) is defined as the *correlation energy*. Post-Hartree-Fock methods attempt to correct for this correlation energy in a variety of methods. Most post-Hartree-Fock methods start with a Hartree-Fock calculation, as a spring-board, and so this brief introduction serves to give an overview of this method. It even inspired Kohn-Sham (KS) theory, using Density Functional Theory (DFT) to compensate for some lost correlation energy. In our implementation, the electrons exist in a predefined finite basis, consisting of linear combinations of Gaussian orbitals (centered on classical nuclei). Because of the creation of the 2-electron integrals, the HF method is naïvely a $\mathcal{O}(N^4)$ method (N being the number of basis functions). If methods such as density fitting are used this can be reduced to $\mathcal{O}(N^3)$, and if large enough samples are considered this can be further reduced to $\mathcal{O}(N^2)$ (although with a large $\mathcal{O}(N^4)$ prefactor), [Helgaker *et al.*, 2014].

The Hartree-Fock (HF) approximation to electronic-distribution is implemented in 3 broadly different forms: Restricted¹³, Unrestricted¹⁴, and Generalized form. They all map an atomic basis into an orthonormal molecular basis. They have practical implementations using the Basis-Set-Ansatz, i.e. LCAO, and in molecular systems the Gaussian-Basis-Set as promoted by many, i.e. Pople and Boys *et al.* HF is perhaps the most well developed electronic structure/distribution method, having many modifications to achieve better results. The computationally cheapest method of proceeding, is working with the AO basis form, which we shall describe below.

¹³Introduced by [Roothaan, 1951].

¹⁴Introduced by [Pople and Nesbet, 1954].

C.2.1 Objective

Our objective would be to obtain the aforementioned shielding coefficients $C_{\alpha}^{\sigma p}$ for an arbitrary molecular system. These Mean-field Orbital¹⁵ (MO) Coefficients rotate the atomic centric orbitals into the correct ones with appropriate mean-field electronic shielding (they introduce an intrinsic length scale). In practice they are applied by:

$$\psi_{p\sigma}^{\text{HF}} = C_{p\sigma}^{\alpha} \phi_{\alpha} \quad .$$

The α -index describes the AO basis-function, ϕ_{α} , while the σ index describes spin, and the p index describes the MO. The MO coefficients may be obtained by the following equation, using the overlap matrix $S_{\alpha\gamma}$ and the Fock Matrix $F_{\alpha\gamma}$ (a function of $C_{\alpha\gamma}^{\sigma}$, introduced in §C.2.2):

$$F_{\alpha\gamma}(C^{\sigma})C_{\gamma\beta}^{\sigma} = \mathcal{E}_{\alpha}^{\sigma} S_{\alpha\gamma} C_{\gamma\beta}^{\sigma} \quad .$$

Because of their non-linearity we can not solve them directly, and instead we plan to solve a *linearized* version (but iteratively until convergence):

$$F_{\alpha\gamma}^{\sigma} C_{\gamma\delta}^{\sigma} = \mathcal{E}_{\alpha}^{\sigma} S_{\alpha\gamma} C_{\gamma\delta}^{\sigma} \quad .$$

C.2.2 SCF Setup

The construction of the Fock matrix is given by:

$$(\alpha|\beta)_{\text{Fock}}^{\sigma} = \underbrace{(\alpha|\beta)_{\text{Kinetic}}^{\sigma} + (\alpha|\beta)_{\text{Nuclei/ECP}}^{\sigma}}_{\text{core}} + (\alpha\beta||\gamma\delta)^{\sigma\tau} D_{\gamma\delta}^{\tau} \quad . \quad (3)$$

It is the effective 1-body Hamiltonian, consisting of a “core” part and contracted 2-body part. The elements of this matrix are the kinetic, nuclei-interaction, AO electron density matrix, and the 2-body Hamiltonian (commonly consisting of just the Electron Repulsion Integral (ERI) term). These terms other than the electron density, are obtained by Gaussian-Basis-Function integrators, which compute the integral between these two basis functions. This is a tedious task which may be outsourced to [PySCF *et al.*, 2018] or [Psi4NumPy *et al.*, 2018].

While the AO density-matrix may be constructed knowing the MO orbital occupation, given by the MO density-matrix, and the MO coefficient shielding. We begin by discussing the MO Density Matrix for the HF-state, this matrix is diagonal with the MO occupations on the diagonal, i.e. $D_{pq}^{\sigma} = B^{\text{HF}\sigma p} \delta_{pq}$. Then the AO-Density Matrix is given as:

$$D_{\alpha\beta}^{\sigma} = C_{\alpha}^{\sigma p} D_{pq}^{\sigma} C_{\beta}^{\sigma q} \quad . \quad (4)$$

Next the electronic binding-energy may be calculated from the equation below (using these terms):

$$\mathcal{E} = \frac{1}{2} D_{\alpha\beta}^{\sigma} ((\alpha|\beta)_{\text{core}}^{\sigma} + F_{\alpha\beta}^{\sigma}) + \mathcal{E}_{\text{nuclei-nuclei}} \quad . \quad (5)$$

A more thorough review of the Roothaan/Pople-Nesbet solution to the Hartree-Fock problem may be found in [Szabo and Ostlund, 2012b] (see chapter 3).

