Accelerating Computational Chemistry Algorithms: towards Accurate Binding Free Energies

Fiona Chuo Yan Yu

November 2025

Australian National University

A thesis submitted for the degree of
Doctor of Philosophy
of The Australian National University

Author Declaration

I declare that the research presented in this thesis comprises original work that I carried out during my candidature at the Australian National University. This work has not been previously submitted for any other degree or diploma at any university or institution.

Fiona Chuo Yan Yu
17 November 2025

Acknowledgements

I would like to begin by expressing my immense gratitude to my PhD supervisor, Prof. Giuseppe Barca, who introduced me to this strange world operating at the intersection of computer science and chemistry. Thank you for you consistent support and guidance over the last few years, especially during difficulties in my personal life. I consider myself very lucky to have had you as my PhD supervisor.

I would also like to thank the other members of my panel. To Prof. Julian Gale, thank you for your consistent, insightful comments and suggestions as well as your passion for computational chemistry. To Assoc. Prof. Minh Bui, thank you for supporting me and offering helpful comments.

To my research group, thank you all for supporting me academically and emotionally. In particular, I would like to thank Dr. Jorge Gálvez Vallejo, Chris Seidl, Ross Pure, Ryan Stocks, Elise Palethorpe and Calum Snowdon. Thank you for helping me set up calculations or set up software on clusters, acting as a soundboard for ideas, double checking my knowledge of science with, or empathising with me through our respective research life journeys.

I would also like to say thank you to my friends for the last few years. There are too many of you to name specifically, and for that I consider myself very lucky to have this many people who care about me. Thank you to those friends who are also enduring or have endured the PhD journey, I felt less alone during the difficult moments of my PhD. Thank you to my friends for your book, film and museum recommendations, these kept my mental state alive. Thank you to those friends who baked brownies or cheesecakes for me, this thesis was fuelled in part by butter and sugar.

Ich möchte auch meiner Deutschlehrerin Eva danken. Ich habe so viel von dir gelernt, und du warst auf jeden Fall ein Grund, warum mein letztes Jahr meines PhDs nicht so schlimm war.

Lastly, I would like to thank my family for their consistent support. To my brother who encouraged me to do whatever I wanted unapologetically. 婆婆和公公，谢谢你们一直以来的祝福。 To my mum, my unwavering supporter, thank you for all your sacrifices in providing for me and my brother, thank you for encouraging me to pursue my dreams which was never an option for you. I hope you come to view this work as part of your achievements.

Abstract

Advances in high performance computing (HPC) are pivotal for accelerating drug discovery by offering the capacity for large scale high accuracy virtual screening. There has been considerable interest in the large scale application of the Quantum Mechanical / Poisson-Boltzmann Surface Area (QM/PBSA) end point method towards predicting binding affinities of protein-ligand systems. However, its application is severely limited by the high computational costs of traditional QM methods as well as the slow performance of existing PBE solvers for PBSA calculations.

This thesis presents methods and algorithms to overcome such bottlenecks. The first half of this thesis is dedicated to accelerating QM calculations. An automated accurate molecular fragmentation scheme is presented that divides large systems into smaller, computationally feasible partitions, whilst enhancing algorithmic parallelisability. Its applicability on protein and lipoglycans/glycolipids is demonstrated. On the other hand, improved initial guess methods for self-consistent field (SCF) calculations are presented (basis set projection and fragmentation) and their performance is systematically analysed against the traditional superposition of atomic density (SAD) scheme. Results consistently indicate the improved performance of SCF calculations with non-SAD schemes.

To address the lack of fast PBE solvers to model solvation, a high-performance GPU-accelerated solver is presented. The algorithm exploits the sparsity pattern exposed in its application on molecular systems to accelerate matrix-vector contractions prevalent in conjugate gradient solvers, outperforming existing multi-core CPU and GPU-based PBE solvers.

These tools are integrated into a QM/PBSA workflow and applied to large biologically relevant protein-ligand complexes to predict binding affinities. The influence of various factors—fragmentation level, protonation states, and solvation methods—on the performance of the proposed QM/PBSA workflow is systematically analysed and compared to other computational approaches including alchemical free energy and scoring function methods.

This thesis seeks to improve upon the accuracy, computational efficiency and feasibility of large scale QM/PBSA workflows by leveraging chemical concepts and HPC optimisations. Beyond drug discovery, the methodologies presented have broad applicability to molecular molecular systems beyond proteins, supporting research in materials science, chemistry and energy applications. Such advancements become increasingly important as HPC systems continue to evolve, offering the potential to study molecular systems at even larger scales and and higher accuracy.

Contents

Introduction

With the average cost of developing a pharmaceutical drug being $1 billion USD, there has been considerable interest in accelerating this process to reduce its cost. The pivotal role of computers in accelerating drug discovery is unmistakable, with the progress of the latter relying heavily on the advancement of the former. This process involves predicting the solvated binding affinity or free energy (ΔG_{bind}) of small drug-like molecules (ligands) to target protein structures containing thousands of atoms for large scale virtual screening—a task that demands considerable computational power and, consequently, high performance computing (HPC) systems. Traditionally, computational advancements were predominantly driven by Dennard scaling and Moore's law, which enabled higher clock speeds. Dennard scaling stated that transistors could operate at higher frequencies by simultaneously shrinking their size and and increasing their on-chip density without raising power density,[1] while Moore's law posited that the number of transistors per chip would double approximately every two years at constant cost.[2] However, physical constraints including excessive heat generation, current leakage problems and elevated power consumptions have led to diminishing returns, marking the end of Dennard scaling and the decline of Moore's law.[3]

Consequently, HPC systems, whose computational power were once primarily drawn from central processing units (CPUs), have progressively shifted to massively parallel, heterogeneous architectures, typically featuring thousands of graphical processing units (GPUs) as accelerators. A detailed overview of heterogeneous architectures and the role of CPUs and GPUs is provided in Chapter 2 Section 2.1.1. These developments have enabled the realisation of exascale machines, capable of performing 10^{18} floating-point operations per second (FLOPs), thereby significantly expanding opportunities for accelerating drug discovery. However, the benefits of such parallelism are not without costs and several computational challenges persist. To further contextualise the computational complexity of modelling protein–ligand binding for drug discovery, the following sections discuss the HPC and computational chemistry considerations associated with these methods.

1.1 High Performance Computing Challenges in Drug Discovery

Modelling protein-ligand systems at large scales presents a tremendous computational challenge, in part due to the considerable size of such systems which are typically of the order of thousands of atoms, as well as the need to simulate many such systems in large scale virtual screening workflows. This is also further complicated by the fact that such systems are solvated and solvation effects must be accounted for. Thus, tackling such a task requires leveraging HPC and careful consideration of the associated challenges.

1.1.1 Computational Demand

Although a multitude of computational techniques have been developed to predict binding affinities, accurate methods are accompanied by steep computational costs due to a combination of reasons including sampling bottlenecks, high formal complexities and slow convergence. A detailed description of the following computational methods mentioned are provided in Chapter 2 Sections 2.3, 2.4 and 2.5.

For example, alchemical free energy (AFE) approaches which involve the alchemical transformation from ligand to another and considered to be one of the most accurate computational techniques, require extensive sampling of many intermediate states to achieve statistical convergence and accurate results.[4] The computational cost is further amplified as each sampling step involves modelling not just the protein-ligand complex, but a much larger system: the complex is also solvated in thousands of explicit water molecules, increasing the total system size of the order of 10,000 atoms. Moreover, typically each intermediate sampling step typically requires a molecular dynamics (MD) simulation, often spanning 1 to 10 ns with sampling frequencies of the order of 10 ps, rendering the sampling of even a single intermediate state to be computationally burdensome. In systems where large conformational changes or slow structural transitions, convergence becomes even slower, further compounding the computational demand.[5,6] Importantly, in large virtual screening workflows where hundreds or thousands of protein-ligand complexes are modelled, such approaches quickly becomes infeasible.

Due to their steep computational cost, the molecular system is typically treated with molecular mechanics (MM) in AFE methods, specifically, force fields (FF). However, it is well known that FFs do not sufficiently capture electronic effects critical in intermolecular binding (*e.g.* charge transfer, π-π interactions, hydrogen bonding and many body effects).[7] Conversely, quantum mechanical (QM) methods which provide highly accurate characterisations of molecular systems, exhibit extremely high formal complexities. For instance, Hartree-Fock (HF) which provides only a simple, qualitative approxima-

tion to the energy of the system already scales as $\mathcal{O}(N^4)$ where N is the number of particles.[8] Paired with more accurate methods such as the second-order Møller–Plesset perturbation (MP2) or coupled cluster (CC) theory levels, the formal complexity quickly rises up to $\mathcal{O}(N^{5-8})$. Therefore, even outside the application of AFE or routines involving dynamics/sampling, QM methods on their own prove to be computationally intractable on such large systems including thousands of atoms.

Even other computational techniques which involve modelling solvation effects implicitly and therefore are cheaper, for instance, the Poisson-Boltzmann Surface Area (PBSA) solvation model, also incur high computational costs. These approaches involve solving the linear Poisson-Boltzmann equation (PBE) equation, which often use solvers such as conjugate gradient (CG) that comprise $\mathcal{O}(n^2)$ scaling matrix-vector multiplication routines, with n is the matrix-side dimension. With fine spatial resolutions typically required to achieve accurate results (of the order of ~ 0.1Å),[9–11] n is frequently of the order one or ten million for protein systems,[12,13] further exacerbating the computational workload.