C.2.3 SCF procedure

Here we introduce the *core-guess* SCF procedure, it begins with diagonalizing the core-Hamiltonian. This obtains eigenstates which must be orthogonalized:

$$(\alpha|\beta)_{\text{core}}^{\sigma} \longrightarrow C_{\alpha\beta}^{\sigma} \xrightarrow{A_{\alpha}^p} C_{\alpha}^{\sigma p}$$

Next, the AO-density matrix is constructed (with the current $C_{\alpha}^{\sigma p}$ and eq. 4) along with the Fock matrix. The diagonalization of the AO-Fock Matrix yields new eigenvectors which must be orthogonalized.

$$C_{\alpha}^{\sigma p} \xrightarrow{\text{eq. 4}} D_{\alpha\beta}^{\sigma} \xrightarrow{\text{eq. 3}} (\alpha|\beta)_{\text{Fock}}^{\sigma} \longrightarrow C_{\alpha\beta}^{\sigma p} \xrightarrow{A_{\alpha}^p} C_{\alpha}^{\sigma p}$$

Then the similarity of the MO coefficients is determined:

$$C_{\alpha}^{\sigma p} \sim C_{\alpha}^{\sigma p}$$

and if they are sufficiently similar the algorithm stops and this is the desired answer, else it continues.

¹⁵alt. Molecular Orbital.

C.3 Adiabatic Kohn-Sham

A generalized version of Hartree and Hartree-Fock theory is the now famous Kohn-Sham (KS) theory. Kohn-Sham theory is so popular that it is often used as synonymous to DFT (this is not the case, as there is interesting research on Orbital-Free versions). Unlike the earlier, Thomas-Fermi theory, which approximates the kinetic-energy by a density-functional, KS theory instead approximates electron-electron interactions by a density-functional and treats the kinetic-energy *exactly*, within this theory. Following Hartree, and Hartree-Fock's theory, the novel feature is the inclusion of the $(\alpha|\beta)_{\text{XC}}^\sigma$ operator which would compute the effective electron-electron interaction, via a density-functional including exchange and correlation. Like, HF theory, this operator is diagonalized and reconstructed until convergence is reached. For a more detailed discussion the reader is referred to [Ullrich, 2011] (chapter 2, reviews static KS-DFT).

We begin by supposing there is an exact functional of the density for the energy of an electronic system given by $\mathcal{E}_{\text{XC}}[\rho]$. This is nice, but not directly useful for our purposes, instead we may define a potential created by this functional, obtained by its variation-with-respect-to-the-density:

$$(\alpha|\beta)_{\text{XC}}^\sigma = \frac{\delta \mathcal{E}[\rho]}{\delta D_{\alpha\beta}^\sigma} = ((\alpha|\beta))^\sigma.$$

This is the aforementioned novel feature. In the adiabatic approximation, this density is independent of time $\dot{D}_{\alpha\beta}^\sigma = 0$, and is the instantaneous ground state density. It this is matrix which includes an additional contribution to the Fock matrix, without explicit Hartree-Fock exchange (as exchange is included in the XC-functional) to build the Kohn-Sham matrix (informally also known as a Fock matrix):

$$F_{\alpha\beta}^\sigma = (\alpha|\beta)_{\text{Fock}}^\sigma = (\alpha|\beta)_{\text{Kinetic}}^\sigma + (\alpha|\beta)_{\text{Nuclei/ECP}}^\sigma + (\alpha|\beta)_{\text{XC}}^\sigma + (\alpha|\beta|\gamma\delta)^{\sigma\tau} D_{\gamma\delta}^\tau.$$

This matrix is solved SCF, much in the same way as the HF-equations.

C.3.1 Response-KS

For LR-KS (a.k.a. TDDFT) or other response theories, further derivatives of the XC energy-functional, $\mathcal{E}[\rho]$, must be known:

$$\begin{aligned} ((\alpha|\beta))^\sigma &= \frac{\delta \mathcal{E}[\rho]}{\delta D_{\alpha\beta}^\sigma}, \\ (\alpha|\beta|\gamma\delta)_{\text{XC}}^{\sigma\tau} &= ((\alpha|\beta|\gamma\delta))^{\sigma\tau} = \frac{\delta^2 \mathcal{E}[\rho]}{\delta D_{\alpha\beta}^\sigma \delta D_{\gamma\delta}^\tau}, \\ (((\alpha|\beta|\gamma|\delta\varepsilon\eta)))^{\sigma\tau\nu} &= \frac{\delta^3 \mathcal{E}[\rho]}{\delta D_{\alpha\beta}^\sigma \delta D_{\gamma\delta}^\tau \delta D_{\varepsilon\eta}^\nu}. \end{aligned}$$

C.4 Multicomponent

A straightforward generalization of the HF method is the Multicomponent Unrestricted Hartree-Fock (muHF or μHF or μSCF). In this method we consider two or more different kinds of Fermions, e.g. protons and electrons, in a single SCF procedure. This method was introduced as Nuclear Electron Orbital (NEO) method and extensively developed by the Hammes-Schiffer group starting with [Webb *et al.*, 2002]. Because this method makes no restriction to the kind of Fermion, e.g. nuclei xor positrons, I prefer the Multicomponent SCF name, e.g. μHF if correlation is entirely absent. One important consideration is the basis set. Electronic structure theory has developed many basis sets for electrons in a variety of conditions (many of these are stored on the Basis Set Exchange, see [Pritchard *et al.*, 2019]). However, other Fermions have not gotten the same treatment.

Naïvely, the exponential-value of the basis-set parameters controls the scale of the basis, changing these introduces a new length-scale. This length-scale is controlled by the mass of the Fermion. Thus, suppose ζ_0 is the exponential of Gaussian-fitted-Slater functions, then $\zeta = \frac{\zeta_0}{M^2}$ is the exponential factor for a particle of mass M . However, unlike atomistic basis functions which expect a wavefunction solution for the inverse-distance potential of the nuclei (e.g. $V(r) = \frac{1}{4\pi\varepsilon_0\varepsilon} \frac{1}{r}$), it is

well-known that nuclei experience more of a harmonic-potential (parabolic), i.e. corresponding to vibrational frequencies. Because nuclei are well-separated in regular matter, they are always in singly occupied states. Therefore specialized basis functions should be used. In conjunction of introducing the method, Hammes-Schiffer's group introduced protonic basis-functions, see [Yu *et al.*, 2020] and density-functionals [Yang *et al.*, 2017].

References

- [Bradbury *et al.*, 2018] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.
- [Davidson, 1975] Davidson, E. (1975). *Comput. Phys.*, 17:87–94.
- [Fock, 1930] Fock, V. C. (1930). *Z. Phys.*, 61:126.
- [Handy, 1980] Handy, N. C. (1980). *Chem. Phys. Lett.*, 74(2):280–283.
- [Hartree, 1928] Hartree, D. R. (1928). *Math. Proc. Camb. Philos. Soc.*, 24(1):111–132.
- [Helgaker *et al.*, 2014] Helgaker, T., Jørgensen, P., and Olsen, J. (2014). *Molecular Electronic-Structure Theory*. John Wiley & Sons.
- [Olsen *et al.*, 1988] Olsen, J., Roos, B. O., Jørgensen, P., and Jensen, H. J. A. (1988). *J. Chem. Phys.*, 89(4):2185–2192.
- [Pople and Nesbet, 1954] Pople, J. A. and Nesbet, R. K. (1954). *J. Chem. Phys.*, 22(3):571–572.
- [Pritchard *et al.*, 2019] Pritchard, B. P., Altarawy, D., Didier, B., Gibson, T. D., and Windus, T. L. (2019). *Journal of Chemical Information and Modeling*, 59(11):4814–4820.
- [Psi4NumPy *et al.*, 2018] Psi4NumPy, Smith, D. G., Burns, L. A., Sirianni, D. A., Nascimento, D. R., Kumar, A., James, A. M., Schriber, J. B., Zhang, T., Zhang, B., Abbott, A. S., *et al.* (2018). *J. Chem. Theory Comput.*, 14(7):3504–3511.
- [PySCF *et al.*, 2018] PySCF, Sun, Q., Berkelbach, T. C., Blunt, N. S., Booth, G. H., Guo, S., Li, Z., Liu, J., McClain, J. D., Sayfutyarova, E. R., Sharma, S., *et al.* (2018). *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 8(1):e1340.
- [Roothaan, 1951] Roothaan, C. C. J. (1951). *Rev. Mod. Phys.*, 23(2):69.
- [SciPy *et al.*, 2020] SciPy, Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). *Nature Methods*, 17:261–272.
- [Seritan *et al.*, 2021] Seritan, S., Bannwarth, C., Fales, B. S., Hohenstein, E. G., Isborn, C. M., Kokkila-Schumacher, S. I., Li, X., Liu, F., Luehr, N., Snyder Jr, J. W., *et al.* (2021). *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 11(2):e1494.
- [Sherrill, 1995] Sherrill, C. D. (1995). An introduction to configuration interaction theory.
- [Slater, 1929] Slater, J. C. (1929). *Phys. Rev.*, 34(10):1293.
- [Szabo and Ostlund, 2012a] Szabo, A. and Ostlund, N. S. (2012a). *Modern Quantum Chemistry*. Dover Publications.
- [Szabo and Ostlund, 2012b] Szabo, A. and Ostlund, N. S. (2012b). *Modern Quantum Chemistry*. Dover Publications.