Clearly, computational methods for modelling protein-ligand systems and predicting binding affinities involve significant computational demands that render the application of such approaches as they are on large protein-ligand systems to be infeasible. The advent of HPC systems, increasingly dominated by accelerators such as GPUs, provides an opportunity to improve these methods by leveraging the massive parallelism of such machines. Therefore, understanding the challenges associated with their parallel implementations also becomes critical.

1.1.2 Parallel Implementation

Parallelism in computing refers to running multiple tasks simultaneously across multiple execution units and its utilisation to accelerate program performance. Though parallel architectures have realised powerful modern supercomputers, the exploitation of such architectures remains non-trivial.

The parallel implementation of a program to exploit parallel architectures is met with several concerns. First, as stated by Amdahl's law,[14] the maximum speedup afforded from parallelism is limited by the serial portion (f_s):

$$\text{speedup} = \frac{1}{f_s + \frac{(1-f_s)}{p}} \tag{1.1}$$

where p is the number of processors available. This law demonstrates that the overall performance gain is constrained when a considerable portion of the computation remains serial, limiting performance gain from parallelism and scalability. Hence, for a parallel implementation to be effective, the total computational workload must be suf-

ficiently large and a substantial part of it must be parallelisable.

Second, on its own, parallelism inherently introduces overhead associated with distributing data work across the execution units and subsequently combining their results. To fully realise performance gains afforded by parallelism, factors including load imbalance, communication overhead and synchronisation must be carefully managed. Overall, it is crucial that the overhead incurred by parallelism remains small relative to the amount of parallelisable work.

Third, the majority of software written for serial execution must be entirely refactored or redesigned to fully exploit parallel architecture. Redesign is often necessary to expose parallelisability and maintenance of accuracy since factors such as race conditions or data dependencies arising from parallelism can lead to deviations in results. Furthermore, a variety of parallel programming models, such as OpenMP or CUDA, are available to target different architectures (CPU and GPU, respectively). Each model presents its own challenges, including high latency data transfers between host and device for GPUs, as well as the need to redesign or refactor the algorithm to exploit hardware specific parallelism. An overview of these parallel programming models can be found in Chapter 2 Section 2.1.2.

With such concerns in mind, researchers have successfully developed a range of techniques to accelerate computational chemistry algorithms, including AFE, QM and PBSA methods, by leveraging parallel architectures. For example, the workflow of AFE methods is especially conducive for exploiting massively parallel architectures as sampling of distinct intermediate states or structures can be performed largely independently.[15] To this end, AMBER introduced their GPU-accelerated AFE modules: thermodynamic integration (TI)[16] and free energy perturbation (FEP).[17] Its FEP module has been demonstrated to deliver two orders of magnitude speedups relative to single core CPU implementations on systems including tens of thousands of atoms.[16,18] NAMD also developed a GPU implementation of FEP which yielded 30$\times$ speedup over a multi-core CPU implementation on a molecular system of more than 98k atoms.[19] Additionally, other tools such as OpenMMTools and YANK have been developed to extend upon other GPU-accelerated MD software such as OpenMM to facilitate AFE computations.[20] A detailed description of the TI and FEP approaches are presented in Chapter 2 Section 2.3.2.

On the other hand, GPU-accelerated QM software such as TeraChem, GPU4PySCF and EXESS have also been developed to capitalise the performance speedups afforded by parallelism and have been scaled up successfully across multiple GPUs.[21–30] For example, a highly parallel eigenvalue solver has been developed for EXESS to forgo the mostly non-parallelisable diagonalisation step—typically part of the traditional self-consistent field (SCF) calculation—thus enhancing algorithmic parallelisability.[31] Other avenues of optimisation also include optimising Fock matrix construction and tailor-

ing the evaluation of electron repulsion integrals on GPUs.[28,32] Furthermore, molecular fragmentation, which involves dividing the system into smaller fragments, has also been implemented within[33,34] to both reduce formal complexity ($\mathcal{O}(N)$) and enhance algorithmic parallelisability of QM algorithms. A detailed explanation of the SCF routine and molecular fragmentation can both be found in Chapter 2 Section 2.4.1.

Efforts have also been dedicated to accelerating implicit solvation models such as PBSA using parallel architectures. These include Delphi's multi-core CPU-based module,[35–39] AMBER's CUDA-enabled *pbsa.cuda* module,[12,40] a GPU-accelerated direct-sum boundary integral method employing a generalised minimum residual (GMRES) solver,[41] and a third-party GPU adaptation of DelPhi.[42,43] Some of these GPU implementations have been reported to demonstrate substantial performance advantages, with DelPhi achieving approximately $10\times$ speedups and AMBER's *pbsa.cuda* up to $50\times$ compared to their CPU-based counterparts.[12,42]

Though the computational performance of these approaches has improved significantly from parallelism, such methods remain seldom, if at all, used in large scale virtual screening workflows. The majority of PBSA software are predominantly CPU-based[38,44] and fail to exploit the massive parallelism of GPUs, whereas GPU-based implementations such as AMBER's *pbsa.cuda* do not efficiently utilise the hardware, as will be demonstrated in Chapter 5. For AFE, though the sampling of intermediate states can be run independently of one another, its prior set-up step often needs to be performed manually by an expert with domain knowledge to decide the order of alchemical transformations from one ligand to another,[45,46] not to mention convergence concerns also arise from independent handling of states.[47] Similarly, QM methods including fragmentation often require manual intervention. The majority of molecular fragmentation methods are not automated, often relying on chemical intuition to fragment systems manually, and if they are automated, they are typically limited to a particular class of systems. These problems concerning AFE and molecular fragmentation are not of a computational or algorithmic origin, but rather, are rooted in the underlying chemical theory. For example, it is the latter which provides an explanation for the best network of alchemical transformations to use in AFE. Therefore, in addition to computational challenges explored herein, it is also essential to consider the chemical models and theory to develop efficient and practical computational techniques for drug discovery.

1.2 Computational Chemistry Challenges in Drug Discovery

In computational chemistry, the main objective involves developing models which strike a balance between the two competing priorities of maintaining accuracy and reducing

computational costs. Today, many computational methods of varying complexities and accuracies have been developed to model the binding behaviour of protein-ligand systems and subsequently predict the associated binding free energies.

On one end of the spectrum lies the widely adopted Molecular Docking approaches due to their low computational overhead.[48] Therein, various binding conformations of protein-ligand complexes are predicted and the corresponding free energies or binding affinities are computed using scoring functions.[49] A more detailed description of docking methods is provided in Chapter 2 Section 2.3.1.

Although docking methods have demonstrated some success in achieving good correlation with experiment,[50,51] their performance strongly depends on the type of molecular system being studied, and overall they still display low predictive ability across protein-ligand systems, namely, low correlation with experimental results. For example, in a comprehensive study investigating the performance of 10 docking software across a dataset comprising 2002 protein-ligand complexes, the highest Pearson correlation coefficient (R) observed was 0.57.[52] Additionally, therein, multiple docking software exhibited negative correlation coefficients where R varied between -0.50 and 0.57. Such an observation is not unique to this study, other assessments have also demonstrated low or negative correlations.[50,51,53–55]

In contrast to docking methods which exhibit generally low predictive ability, on the other end of the spectrum of complexity are the aforementioned alchemical free energy methods. As described earlier in Section 1.1, such approaches are regarded as highly accurate however are burdened by high computational costs as well as manual intervention being required for the set-up procedure before any thermodynamic calculations, severely limiting its scalability. Specifically, this set-up stage involves defining all the states, which is typically the same target protein paired with many different ligands, as well the thermodynamic pathway between these states. The burden of such a manual task is exacerbated when hundreds or thousands of protein-ligand systems need to be modelled in large scale virtual screening. This automation of AFE methods currently remains an active area of research, with the majority of efforts in early development in or are limited to specific software programs.[46,56–58] In view of this, AFE methods remain largely unsuitable for large scale virtual screening and are limited to the lead optimisation stage of drug discovery, where the 'lead' candidates identified are further optimised.[59–61]

To this end, researchers have looked towards alternative approaches which offer a balance between accuracy and computational efficiency. In particular end point methods, which are only concerned with the end states (bound complex, unbound protein and unbound ligand) and are therefore much cheaper than AFE methods have gained significant traction. Among these, a popular method termed Molecular Mechanics Poisson-Boltzmann Surface Area (MM/PBSA) has emerged as an appealing alterna-

tive for estimating binding free energies. A full description of MM/PBSA is provided in Chapter 2 Section 2.3.3.

Briefly, in MM/PBSA, the binding free energy (ΔG_{bind}) of a ligand to a protein is approximated as

$$\Delta G_{bind} = \langle \Delta E_{MM} \rangle + \langle \Delta G_{solv} \rangle \tag{1.2}$$

where ΔE_{MM} and ΔG_{solv} denote the gas phase interaction and solvation energies, respectively, whilst $\langle \ldots \rangle$ refers to an ensemble average over conformations obtained from a sampling procedure, typically MD simulations with explicit solvent. ΔG_{solv} is computed with the PBSA model and is further explained in Section 2.3.3. On the other hand, ΔE_{MM} is computed with molecular mechanics and is further decomposed as

$$\Delta E_{MM} = \Delta E_{int} + \Delta E_{elec} + \Delta E_{vdW} \tag{1.3}$$

where ΔE_{int} denotes the internal energy or 'bonded' interactions (*e.g.* bond stretching, angle-bending, dihedral angle torsion, and out-of-plane distortions), whilst the 'non-bonded' terms are made up of ΔE_{elec} and ΔE_{vdW} which denote the electrostatic and the van der Waals (vdW) energies, respectively. Generally, the electrostatic and vdW energies are treated with the Coulomb and Lennard Jones (LJ) interactions, respectively. The formulation of these classical terms are provided in Section 2.2.