- [Ullrich, 2011] Ullrich, C. A. (2011). *Time-dependent density-functional theory: concepts and applications*. Oxford University Press.
- [Webb *et al.*, 2002] Webb, S. P., Iordanov, T., and Hammes-Schiffer, S. (2002). *J. Chem. Phys.*, 117(9):4106–4118.
- [Yang *et al.*, 2017] Yang, Y., Brorsen, K. R., Culpitt, T., Pak, M. V., and Hammes-Schiffer, S. (2017). *J. Chem. Phys.*, 147(11):114113.
- [Yu *et al.*, 2020] Yu, Q., Pavošević, F., and Hammes-Schiffer, S. (2020). *J. Chem. Phys.*, 152(24):244123.

APPENDIX C

JCP PAPER

Dynamics of rare gas solids irradiated by electron beams

Cite as: J. Chem. Phys. 152, 144303 (2020); doi: 10.1063/1.5134801

Submitted: 1 November 2019 • Accepted: 23 March 2020 •

Published Online: 10 April 2020



J. Candanedo,^{1,a)} C. Caleman,^{2,3} N. Timneanu,² O. Beckstein,^{1,4} and J. C. H. Spence¹

AFFILIATIONS

¹Department of Physics, Arizona State University, Tempe, Arizona 85282, USA

²Department of Physics and Astronomy, Uppsala University, Uppsala SE-75120, Sweden

³Center for Free-Electron Laser Science, Deutsches Elektronen-Synchrotron, Notkestraße 85, Hamburg, Germany

⁴Center for Biological Physics, Arizona State University, Tempe, Arizona 85282, USA

Note: This paper is part of the JCP Special Topic on Ultrafast Molecular Sciences by Femtosecond Photons and Electrons.

^{a)}Author to whom correspondence should be addressed: jcandane@asu.edu

ABSTRACT

The remarkable success of x-ray free-electron lasers and their ability to image biological macromolecules while outrunning secondary radiation damage due to photoelectrons, by using femtosecond pulses, raise the question of whether this can be done using pulsed high-energy electron beams. In this paper, we use excited state molecular dynamics simulations, with tabulated potentials, for rare gas solids to investigate the effect of radiation damage due to inelastic scattering (by plasmons, excitons, and heat) on the pair distribution function. We use electron energy loss spectra to characterize the electronic excitations responsible for radiation damage.

Published under license by AIP Publishing. <https://doi.org/10.1063/1.5134801>

I. INTRODUCTION

Radiation damage imposes fundamental limitations on the fidelity and resolution of imaging and diffraction methods. Recently, it has been suggested and demonstrated that damage may be overcome using femtosecond x-ray pulses to out-run the damage processes.¹ Using these pulses, the serial crystallography approach has made it possible to solve the structure of protein micro-crystals, record the femtosecond dynamics of light-sensitive proteins in micro-crystals, and to observe protein dynamics at atomic resolution during enzyme catalysis.² This ability to outrun damage is important because it circumvents the need to immobilize by freezing proteins to avoid damage in protein crystallography and so allows observations of dynamics under near-physiological conditions of temperature and chemical environment.³ The aim of this research is to determine if this can also be done using brief electron beam pulses for the collection of transmission electron diffraction data and, if not, to understand the severity of the damage processes that remain. As a preliminary, in this paper, we study electron-beam radiation damage in the simple Rare Gas Solids (RGSs) in order to avoid the complications of interatomic bonds and their excitations.

Differences between diffraction and damage processes using x rays and electrons have been compared by Henderson⁴ (continuous exposure) and Spence⁵ (pulsed mode). Unlike x rays, electrons are diffracted from both the nuclei (Rutherford scattering) and the electron density. While the photoelectron cross section greatly exceeds that for elastic scattering for x rays, elastic and inelastic scattering cross sections for electrons are comparable. Laser amplification is now available for hard x rays, but not for electron beams, which provide fewer electrons per pulse but have a much larger elastic cross section. The Coulomb repulsion between electrons makes focusing difficult simultaneously at the brief and high currents. Flux is the product of source brightness and emittance, with emittance proportional to the product of beam divergence and source size. For transmission Electron Diffraction (ED) from thin crystals, to avoid overlap of Bragg spots, this beam divergence must be less than the Bragg angle, which for typical photocathode emittance values results in a focused beam many micrometers wide. As a result, Ultrafast Electron Diffraction (UED) experiments are undertaken with very low fluence at the sample per shot, and samples, in the form of a thin slab, should be homogeneous over an area of several micrometers. A typical fluence⁶ (integrated flux) of $\sim 10^{-4} - 10^{-6} \text{ e}/\text{\AA}^2$ (or *dose* as defined in the

Cryo-EM literature) is therefore much less than that used in Cryo-EM. These are the conditions for which our simulations are performed. Since the Cryo-EM dose of about $<10 \text{ e}/\text{\AA}^2$ produces very little damage, our first conclusion is that Coulomb interactions in the beam and the resulting large beam diameters used with picosecond pulses in the UED field will produce negligible amounts of damage. Our simulations are therefore for much longer pulses than used in UED.

The simplest form of radiation damage and energy transfer from the beam to the sample occurs via direct Primary Knock-on Atom (PKA) elastic collisions between beam electrons and the nuclei (see Sec. II A). Other than PKA, radiation damage is caused by inelastic collisions between the sample and beam electrons, causing the sample to flow on a new potential energy surface. This change in nuclear-nuclear correlation may be captured by using a time-resolved Radial Distribution Function (trRDF), which may be computed using Molecular Dynamics (MD) simulations. In this paper, we describe a program that performs the required excited-state MD to simulate RGS irradiation for electrons (also x-ray) beam pulses. The specific aim of this paper is to use this program to compute the trRDF during irradiation by an electron beam pulse.

Concurrently, dynamical (space-time) correlation may be captured by the Van Hove function,⁷ which is directly related, by the imaginary part of its Fourier transform, to the loss or the response function. This may be directly measured using the Electron Energy Loss Spectra (EELS). EELS spectra may be recorded from nanometer-sized regions in an electron microscope simultaneously with the elastic scattering. These spectra may be interpreted in terms of a frequency and momentum dependent dielectric function,⁸

$$\frac{d\sigma}{dq d\omega} = \frac{2\lambda^2}{\pi q} \text{Im} \left(-\frac{1}{\varepsilon(q, \omega)} \right). \quad (1)$$

Here, $\varepsilon(q, \omega)$ is the dynamic dielectric function, with $\lambda = \frac{h}{mv}$, where λ is the wavelength (in Bohr radius) of the incident particle, the scattering vector $q = \frac{2 \sin \theta}{\lambda}$, and the energy loss $E = \hbar\omega$. EELS spectra for the RGS which form at low temperature have been published and interpreted, identifying the specific crystal excitations to which the beam gives up energy as it traverses a thin slab of crystal. The results are shown in Fig. 1 of the study of Bernstorff and Saile.⁹

Two radiative loss mechanisms also captured by EELS spectra are the Bremsstrahlung and Cherenkov effects. Both of these interactions emit light that can be assumed to escape but contribute

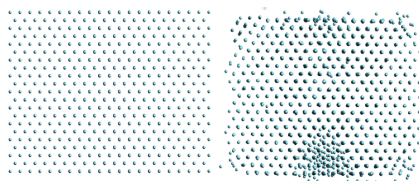


FIG. 1. A visual comparison between a pristine (left) and irradiated (right) Ar_{4631} cluster by the $10 \text{ e}/\text{\AA}^2$ pulse mentioned in the text.

negligibly to the dynamics of the sample. This is because RGSs are transparent to visible light due to the large band gap E_G , and to x rays, created by the Bremsstrahlung because of the small sample size and cross section. The nonradiative forms of energy loss by electrons beams in matter are valence electron excitation and ionization, inner-shell excitation and ionization, plasmons, excitons, and direct phonon excitations. Although inner shell excitations transfer more energy to the sample, they are much rarer events with smaller cross sections and are ignored in this model. Direct phonon excitations are similarly ignored due to their small cross sections relative to plasmons and excitons.

II. METHODS

A. The model

The dominant features seen in experimental EELS spectra, consist of two types: plasmon and exciton excitation. However, we first consider the ballistic PKA events, akin to Rutherford scattering. The maximum energy transfer from the beam electron to a nucleus occurs for a head-on collision and may result in irreversible nuclear displacement if this energy transfer exceeds the displacement energy, which depends on the atomic number (it is 20 eV for copper). The effect has been studied extensively in high-voltage (e.g., 1 MeV) TEM using a continuous electron beam. The total cross section for this process (about 10^{-6} \AA^2) is much smaller than that for elastic scattering or inelastic excitation of valence electrons—for an atomic number of 20, these are both equal to about 10^{-2} \AA^2 . The volume fraction of nuclei displaced by a head-on collision is¹⁰ $\text{Cd} = \frac{\sigma T}{|\epsilon|}$, where σ is the cross section for displacement of a nucleus in a head-on collision (scattering angle 180° , largest energy transfer), j is the beam current density, and T the exposure time. For light elements at 1 MeV, $\text{Cd} = 0.0125$ with a continuous TEM current of $0.2 \mu\text{A}$ and a beam diameter of $5 \mu\text{m}$ after one minute. Since typical UED average currents are typically about 1 nA (for a bunch charge of a pA at a repetition rate of 1 KHz) and beam diameters, limited by space-charge, even larger, the volume fraction of atomic displacements in UED will be far less than this, and the effect can be neglected compared to other inelastic processes at these current densities.