Unlike AFE methods which require extensive sampling of intermediate states and prior manual definition of alchemical transformation pathways, MM/PBSA generally only involves one sampling routine per protein-ligand complex where the conformations of the bound complex, unbound protein and unbound ligand are extracted from.

Although MM/PBSA has become a popular tool in estimating binding affinities due to its balance between computational cost and accuracy, its large-scale application for virtual screening is still limited by several concerns. As described previously in Section 1.1, molecular mechanics do not sufficiently capture electronic effects critical in intermolecular binding and the incorporation of quantum effects in the binding description due to its improved accuracy over the traditional MM/PBSA method has gained significant attention.[5,7,62–66] For example, Maier *et al.* attained QM-based binding energies in excellent agreement with experiment across seven protein targets (R ranged between 0.81 and 0.97), compared with MM/PBSA, where the R coefficients varied between -0.22 and 0.89.[5] In another study, Gundelach *et al.* performed QM/PBSA calculations on a set of 10 ligands with the BRD4 target. Therein, QM/PBSA yielded significantly improved correlations with Spearman correlation coefficients (R_s) varying between 0.76 and 0.83, contrasting that of MM/PBSA where R_s ranged from -0.30 to 0.01.[66]

Hence, there has been considerable interest in the inclusion of quantum mechanics in end point methods, specifically, Quantum Mechanical / Poisson-Boltzmann Surface

Area (QM/PBSA) workflows for the prediction of binding free energies of protein-ligand systems. However, as described in Section 1.1, current linear PBE solvers for molecular systems remain prohibitively slow for large scale virtual screening. Furthermore, QM methods exhibit extremely high formal complexities and even employing other approaches such as molecular fragmentation to reduce its steep scaling are not without challenges as will be explained in Chapter 3. Due to the computational costs and current implementations of QM and PBSA algorithms, the application of QM/PBSA workflows has been largely confined to single target systems.[66–69] Therefore, this works includes two overarching aims: 1) to develop methods and algorithms to realise a computationally efficient QM/PBSA workflow at large scales; and 2) perform systematic analysis of its performance across diverse targets to identify potential limitations and optimise protocols.

1.3 Overview

The literature presented in this Chapter have highlighted the existing significant computational and chemistry challenges of existing computational methods in modelling large protein-ligand systems for large scale virtual screening of drug discovery, emphasising the need for an accurate, computationally efficient method for the computation of binding affinities at large scales. In particular, the end-point method, QM/PBSA, has emerged as a favourable strategy towards its prediction. However, its widespread adoption in large scale virtual screening remains constrained by several factors:

1 **Challenges of quantum mechanical methods.** The use of quantum mechanical methods is severely hindered by its steep computational scaling.

2 **Lack of fast Poisson-Boltzmann solvers.** The lack of fast, scalable solvers of the linear Poisson –Boltzmann equation for large biological systems such as protein–ligand complexes remains a bottleneck.

Finally, in addition to the above concerns, another area of concern is the overall lack of systematic performance evaluations of QM/PBSA workflows. Hence, another concern tangential to those listed above is:

3 **Limited performance evaluations of QM/PBSA workflows.** QM/PBSA workflow evaluations have largely been limited to single target systems due to its high computational cost and there is an overall lack of studies analysing its performance systematically across multiple targets.

Therefore, a major goal of this thesis is to develop algorithms and methodologies to realise computationally feasible large scale QM/PBSA workflows, as well as to explore

the role of different factors towards the accuracy of its prediction. The above concerns are addressed through the following aims, which are achieved in the corresponding chapters. Briefly, Aims 1 and 2 focus on developing techniques for accelerating QM methods for gas phase energies, Aim 3 is concerned with predicting solvation energies for large molecular systems, and Aim 4 combines techniques from the previous chapters to yield a computationally feasible QM/PBSA workflow and examine its performance.

Aim 1 Develop an accurate, automated molecular fragmentation algorithm for large biological systems (**Chapter 3**).

Aim 2 Explore alternative initial guess schemes towards accelerating quantum chemical calculations (**Chapter 4**).

Aim 3 Develop a high performance GPU-accelerated linear Poisson-Boltzmann solver for large molecular systems (**Chapter 5**).

Aim 4 Perform a benchmark analysis of a proposed QM/PBSA workflow (**Chapter 6**).

The computational techniques developed in herein for modelling protein-ligand systems provide researchers with powerful tools to study biological systems at large scales and high accuracy. These advances have broad applicability not just within the fields of drug discovery, chemistry and biology but also across disciplines such as materials science and energy since, as will be later demonstrated in the relevant chapters, the methods presented in this work can be generally applied to chemical systems outside of those with biological significance. Furthermore, the availability of a computationally efficient QM/PBSA method represents a significant step forward in accelerating drug discovery two main ways. First, by leveraging the parallel architecture of modern machines and novel algorithmic designs, the speed at which drug candidates can be screened is significantly faster. Second, due to the improved accuracy of screening techniques, the resulting, filtered pool of high potential pharmaceutical candidates is of substantially higher quality. Taken together, this work presents techniques which extend the frontier of computational modelling across multiple disciplines, as well as driving the advancement of drug discovery.

Publication List

First author publications:

1. **Yu, F.C.**, Galvez Vallejo, J.L. and Barca, G.M., 2024. Automatic molecular fragmentation by evolutionary optimisation. *Journal of Cheminformatics*, *16*(1), p.102.

2. **Yu, F.C.**, Seidl, C., Palethorpe, E. and Barca, G.M., 2025. Acceleration of Self-Consistent Field Calculations Using Basis Set Projection and Many-Body Expansion as Initial Guess Methods. *Journal of Chemical Theory and Computation, 21*(3), pp.1230-1248.

Co-authored publications:

1. Galvez Vallejo, J.L., Snowdon, C., Stocks, R., Kazemian, F., **Yu, F.C.**, Seidl, C., Seeger, Z., Alkan, M., Poole, D., Westheimer, B.M. and Basha, M., 2023. Toward an extreme-scale electronic structure system. *The Journal of Chemical Physics, 159*(4).

2. Stocks, R., Galvez Vallejo, J.L., **Yu, F.C.**, Snowdon, C., Palethorpe, E., Kurzak, J., Bykov, D. and Barca, G.M., 2024, November. Breaking the million-electron and 1 EFLOP/s barriers: Biomolecular-scale ab initio molecular dynamics using MP2 potentials. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1-12). IEEE.

Background and Methods

This chapter presents an overview of the core concepts and methodologies discussed across chapters.

2.1 High Performance Computing

2.1.1 Supercomputer Hardware

The idea of supercomputing is to combine the computational power of multiple computer systems and running these in parallel to perform highly computation-intensive or data-intensive tasks. The advent of supercomputing has allowed researchers to tackle large scale problems that would otherwise be impossible with conventional computers, due to time and cost constraints. The performance of a supercomputer is determined by the number of floating point operations per second (FLOPs).

Modern supercomputers have shifted towards heterogeneous architectures, comprising both CPUs and GPUs, these have enabled exascale machines capable of performing more than 10^{18} FLOPs such as Frontier, El Capitan and Aurora, with the majority of computational power drawn from GPUs. In addition to high FLOPs, supercomputers with heterogeneous architectures are characterised by their high energy efficiency and high-bandwidth memory.

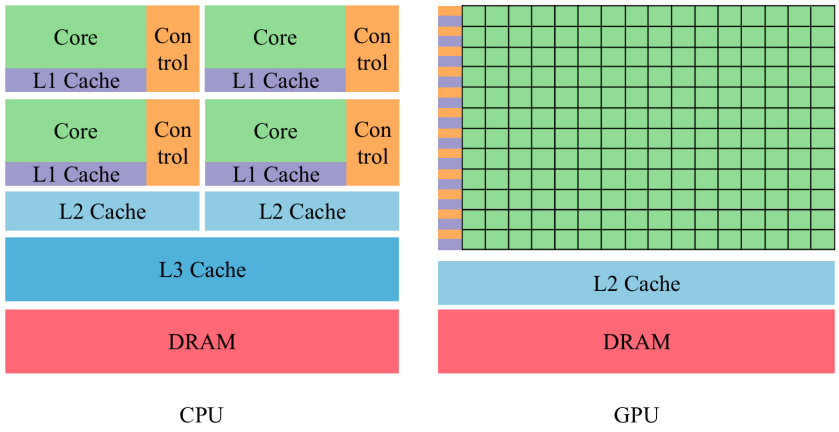


Figure 2.1: Comparison of the CPU and GPU architectures.

CPUs and GPUs are designed for different purposes; the design of CPUs enables it to execute a sequence of operations (thread) extremely fast and is able to execute a few in parallel, whereas the GPU is designed to execute thousands of these in parallel.[70] At the same time, GPUs tend to exhibit lower clock speeds than CPUs but the ability to perform thousands of tasks in parallel renders GPUs to have greater instruction throughput than CPUs. These different functions of CPUs and GPUs are reflected in their architecture which is shown pictorially in Figure 2.1. As illustrated in Figure 2.1, GPUs typically exhibit many more cores (hundreds or thousands) compared to CPUs which only comprise a few cores. This stark contrast gives rise to the varying degree of parallel efficiency in CPUs and GPUs; GPUs exhibit a much higher parallel efficiency than CPUs.

Since GPUs are designed to excel at highly parallel computations, a greater number of transistors are dedicated towards data processing (*e.g.* floating point operations), often making GPUs more energy efficient than CPUs for massively parallel workloads.[70] GPUs are designed to be latency hiding, that is, GPUs are able to disguise memory operation latencies with computation. In the case of CPUs, large data caches (*e.g.* L3 Cache) and complex flow control procedures are employed to minimise long memory access latencies.

The architecture of GPUs and CPUs are designed differently to achieve distinct aims. One salient distinction between the architectures of the two different processing units are the threads. Threads on CPUs are much more heavyweight than those on GPUs and therefore thread context switches are much faster and cheaper on GPUs.[71] Coupled with the fact that GPUs are designed to handle many lightweight threads concurrently, GPUs are able to achieve a higher instruction throughput than CPUs. However, GPUs are not inherently "better" than CPUs nor is the opposite true. CPU computing excels at control-intensive tasks, whilst GPU computing excels at data-parallel computation-intensive tasks. Thus, GPUs are not intended to replace CPUs, but rather,

leveraging GPU computing to complement CPU computing is the main objective in developing software that leverages the heterogeneous architecture of modern supercomputers. Such software comprise two parts: host code and device code. Host code is executed on the CPU whereas the device code runs on the GPU.[71]

Therefore, to efficiently utilise modern supercomputers, programs exhibit a combination of both parallel and serial portions, and a mix of both CPUs and GPUs is generally required to maximise performance. As GPUs enable highly parallel computations, there has been a large focus on heterogeneous architectures in modern supercomputers.

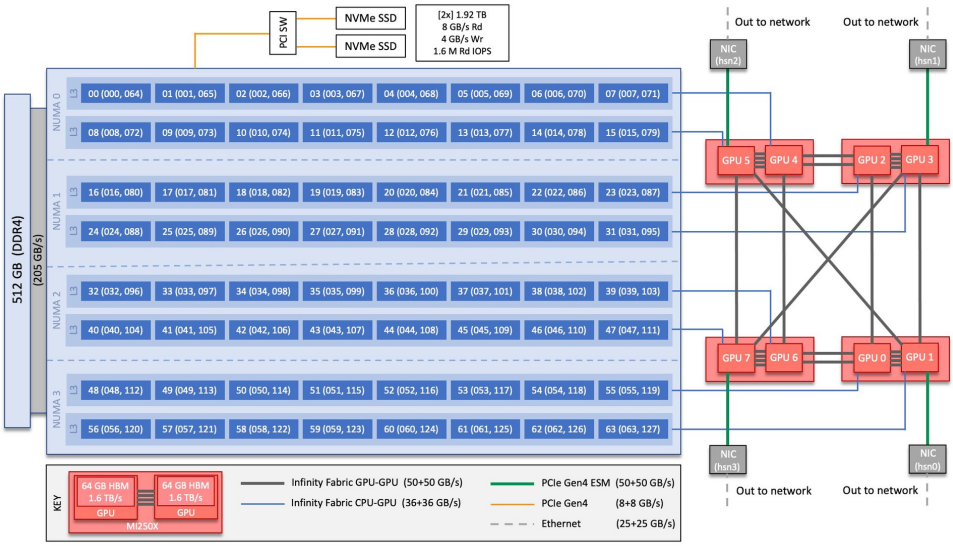


Figure 2.2: The architecture of a Frontier node. Each node consists of one consists of 64-core AMD Optimised 3rd Gen EPYC CPU and four AMD MI250X.

Another aspect of heterogeneous architectures to highlight is the high-bandwidth (HBM) of GPUs. HBM is crucial to enabling large scale computations on exascale systems. As its names suggests, HBM offers significantly higher bandwidth than conventional memory, providing higher computational throughput.[72] For example, as shown in Fig. 2.2, each GPU in the exascale Frontier machine, possesses 64 GB of HBM which can be accessed at a peak of 1.6 TB/s.

Figure 2.2 displays the architecture of a node of an exascale machine (Frontier). Each node of the Frontier supercomputer consists of one Optimised 3rd Gen AMD EPYC CPU and four Purpose Built AMD Instinct 250X GPUs (MI250X) as illustrated in Figure 2.2. The CPU has access to a 512 GB DDR4 memory and a bandwidth of 205 GB/s. Each MI250X contains two GPUs where each GPU is capable of performing 26.5×10^{12} FLOPs/s at its peak. The two GPUs on a MI250X are connected with Infinity Fabric with a bandwidth of 200 GB/s. Each EPYC CPU comprises 64 cores whereas each GPU has 14080 cores, highlighting the reliance on massive parallelism to achieve peak performance. Other exascale machines such as Aurora and El Capitan exhibit a

similar heterogeneous architecture with a reliance on GPUs for computational power.

Whilst this architecture has enabled Frontier to break the barrier of exascale computing, constraints associated with GPUs persist. Using the Frontier node as an example, one such limitation involves the different bandwidths, specifically, the Infinity Fabric connection between the GPU and CPU exhibit a much lower bandwidth (36 GB/s) than the communication between the main memory and the CPU (205 GB/s) and the connection between the GPU processor and the GPU memory (1.6 TB/s). Such a limitation is not constrained to Frontier, the majority of modern supercomputers with heterogeneous architectures exhibit similar discrepancies in the bandwidths. Thus, software developed for heterogeneous architecture is generally accompanied by this high latency host (CPU) to device (GPU) data transfer.

2.1.2 Parallel Programming Models

With heterogeneous architecture becoming more commonplace and the various hardware on which parallelism can be achieved on (multi-core CPUs and GPUs), differing parallel programming models have arisen accordingly. For example, OpenMP is a popular programming model to handle CPU parallelism. Initially developed to support CPU parallelism, OpenMP has since been extended to support accelerators such as GPUs. On the other hand, CUDA targets only GPU architectures. In this section, an overview is provided for both OpenMP and CUDA.

OpenMP

Open Multiprocessing, abbreviated as OpenMP, is an Application Programming Interface (API) for shared memory parallel programming that targets the low-level languages C, C++ and Fortran. OpenMP is one of the most commonly used programming models to achieve CPU parallelism. OpenMP implements parallelism by converting serial code into multi-threaded parallel code. This is achieved through the use of directives wherein the programmer annotates the code with directives recognised by compilers, which subsequently compiles it in an executable that can run in parallel. Specifically, in C/C++, the directives keywords always come after a pragma:

```
#pragma omp directive_name [clause_list]
```

Common examples of *directive_name* includes `parallel`, `for` and `single`. The clause list provides further detail to the compiler, for instance, this could concern information relating to data sharing or scheduling (dictates granularity of the workload).

OpenMP operates on a fork/join model[73] where the program executes serially until a `parallel` directive is encountered (see Figure 2.3). When this occurs, a group of threads is generated and each thread executes the task in the structured block following

the line where the `parallel` directive is declared. The structured block contains the parallel region where multiple threads are active and potentially run in parallel. At the end of parallel region, threads are joined and the code continues to run serially.

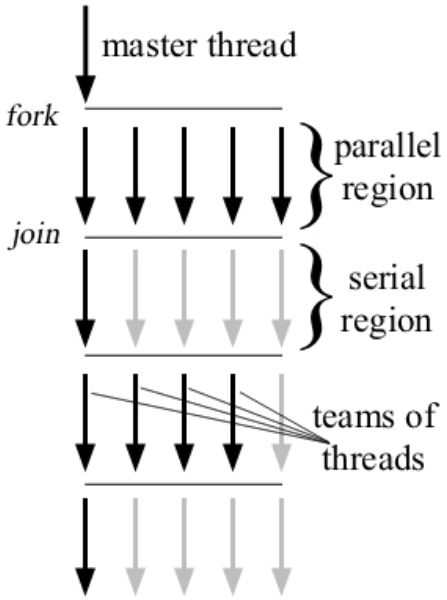


Figure 2.3: Schematic of the OpenMP execution model on CPUs.

OpenMP is a relatively simple model to implement to accelerate CPU-based software. Indeed, the DelPhi program which will be used later in Chapter 6, one of the most widely used linear PBE solvers, employs OpenMP for multi-core CPU acceleration.

Beyond CPU parallelism, as aforementioned, OpenMP has also been developed to support GPU parallelism. Although no programs employing OpenMP for GPU parallelism is used in this work, it is mentioned here for completeness.

In C/C++, the directive `target` is used to achieve this. Specifically, it maps variables to the data environment on the device and executes the construct on it.

A typical OpenMP GPU program has the following execution model:

1. Create variables on the CPU.
2. Copy data on the CPU to the GPU. Execution is transferred to the device.
3. Execution on the GPU.
4. Copy data from GPU memory to CPU memory.

Steps 2 to 4 belong to the `target` region. As will be shown in the following section, this GPU execution model bears similarity to that of CUDA.

CUDA

To achieve GPU parallelism, another successful programming model is CUDA. CUDA programming is especially powerful at handling data parallel computations. Data parallelism refers to operating on different data simultaneously and distributing data across cores. One unique characteristic of CUDA is that it employs a two-level thread hierarchy to organise threads.[71] As illustrated in Figure 2.4, this thread hierarchy is decomposed into blocks of threads and grids of blocks. All threads in the same grid share a global memory space, and threads belonging to the same block can cooperate. Specifically, cooperation involves block-local shared memory and synchronisation. Threads from different blocks cannot cooperate. Furthermore, each block is divided into multiple warps where each warp is a collection of 32 threads.[74] This is the most basic execution unit in CUDA programming. Every thread in the same warp must execute the same instruction at the same time on its own private data.

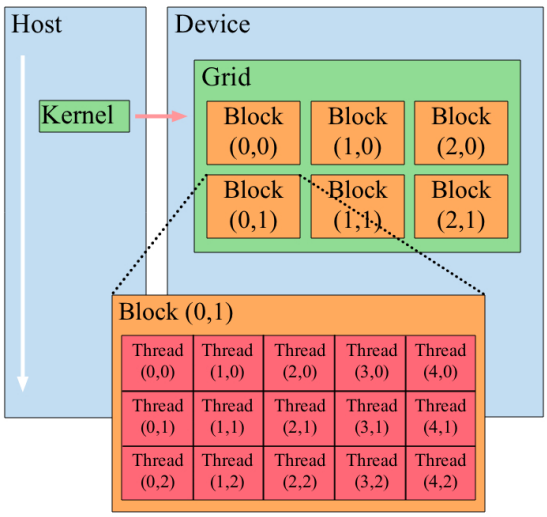


Figure 2.4: Two-level thread hierarchy in CUDA.