Plasmon excitations are the dominate form of collective electronic excitation specially for higher Z RGS⁹ (also for proteins in ice¹¹), and they are highly localized and damped, as suggested by their broad spectral linewidth corresponding to a lifetime of ~ 100 as. The products of plasmon decay are believed to be the main cause of radiation damage in Cryo-EM, on the basis of their EELS spectra.¹¹ For simulations with a time step larger than 100 as, the plasmon excitation decay is simply modeled as a re-scaling of sample's nuclear velocities by the plasmon energy. The plasmon energy is proportional to the square root of the density of valence electrons. Although a perhaps bad approximation, the RPA dielectric function is used in conjunction with Eq. (1) to determine the plasmon excitation cross section.

Unlike plasmons, Frenkel exciton excitations in RGS are relatively long lived (~ 25 ns),¹² and are also prominent in experimental EELS spectra. In order to incorporate the collective effect on the structural dynamics, the complete static Coulomb interaction (explicit electrostatics between all nuclei and electrons) was

computed for the sample. Excitons are then incorporated by adding a new set of Rydberg/Wannier states ($\phi_n(r)$) in addition to the non-interacting Hartree-Fock (HF) states, given by the Wannier equation [Eq. (2)] with static dielectric constant $\epsilon(0)$ [$\alpha_i(0)$ being the static polarizability of atom i with Vol being the sample's volume and μ being the exciton mass],

$$\left(-\frac{\hbar^2}{2\mu}\Delta + \frac{1}{\epsilon(0)}\frac{e^2}{r}\right)\phi_n(r) = E_n\phi_n(r), \quad (2)$$

$$\epsilon(0) = 1 + \frac{4\pi}{\text{Vol}}\left(\sum_i^{\text{atoms}}\alpha_i(0)\right). \quad (3)$$

At every time step, t , of the stochastic simulation, all atoms are subject to the following operator:

$$X(t)ws^\dagger + Y(t)w^\dagger s. \quad (4)$$

Where $\{w^\dagger, w\}$ and $\{s^\dagger, s\}$ are creation and annihilation operators for Wannier orbitals and Slater (HF) orbitals, respectively. Both $X(t)$ and $Y(t)$ take random Boolean values, $\{0, 1\}$. The probability of 1 is dependent on the half-life¹² of the Poisson process of decay ($X(t)$) and the excitation cross section and beam fluence at time t ($Y(t)$). Since the sample is homogeneous, excitons are supposed to be stationary, affecting only the total electrostatics and thereby dynamics.

In previous simulations of X-ray Free-Electron Lasers (XFEL) beam damage,¹³ secondary radiation damage (that is, atomic ionization due to ejected photo-electrons) was considered. This is very different in the case of electron beams, where the atomic electrons ejected by the beam mostly have energies of a few electron volts and where lower energy excitons and plasmons dominate the spectra. These low energy secondary electrons are detected and used for image-formation in the Scanning Electron Microscope (SEM). They have been detected and studied in detail in time co-incidence with the plasmons seen in EELS spectra in order to understand their origin.¹⁴ In this work, we have assumed that secondary electrons do not have sufficient energy to ionize neighboring RGS atoms. Due to the Ramsauer-Townsend effect, we can expect that because of the high band gap of rare gas solids, the inelastic mean free path of ejected secondary electrons will be greatly increased.¹⁵

B. Stochastic molecular dynamics

In order to study radiation damage, we wish to obtain the time-resolved structure factor and the radial distribution function (trRDF), both of which may be computed by MD simulations. MD was used to track the location of nuclei at every time step during irradiation. The MD was implemented by integrating Langevin's equations with a heat bath parameter γ (in a.u.). In the limit $\gamma \rightarrow 0$, we recover Newtonian dynamics. The primary difficulty with this implementation of MD is the need for constant modifications of the potential energy surface and the subsequent excited state dynamics as ionization events occur during the MD. These simulations should be performed on RGS clusters on time scales of typical electron pulses (~ 10 ps) and length scales of typical proteins (~ 100 Å).

Ideally, the *Ab Initio* MD (AIMD) simulation with dynamics based on the time-dependent Schrödinger equation would be desirable to model such a complex environment. However, due to size

and time limitations, post-Hartree-Fock methods are not feasible. Similarly, dynamics based on Kohn-Sham Density Functional Theory (KS-DFT) is also not quite feasible. Although larger space-time scales may be accessed than those by post-HF methods, the desired space-time scale is still not achievable. In addition, KS-DFT as currently developed may not fully capture the dynamical electron correlations, which result in van der Waals interactions, the only form of cohesion in RGS.

Instead, the Hartree-Fock equations were solved using a STO-6G basis set for a range of inter-nuclear distances between two RGS atoms, with either or both excited (excited orbitals were solutions to Wannier's equation). This formed a potential energy surface and was used to create a tabulated potential (implemented with resolution of 0.005 Å) on which nuclei may be propagated via classical MD. In addition to this tabulated potential, a London dispersion potential was included from the first Padé approximant of the Casimir-Polder interaction to incorporate long-range cohesion.

III. RESULTS

A single simulation is demonstrated to show the capabilities of this program. In this simulation, a 1 MeV electron Gaussian pulse of 1 ns (0.5 ns FWHM) duration, $10 \text{ e}/\text{Å}^2$ fluence, and $1 \mu\text{m}$ spot size ($\frac{\pi}{4} \mu\text{m}^2$ beam area) irradiated a solid Argon cluster, Ar₄₆₃₁, centered and equilibrated at 50 K before the pulse. Visually, the results are shown on Fig. 1, and this figure compares a cluster at a pristine condition to after the pulse has left the sample. The energy of the molecular system is shown in Fig. 2 and demonstrates discrete jumps, both exciting and relaxing, in potential energy corresponding to the exciton vertical transitions [see Eq. (4)]. The difference in the total energy before and after the simulation (or pulse duration) constitutes the energy deposited in the sample (in this case 2.1 keV) or a dose (when divided by clusters mass) of 1.1 MGy.

Because of the stochastic nature of this simulation, averages over many simulations are needed to simulate experimental results.

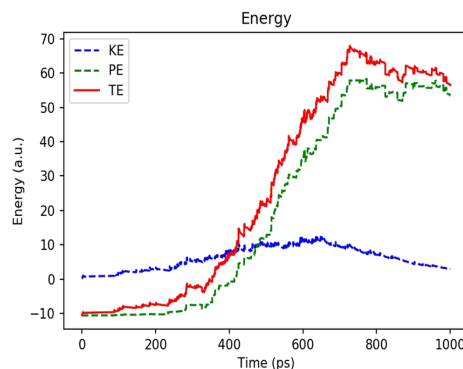


FIG. 2. This plot shows the evolution of the system's energy as a function of time during irradiation of the pulse mentioned in the text. $\text{KE} = \frac{1}{2} \sum_i m_i v_i(t)^2$, $\text{PE} = \frac{1}{2} \sum_{ij} V_{ij}(t)$, and $\text{TE} = \text{KE} + \text{PE}$.

21 September 2023 03:04:48

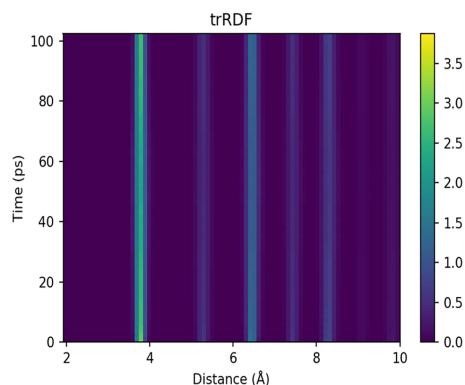


FIG. 3. Averaged (over 24 simulations) time-resolved (vertical axis) radial distribution function (z-axis, i.e., the color in a.u.) plotted against the interatomic distance (horizontal axis) for an experiment irradiating Ar_{4631} with a 1 MeV electron beam with $1 \text{ e}/\text{\AA}^2$ total fluence and $1 \mu\text{m}$ spot size.

In this case, a set of 24 simulations of a 1 MeV electron pulse of 100 ps (50 ps FWHM) duration, $1 \text{ e}/\text{\AA}^2$ fluence, and $1 \mu\text{m}$ spot size was used to irradiate the aforementioned cluster, Ar_{4631} . The results of the simulation-averaged atom-averaged trRDF are displayed in Fig. 3. This figure shows minimal radiation damage with this $1 \text{ e}/\text{\AA}^2$ fluence pulse. Energy calculations show an averaged deposited dose of 193 kGy.

Also investigated was the effect of the coupling, γ , of the sample to a Langevin thermostat from just plasmon excitations (ignoring exciton excitations). The results from varying this parameter to a 1 MeV electron pulse of 100 ps (50 ps FWHM) duration, $10 \text{ e}/\text{\AA}^2$ fluence, and $1 \mu\text{m}$ spot size are shown in Fig. 4. This figure shows that weakly coupled systems, i.e., isolated clusters in vacuum, suffer from radiation damage to a higher degree than samples connected to a thermostat (which would effectively cool the sample).