A typical CUDA program has the following structure:

1. Allocate memory to the GPU.
2. Copy data from the CPU to the GPU memory.
3. Invoke CUDA kernels to operate on the data located in the device memory.
4. Copy data back from GPU memory to CPU memory.
5. Destroy the GPU memory.

Unlike CPU-based software, data transfers between the host and the device are mandatory in any heterogeneous program. However, host-device data transfers have

much higher latencies than host-host and device-device transfers as mentioned in Section 2.1.1. Therefore, heterogeneous programs are always accompanied by high latency host-device transfers and should be offset by sufficient on-device computation for efficiency.

The CUDA programming model is employed in Chapter 5 where a GPU-accelerated linear Poisson-Boltzmann solver is presented.

2.2 Molecular Dynamics Simulations

Molecular Dynamics (MD) is a popular computational technique to simulate the dynamics of solvated protein-ligand complexes. As described in Section 2.3, it is often used for sampling procedures in AFE and end point methods such as MM/PBSA as well as QM/PBSA in Chapter 6 to sample the conformations of the end states. It involves the simulation of trajectories of particles in a system over a specified time frame[75] and the motion of particles is predicted by numerically integrating Newton's equations of motions where at each timestep, with coordinates, momentum, forces and potential energies being iteratively calculated.[76] In classical MD, used herein, atoms and bonds to be modelled as spheres and "springs", respectively, and therefore lack explicit treatment of the electronic structure unlike that of QM methods. A requirement of MD simulations are force fields (FFs), many of which are largely empirical. These are potential energy functions and associated parameters, which permit energies of configurations to be computed analytically and consequently, provides the forces acting on the particles for the simulation of their motion in the system.[77]

In these FF models, Coulomb's law and Lennard-Jones (LJ) potentials are employed to account for electrostatic interactions and dispersion/repulsion forces, respectively. Force fields generally vary in their geometric description of the molecules including equilibrium bond angle and distance, potential charge on the atoms, the number of sites, and the LJ parameters.[78] The potential energy associated with the bonds (E_{bond}), angles (E_{angle}) and dihedral angles ($E_{dihedral}$) are given by[79]

$$E_{bond} = \sum_{bonds} \frac{a_i}{2}(l_i - l_{i0})^2 \tag{2.1}$$

$$E_{angle} = \sum_{angles} \frac{b_i}{2}(\theta_i - \theta_{i0})^2 \tag{2.2}$$

$$E_{dihedral} = \sum_{dihedrals} \frac{c_i}{2}\left(1 + \cos\left(n\omega_i - \gamma_i\right)\right) \tag{2.3}$$

where l_i and θ_i are bond distances and angles, respectively. Those with a subscript of zero denote the equilibrium quantity. a_i and b_i correspond the force constants. The $E_{dihedral}$ term refers to rotations around covalent bonds and these have periodic energy

terms determined by n, with the rotational barriers provided by c_i. The formulation for $E_{dihedral}$ can vary depending on the FF. The sum of the Coulomb and LJ interactions gives the interaction energies between molecules, where the LJ (E_{lj}) and Coulomb (E_c) components of the energy are given below.[80] ϵ_{ij} and σ_{ij} correspond to the energetic and geometric Lennard-Jones parameters, respectively.

$$E_{lj} = \sum_{i>j} 4\epsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right) \tag{2.4}$$

$$E_c = \sum_{i>j} \frac{C q_i q_j}{r_{ij}} \tag{2.5}$$

where r_{ij} denotes the distance between atoms i and j.

Herein, only classical FFs are employed, however, advancements have been made to develop more accurate FFs, including polarisable[81–83] and machine learned interatomic potentials.[84]

In QM/PBSA herein (see Chapter 6) and MM/PBSA workflows the following MD runs are performed: heating, equilibrium and production. The heating step is performed in the *NVT* ensemble where the number of particles and volume are kept constant whereas the temperature fluctuates around an equilibrium value. On the other hand, the equilibrium and production runs are performed in the *NpT* ensemble which is similar to *NVT* but the volume is not held constant and the pressure fluctuates around an equilibrium value. In Chapters 5 and 6, classical MD simulations are performed on various protein-ligand structures. Chapter 5 uses the final structure from an equilibrium run since only the computational performance of the proposed PBE solver is of interest and one structure per protein-ligand complex is sufficient. In Chapter 6, conformations from the production run are extracted to compute gas phase interaction energies and solvation energies for the QM/PBSA workflow.

2.3 Computational Methods for Predicting Binding Affinities

In this section, an overview of three methods for the prediction of binding free energies of protein-ligand systems is presented. Firstly, a thermodynamic description of binding free energy is provided before detailing the current state-of-the-art computational models for its prediction. For the purposes of this work, binding affinity is synonymous for binding free energy.

Free energy is a fundamental thermodynamic concept that refers to the amount of energy available to perform work. It dictates the direction in which a thermodynamic process will occur, and importantly, determines whether a reaction will spontaneously

occur.[85]

In drug discovery, binding free energies are crucial quantities to consider as part of the hit-to-lead and lead optimisation phases in drug development.[86] Therein, potential 'hit' drug candidates are identified and undergo further optimisation to 'lead' candidates.[87] Binding free energies quantify how thermodynamically favourable the binding process of a drug candidate molecule to a target receptor is and serves as the preferred metric when comparing the utility of drug candidates for a specific target receptor.

Of particular interest are protein-ligand complex systems, specifically, the binding of a ligand to a protein structure, as the majority of receptors in the human body comprise a significant protein component.[88] Studying the binding behaviour of a smaller molecule (*i.e.* ligand) with a protein structure is key to understanding the binding of drug molecules to target receptors.

Under physiological conditions where pressure and temperature are considered constant, the binding free energy (ΔG_{bind}) of a protein-ligand complex is

$$\Delta G_{bind} = \Delta H - T\Delta S \tag{2.6}$$

where T is the temperature, and ΔH and ΔS denote the change in enthalpy and entropy, respectively, associated with the binding event.

For each protein target, virtual screening involves measuring its binding affinity with hundreds or thousands of potential drug candidates (*i.e.* ligands). Therefore whilst attaining accurate predictions of the binding free energy values is important, of greater importance is the ability to rank the relative binding strengths of the ligands against each other. Hence, binding affinity and binding free energy are treated as equivalent in this work. It is the correlation with experiment which provides more meaningful assessments of the accuracy of a computational method. Specifically, the Pearson (R) correlation coefficient has become one of the most commonly used correlation metrics to assess the accuracy of computational methods. Achieving high correlation between predicted results and experiment is the primary goal of computational techniques in drug discovery. In Chapter 6, R is used as the metric to assess performance of the QM/PBSA approach.

2.3.1 Molecular Docking

Molecular Docking has become a popular method for predicting binding affinities due to its low computational overhead.[48] As stated in Chapter 1, molecular docking begins by predicting the conformers of the bound protein-ligand structure and calculating the binding affinities of the latter. This is accomplished by first generating various binding conformations of the ligand bound to the binding site of the target protein system. Next, a scoring function is employed to approximate the protein-ligand affinity and rank the

binding poses accordingly.[89,90]

The quality of a molecular docking scheme is related to the scoring function employed.[91] These scoring functions can be largely categorised into three main groups: 1. force field-; 2. empirical-; and 3. knowledge-based. Force field-based scoring functions employ classical molecular mechanics energy functions and the interaction energy is approximated as the sum of bonded (intramolecular) and non-bonded (intermolecular) terms from a classical force field. Solvation can be accounted for through continuum models such as Poisson Boltzmann or Generalised Bohr/Surface Area approaches.[92]

In empirical-based functions, the binding free energy is represented as the weighted sum of various terms such as hydrogen bonding, hydrophobicity and solvation.[93] Therein, weights are optimised to reproduce the experimental binding affinities of a specific protein-ligand dataset.[94,95] Due to its simple formulation, empirical-based scoring functions are more computationally efficient than their force field-based counterparts.[96] However, the overall applicability of empirical-based functions is largely dependent on the diversity and quality of the dataset the weights were derived from.[94]

In knowledge-based scoring functions, the binding affinity is expressed as the sum of protein-ligand atom pair interactions.[93] These scoring functions utilise information from protein-ligand structures experimentally determined, namely, the protein-ligand atom pair contacts to calculate a probability density function for each atom pair in the protein-ligand complex being studied.[97] Across the three types of scoring functions, knowledge-based scoring functions aim to balance speed and accuracy by calculating interatomic pair energies from experimentally determined structures, rather than fitting to binding affinities as in empirical scoring functions.[98]

2.3.2 Alchemical Free Energy Methods

Alchemical free energy approaches are theoretically rigorous and accurate, and leverage the concept of free energy being a state function. Being a state function, only the beginning and final states are required; it is independent of the path taken to transition from one state to another. In the context of AFE approaches, states typically refer to different ligands with the same surroundings (*e.g.* bound to the same target protein or in an aqueous environment). For example, consider the alchemical transition from state B to D in Fig. 2.5 where the functional group changes from a hydrogen to an ethyl group. As discussed earlier, the role of computational methods in the drug development pipeline is to select "high quality" drug candidates, *i.e.* molecules which are predicted to have high binding affinities, by filtering out candidates which display low binding potential. Consequently, though predicting ΔG_{bind} is important, it is the relative binding free energy ($\Delta\Delta G_{bind}$), namely, the difference in the binding free ener-

gies between ligands, which is more relevant. Using Figure 2.5 as an illustration, whilst it is possible to attain $\Delta\Delta G_{bind}$ between two ligands by observing the transitions from the aqueous unbound state to the bound state (*i.e.* from A to B and C to D) which involves the alchemical transformation of many atoms in the protein (generally thousands), it is much more computationally efficient to observe the transitions where only the functional group of the ligand is modified (*e.g.* B to D or A to C in Figure 2.5). Thus, most applications of AFE methods involve the alchemical transition from one ligand to another.