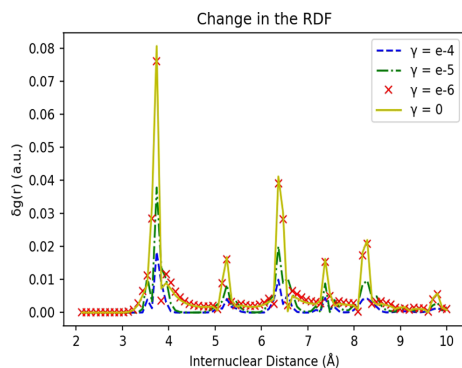


FIG. 4. The change in the integrated trRDF when compared to unirradiated sample for four different couplings to the Langevin thermostat.

IV. CONCLUSION

This program allows the analysis of radiation damage mechanisms and excitations responsible by suppressing each of them in turn. It makes use of EELS spectra to identify important excitations for a given system. Damage mechanisms may be studied under a wide range of conditions. These include beam energy, exposure time, fluence (dose), and sample temperature. The program's choice of RGS is well-suited as a first step in understanding radiation damage in soft-matter on these time and length scales. RGS are known for their small cohesive energies and thus radiation sensitivity. It is thus unsurprising that a dose of 1–5 MGy appears to cause dissociation of the cluster on these time-scales. By comparison, the maximum exposure/dose used in Cryo-EM (protein/soft-matter) samples are roughly $<30 \text{ MGy}$.⁴ These samples in addition to being in the bulk (with neighboring unirradiated sample) have much larger cohesive energies (from covalent bonds) than these RGS clusters. Apart from covalent bonds, RGS and protein matter have similar EELS spectra¹¹ in the form of excitons (albeit much less dominant) and plasmons. van der Waals interactions are important in both systems (RGS and protein crystals).

Although radiation damage from XFELs are largely irreversible, radiation damage from electron beams may be reversible. Excitons mostly undergo radiative decay, while plasmons decay into phonons producing heat. This heat may or may not be conducted away, depending on the sample geometry. Excitons are more damaging (potentially irreversible) when they are in close proximity in space and time (Fig. 1 shows an example of this effect). Distancing these electronic excitations in time (exciting them at different times and allowing them to decay) could explain the experimental results of ultrafast electron beams in paraffin showing increased resistance to damage.¹⁶

The exposures required to resolve the structure of an individual protein (in single-particle mode) require at least one elastic event per non-hydrogen atom⁴ and therefore require a total exposure of $\sim 10^3 \text{ e}/\text{\AA}^2$. This is not possible in a single shot with pulsed photocathode sources due to space-charge limitations.

SUPPLEMENTARY MATERIAL

See the [supplementary material](#) for additional specifics. The code performing all these simulations is freely available at <https://github.com/jcandane>.

ACKNOWLEDGMENTS

This work was funded by NSF STC (Award No. 1231306) by the Helmholtz Association through the CFEL at DESY, the Swedish Research Council (VR), and the Swedish Foundation for International Cooperation in Research and Higher Education (STINT).

REFERENCES

- ¹R. Neutze, R. Wouts, D. van der Spoel, E. Weckert, and J. Hajdu, *Nature* **406**, 752 (2000).
- ²J. C. H. Spence, *IUCr* **4**, 322–339 (2017).
- ³V. Šrajer and M. Schmidt, *J. Phys. D: Appl. Phys.* **50**, 373001 (2017).
- ⁴R. Henderson, *Q. Rev. Biophys.* **28**, 171–193 (1995).
- ⁵J. C. H. Spence, *Struct. Dyn.* **4**, 044027 (2017).

- ⁶J. Maxson, D. Cesar, G. Calmasini, A. Ody, P. Musumeci, and D. Alesini, *Phys. Rev. Lett.* **118**, 154802 (2017).
- ⁷L. Van Hove, *Phys. Rev.* **95**, 249–262 (1954).
- ⁸R. Egerton, *EELS in the Electron Microscope* (Springer, 2011).
- ⁹S. Bernstorff and V. Saile, *Opt. Commun.* **58**, 181–186 (1986).
- ¹⁰L. Reimer, *Transmission Electron Microscopy* (Springer, 2013).
- ¹¹S. Q. Sun, S.-L. Shi, J. A. Hunt, and R. D. Leapman, *J. Microsc.* **177**, 18–30 (1995).
- ¹²J. W. Keto, R. E. Gleason, and F. K. Soley, *J. Chem. Phys.* **71**, 2676–2681 (1979).
- ¹³C. Caleman, M. Bergh, H. A. Scott, J. C. H. Spence, H. N. Chapman, and N. Timneanu, *J. Mod. Opt.* **58**, 1486–1497 (2011).
- ¹⁴F. J. Pijper and P. Kruit, *Phys. Rev. B* **44**, 9192 (1991).
- ¹⁵J. C. H. Spence, *Micron* **28**, 101–116 (1997).
- ¹⁶E. J. VandenBussche and D. J. Flannigan, *Nano Lett.* **19**, 6687–6694 (2019).

APPENDIX D
JCP PERMISSION

The JCP article is posted with the required permissions, Julio J. Candanedo

Copyright © 2023 Julio Candanedo