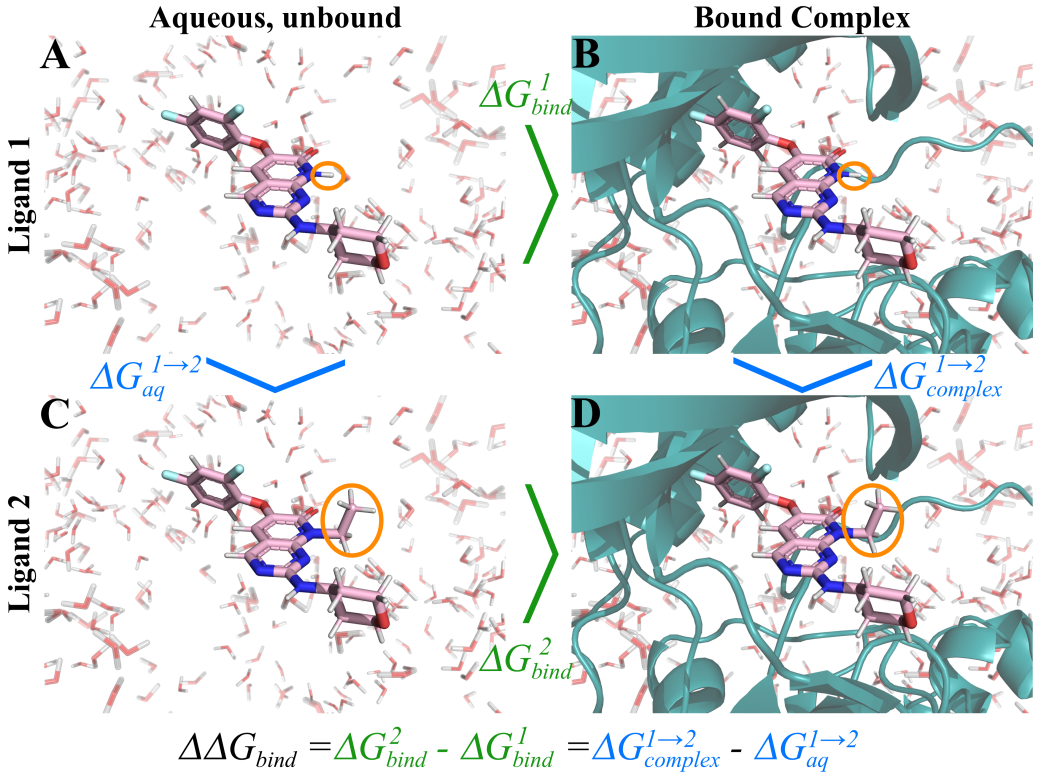


Figure 2.5: Thermodynamic cycle for calculating the relative binding free energy $\Delta\Delta G_{bind}$ for the transformation of ligand 1 to ligand 2. Functional groups circled in orange correspond to regions which undergo alchemical changes during the AFE procedure. Text and arrows in blue (green) represent the transition from ligand 1 (aqueous states) to ligand 2 (bound states).

Two popular AFE methods include free energy perturbation (FEP)[99] and thermodynamic integration (TI).[100,101] Both methods involve sampling intermediate states as the system transforms from an initial to a final state to compute relative free energies. This is typically indicated with a coupling parameter, λ, between the initial ($\lambda = 0$) and final ($\lambda = 1$) states. λ values between 0 and 1 noninclusive correspond to intermediate states. TI involves integrating the derivative of the Hamiltonian with respect to λ from $\lambda = 0$ to $\lambda = 1$.[4] Specifically, the relative free energy as the system transforms from the

initial to final state is given by:

$$\Delta G_{0\to 1} = \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_\lambda d\lambda \tag{2.7}$$

where $\langle \ldots \rangle_\lambda$ denotes an ensemble average at a particular λ. Typically, this integration in Eq. (2.7) is performed numerically and the relative free energy is computed as the weighted sum of the Hamiltonian derivative at various λ values between 0 and 1. At each of these λ windows, sampling is performed often with Molecular Dynamics (MD) to compute the ensemble average of the Hamiltonian derivative.

Alternatively, in FEP, the change in free energy from state i to j can be computed using Zwanzig's equation[99]

$$\Delta G_{i\to j} = -k_B T \ln \left\langle \exp\Big(-\frac{1}{k_B T}(H_j - H_i) \Big) \right\rangle_j \tag{2.8}$$

where k_B is Boltzmann's constant, T is temperature, and H_i and H_j denote the Hamiltonian at states i and j, respectively. Beyond Zwanzig's equation, the Bennett Acceptance Ratio (BAR)[102] is also widely used in FEP methods. Therein, the free energy difference is calculated using the Fermi function f in the following relation

$$\Delta G_{i\to j} = k_B T \ln \left(\frac{\langle f(H_j - H_i + C) \rangle_i}{\langle f(H_i - H_j + C) \rangle_j} \right) + C, \quad f(x) = \frac{1}{1 + \exp(\frac{x}{k_B T})} \tag{2.9}$$

where C is a constant and is determined iteratively to satisfy the following equation[103]

$$\langle f(H_j - H_i + C) \rangle = \langle f(H_i - H_j + C) \rangle \tag{2.10}$$

Although both Eq. (2.8) and Eq. (2.9) may suggest that only two states need to be sampled (*i.e.* by setting $i = 0$ and $j = 1$) to attain the relative free energies between the end states, similar to TI which includes integrating at various intermediate states between $\lambda = 0$ and $\lambda = 1$, sampling of intermediate states is also necessary in FEP to allow a gradual transition from the initial to the final state.[104] This is to ensure sufficient overlap in the phase space between states, otherwise configurations sampled in one state will be high energy in another state and contribute little to the exponential average terms in Eq. (2.8) and Eq. (2.9),[105] which is usually the case in biomolecular simulations relevant for drug discovery.[47] To handle multiple states, the BAR scheme has been extended to the Multistate Bennett Acceptance Ratio (MBAR) method.[106]

AFE methods have demonstrated high predictive ability in a broad range of protein-ligand complexes. Perhaps one of the most notable FEP studies is by Wang *et al.* where R values between 0.71 and 0.89 and MUEs of less than 1.16 kcal mol^{-1} were observed across six protein targets spanning 149 protein-ligand complexes.[55] Since its publica-

tion, this study has served as the benchmark for other AFE studies on the same target systems as that examined by Wang *et al*.[18,107,108] TI methods have also been shown to exhibit similar accuracies as that of FEP. For instance, in a study employing various AFE software, the average R values varied between 0.91 and 0.95 for TI and 0.93 and 0.96 for FEP, with corresponding $\Delta\Delta G_{bind}$ MUEs of 0.41 to 0.51 and 0.42 to 0.51 kcal mol^{-1}, respectively.[108] Overall, FEP and TI applications on protein-ligand studies have regularly demonstrated high correlations with experiment and low $\Delta\Delta G_{bind}$ errors.[18,55,107–112]

2.3.3 Molecular Mechanics / Poisson-Boltzmann Surface Area

In the Molecular Mechanics / Poisson-Boltzmann Surface Area (MM/PBSA) method, the free energy of binding between a protein (P) and ligand (L) to form a complex (PL) is approximated as

$$\Delta G_{bind} = \langle G_{PL} \rangle - \langle G_P \rangle - \langle G_L \rangle \tag{2.11}$$

where G_{PL} is the free energy of the protein-ligand complex whilst G_P and G_L are the free energies of the unbound protein and ligand, respectively. $\langle \ldots \rangle$ denotes an ensemble average over conformations obtained from a sampling procedure, typically molecular dynamics (MD) simulations with explicit solvent. As implied from Eq. (2.11), MM/PBSA methods are only concerned with the end states, namely, the unbound protein, unbound ligand and the bound protein-ligand complex, contrasting AFE methods where sampling of intermediate states are required.

Equation (2.11) is further decomposed as

$$\Delta G_{bind} = \langle \Delta E_{MM} \rangle + \langle \Delta G_{solv} \rangle - \langle T\Delta S \rangle \tag{2.12}$$

Eq. (2.12) bears similarity to Eq. (2.6), where the enthalpy term of Eq. (2.6) is decomposed into two terms: the gas phase interaction energy ΔE_{MM} and the solvation free energy ΔG_{solv}. The formulation of ΔE_{MM} is provided in Section 1.2. On the other hand, the solvation free energy term is represented as the sum of the polar and non-polar contributions

$$\Delta G_{solv} = \Delta G_{solv}^{\ polar} + \Delta G_{solv}^{\ non\ polar} \tag{2.13}$$

here, the polar contribution is attained with the implicit Poisson-Boltzmann solvation model whereas the non-polar contribution is evaluated as a linear function of the solvent accessible surface area $(SASA)$ of the molecular structure. Specifically,

$$G_{solv}^{\ non\ polar} = \gamma \cdot SASA + \beta \tag{2.14}$$

where γ is a surface tension constant and β is a correction constant.

As depicted in Fig. 2.6 in the dotted black line, both the solvent accessible surface (SAS) and the solvent excluded surface (SES) are generated by tracing a solvent probe sphere across the surface defined by the union of van der Waal spheres of all atoms in the molecule. Specifically, the SAS is formed by following the center of the probe sphere, whereas the SES is formed by following its outer edge. $SASA$ used in Eq. (2.14) is simply the area of the SAS.

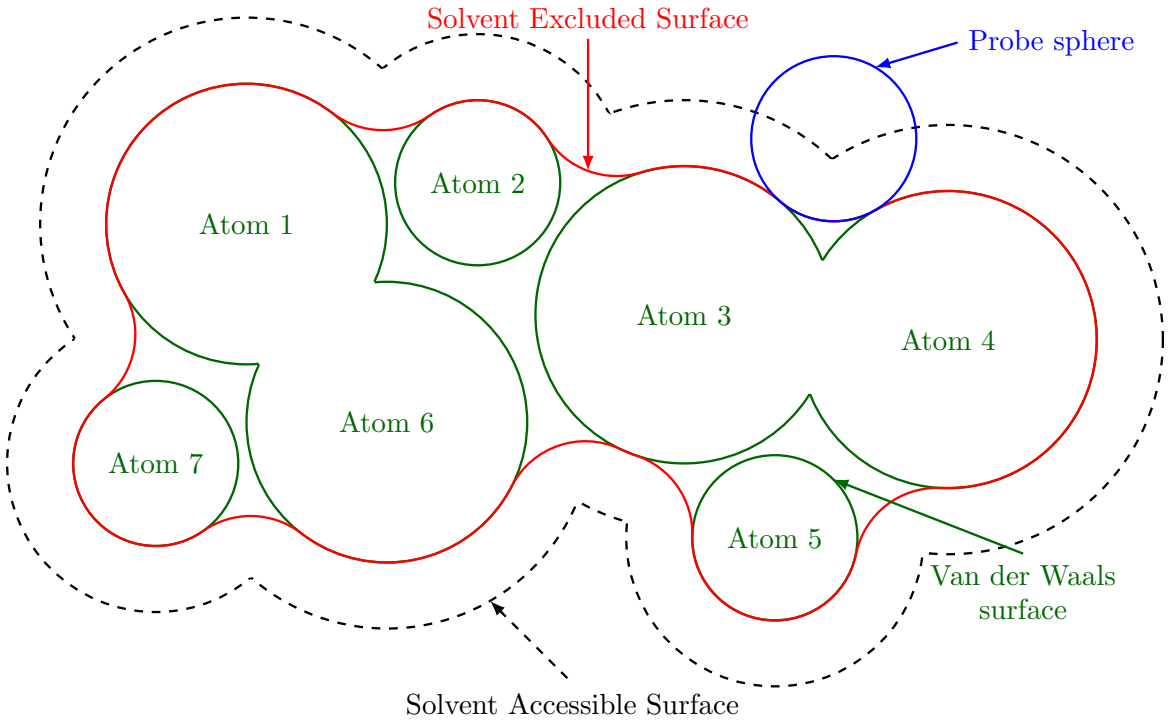


Figure 2.6: 2D diagram illustrating the solvent excluded and solvent accessible surface areas. Both surfaces are obtained by tracing the probe sphere along the surface of the van der Waals surface.

Lastly, ΔS of Eq. (2.12) represents the change in conformational entropy from the binding event and comprises the translational, rotational and vibrational contributions. In many applications of MM/PBSA, the entropy term is often neglected for two main reasons. First, vibrational entropy is typically treated with Normal Mode Analysis at the MM level, which is a computationally expensive routine as this involves the diagonalisation of the Hessian matrix.[113] Second, for congeneric ligands—compounds that share the same core structure with variations in its substituent groups and are typically of interest—binding to the same target protein, generally only the relative binding energies or correlation with experiment are of interest. Consequently, the entropic contributions are assumed to be similar owing to their structural similarity.[114] In this work, namely, in Chapter 6, only congeneric ligands are studied and therefore entropic contributions are not considered herein.

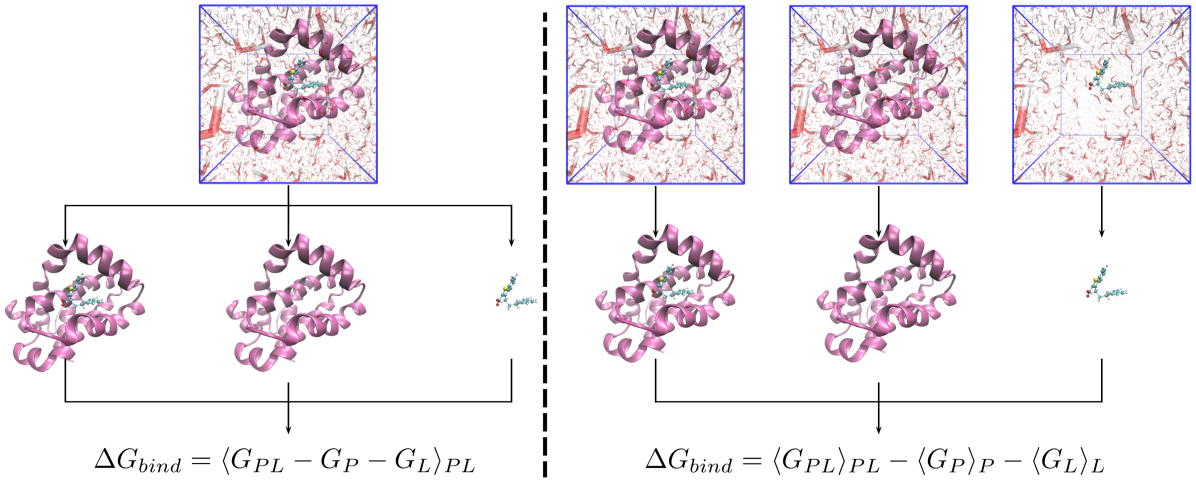


Figure 2.7: Workflow for the *left* 1A-MM/PBSA protocol and the *right* 3A-MM/PBSA protocol. Molecule in pink (cyan) represents the protein (ligand). The first step involves performing molecular dynamics simulations with explicit solvent, after which the solvent molecules are stripped to obtain the conformations of the bound protein-ligand complex and unbound protein and ligand.

The general workflow of the MM/PBSA procedure is as follows. 1. MD simulation(s) are performed in explicit solvent to generate conformers of the protein and ligand structures. 2. All solvent molecules are stripped from the MD trajectory and an implicit solvation model (*e.g.* PB) is used to compute the solvation energy, whilst the interaction energy is computed with MM. Step 1 comprises either one or three separate simulations, these differ in how the conformers of the unbound protein and ligand are obtained. As illustrated in Fig. 2.7, the one-average (1A) scheme includes a single simulation of the protein–ligand complex, from which conformations of the bound complex as well as the unbound protein and ligand are extracted. In contrast, the three-average (3A) scheme employs three separate simulations: one for the bound complex and one each for the unbound protein and ligand. A two-average (2A) approach has also been suggested which includes an additional simulation of the unbound ligand to capture the ligand reorganisation energy.[115] However, it remains seldom employed relative to either of 1A or 3A. The 1A-MM/PBSA approach is more widely applied than its 3A counterpart as it is computationally cheaper involving only one MD simulation. A consequence of this is that internal energies (ΔE_{int} in Eq. (1.3)) cancel out, and only the non-bonding terms need to be computed. Furthermore, the 1A method benefits from error cancellation which reduces the effect of incomplete sampling, and tends to exhibit better accuracy and precision than the 3A approach.[114,116–118] However, it is predicated on the assumption that the structures of the isolated protein and ligand are sufficiently similar to that of the bound state, which may not be valid for systems that exhibit large conformational changes upon binding.

No MM/PBSA workflows are performed in this work, however, its QM-based counterpart, QM/PBSA, is performed whose workflow bears similarity to that of MM/PBSA. In particular, a 1A-QM/PBSA approach is adopted in Chapter 6.

2.4 Quantum Mechanical Methods

Quantum mechanical (QM) methods are used repeatedly in this work (Chapters 3, 4 and 6). This section provides an overview of QM theory and the QM methods used herein.

The central problem in quantum mechanical methods is solving the non-relativistic time-independent Schrödinger equation of a chemical system:

$$\hat{H}|\psi\rangle = E|\psi\rangle \tag{2.15}$$

where E is the energy of the system, $|\psi\rangle$ is time-independent wave function, henceforth, called the wave function, and $\hat{H}$ is the Hamiltonian operator and can be further decomposed as

$$\hat{H} = \hat{T}_e + \hat{T}_n + \hat{V}_{ne} + \hat{V}_{ee} + \hat{V}_{nn} \tag{2.16}$$

where $\hat{T}_e$ denotes the electronic kinetic operator, $\hat{T}_n$ represents the nuclear kinetic operator, $\hat{V}_{ne}$ accounts for the nucleus-electron attraction, $\hat{V}_{ee}$ represents the electron-electron repulsion and $\hat{V}_{nn}$ captures the nucleus-nucleus repulsion.

Eq. (2.15) can only be solved analytically for one-electron systems such as a hydrogen atom or helium cation due factors such as the many body problem.[8,119] To solve Eq. (2.15), a number of approximations is made. The first of which is the Born- Oppenheimer approximation[120] where the nuclear and electron components are separated. This approximation uses the idea that nuclei are much heavier and display little quantum effects whilst electrons are very light particles whose velocities are significantly higher than that of the nuclei. Therefore, the motion of the electrons and nuclei can be decoupled and Eq. (2.15) can be recasted as simply considering only the motion of the electrons. This gives the electronic Schrödinger equation:

$$\hat{H}_e|\psi_e\rangle = E_e|\psi_e\rangle \tag{2.17}$$

where the electronic Hamiltonian $\hat{H}_e$ considers only the electronic components of $\hat{H}$:

$$\hat{H}_e = \hat{T}_e + \hat{V}_{ne} + \hat{V}_{ee} \tag{2.18}$$

With the motion of the nuclei considered stationary with respect to that of the electrons, the nuclear kinetic energy is set to zero and the nuclear-nuclear repulsion is considered

a constant.

Solving Eq. (2.17) is the main objective of many quantum mechanical methods and provides the eigenvalue (*i.e.* electronic energy) and eigenfunctions of the chemical system which describe its electronic state. All mentions of energy attained from QM methods refer to the electronic energy. All mentions of the Schrödinger equation henceforth refer to Eq. (2.17) and $_e$ subscripts will be removed.

Thus far, the Hamiltonian in Eq. (2.17) considers only the spatial coordinates (r) of the electrons, however, the spin (ω) must also be included to fully characterise an electron. Therefore, an N-electron system is represented by a wave function $\Psi(x_1, x_2, \ldots, x_N)$ where $x_i = \{r_i, \omega_i\}$.

The wave function Ψ must also satisfy the anti-symmetry property:

$$\Psi(x_1, x_2, \ldots, x_i, x_j, \ldots, x_N) = -\Psi(x_1, x_2, \ldots, x_j, x_i, \ldots, x_N) \tag{2.19}$$

where its sign flips from the exchange of two electrons, x_i and x_j.

Each electron occupies its own single-electron wave function also termed an orbital. Each orbital is represented as a spin orbital χ_i that characterises both the spatial and spin component of an electron.

A simple approximation (*i.e.* Hartree Approximation) of the N-electron wave function is represented as product of spin orbitals,[121] specifically:

$$\Psi(x_1, x_2, \ldots, x_N) = \chi_1(x_1)\chi_2(x_2)\ldots\chi_N(x_N) \tag{2.20}$$

However, Eq (2.20) does not satisfy the antisymmetry principle. The anti-symmetry principle can be satisfied by representing Ψ with a Slater determinant ψ.[122] For a N-electron system, ψ takes on the following form

$$\psi = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(x_1) & \chi_2(x_1) & \ldots & \chi_N(x_1) \\ \chi_1(x_2) & \chi_2(x_2) & \ldots & \chi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(x_N) & \chi_2(x_N) & \ldots & \chi_N(x_N) \end{vmatrix} \tag{2.21}$$

In Eq. (2.21), each row and column describes an electron and spin orbital, respectively. Therefore, exchanging two electrons corresponds to exchanging two rows, flipping the sign of the Slater determinant and fulfilling the anti-symmetry requirement.

To determine the wavefunction, the orbitals χ_i must be attained first. This is accomplished by the variational principle, which states that the best set of orbitals are those which give the lowest energy, *i.e.* the expected value of the electronic Hamiltonian operator $\hat{H}$

$$E_0 = \langle\psi|\hat{H}|\psi\rangle \tag{2.22}$$

By minimising the energy with respect to the orbitals, a set of equation can be attained of the following form

$$f(i)\chi(x_i) = \varepsilon_i \chi(x_i) \tag{2.23}$$

These are the Hartree-Fock (HF) equations. An overview of the HF method is provided in the following section.

2.4.1 Hartree-Fock Theory

In Eq. (2.23), $f(i)$ is called the Fock operator and has the following form

$$f(i) = -\frac{1}{2}\nabla_i^2 - \sum_{A=1}^{M} \frac{Z_A}{r_{iA}} + \nu^{HF}(i) \tag{2.24}$$

The first term is the kinetic operator, the second term accounts for the nuclear-electron attraction where Z_A is the atomic number of nucleus A and r_{iA} is the distance between electron i and nucleus A. $\nu^{HF}(i)$ is the average potential (mean field) experienced by electron i due to the presence of the other $N-1$ electrons and is defined as

$$\nu^{HF}(i) = \sum_j (\hat{J}_j - \hat{K}_j) \tag{2.25}$$

where $\hat{J}$ is the Coulomb operator that captures the classical electrostatic interactions between electrons whose elements are given by

$$J_{\mu\nu} = \sum_{\lambda\sigma} D_{\lambda\sigma}(\mu\nu|\lambda\sigma) \tag{2.26}$$

whereas $\hat{K}$ is the exchange operator and is a result of the antisymmetry of the wave-function[123] and its elements are given by

$$K_{\mu\nu} = \frac{1}{2}\sum_{\lambda\sigma} D_{\lambda\sigma}(\mu\lambda|\nu\sigma) \tag{2.27}$$

where $(\mu\lambda|\nu\sigma)$ is four-center two-electron integrals or otherwise known as electron repulsion integrals (ERIs)[124] and defined as

$$(\mu\lambda|\nu\sigma) = \int d\boldsymbol{r}_1 d\boldsymbol{r}_2 \phi_\mu^*(\boldsymbol{r}_1)\phi_\lambda(\boldsymbol{r}_1) r_{12}^{-1} \phi_\nu^*(\boldsymbol{r}_2)\phi_\sigma(\boldsymbol{r}_2) \tag{2.28}$$

$\nu^{HF}(i)$ depends on the spin orbitals and therefore so too does the Fock operator, therefore, the HF equations must be solved iteratively and this is accomplished with the Self-Consistent Field (SCF) method.[8]

Self-Consistent Field Procedure

In this section, we focus on the SCF procedure for the restricted Hartree-Fock (RHF) problem. RHF is a formulation of HF suitable for closed-shell systems where electrons are paired. Here, the eigenvalue problem is[125,126]

$$\boldsymbol{FC} = \boldsymbol{SC}\boldsymbol{\epsilon} \tag{2.29}$$

where $\boldsymbol{F}$ is the Fock matrix, $\boldsymbol{C}$ is the molecular orbital (MO) coefficient matrix which contains the unknown coefficients of MOs. Eq. (2.29) is known as the matrix form of the Roothan-Hall equations. A molecular orbital ψ_i in RHF is represented as a linear combination of basis functions (*e.g.* atomic basis functions) $\{\phi_\mu\}$

$$\psi_i = \sum_{\mu=1}^{N} C_{\mu i}\phi_\mu \tag{2.30}$$

where $C_{\mu i}$ are the coefficients that describe the weighting of each basis function ϕ_i to the MO and taken together make up the coefficient matrix $\boldsymbol{C}$. The expansion in Eq. (2.30) is exact if the set of basis functions $\{\phi_\mu\}$ is complete (*i.e.* infinite). Given the infeasibility of a complete set, the choice of basis sets is an important factor when performing QM calculations.

$\boldsymbol{S}$ of Eq. (2.29) is the overlap matrix that accounts for the non-orthogonality of basis functions used to represent AOs, and $\boldsymbol{\epsilon}$ is a diagonal matrix containing MO energies.

As described earlier, the HF equations must be solved iteratively and this is achieved with the SCF procedure. The general steps of an SCF procedure are outlined as follows

1. **Initial guess** An initial guess of the molecular orbitals or density matrix ($\boldsymbol{D}$). The density matrix is formed from the MOs and are used to construct $\boldsymbol{F}$. In Chapter 4, various initial guess approaches are explored to accelerate SCF calculations.

2. **Fock matrix construction** Construct the Fock matrix using the components of the Fock operator including one-electron kinetic and nuclear attraction operators and two-electron Coulomb and exchange operators.

3. **Diagonalise Fock matrix** Diagonalise the Fock matrix to attain a new set of MOs and energies.

4. **Convergence check** With the new set of MOs and energies, substitute into Eq. (2.29) and check for convergence. Repeat steps 2 to 4 until convergence is achieved. Convergence is achieved if the difference in a metric between successive iterations is less than a specific threshold. In this work, the metric utilised is the L1 norm of the commutator ($\boldsymbol{FDS} - \boldsymbol{SDF}$). Although metric is sometimes referred to as the CDIIS error in the literature,[127] it is referred to herein simply as the DIIS error.

On the other hand, in the unrestricted HF (UHF) method which is generally used for open-shell systems without paired electrons (*e.g.* triplet states in Chapter 4), the two possible spin states α and β are treated separately. In the RHF method, where the electrons are paired, the density matrix is given by

$$D_{\mu\nu} = 2\sum_{i}^{N_e/2} C_{\mu i}C_{\nu i} \tag{2.31}$$

where N_e is the number of electrons. In contrast for UHF, the α and β electrons are treated separately and the corresponding density matrices have elements of

$$D^{\alpha}_{\mu\nu} = \sum_{i}^{N_\alpha} C^{\alpha}_{\mu i}C^{\alpha}_{\nu i} \tag{2.32}$$

$$D^{\beta}_{\mu\nu} = \sum_{i}^{N_\beta} C^{\beta}_{\mu i}C^{\beta}_{\nu i} \tag{2.33}$$

and the total electron density matrix is the sum of $\boldsymbol{D}^{\alpha}$ and $\boldsymbol{D}^{\beta}$.

Furthermore, the unrestricted alpha or beta Fock matrix differs to the restricted one only in the exchange component (see Eq. (2.27)). Specifically, in UHF the alpha exchange matrix is given by

$$K^{\alpha}_{\mu\nu} = \sum_{\lambda}^{N}\sum_{\sigma}^{N} D^{\alpha}_{\lambda\sigma}(\mu\lambda|\nu\sigma) \tag{2.34}$$

UHF calculations are only performed in Chapter 4 on triplet electronic states. All other mentions of HF refer to the RHF method.

Although the HF method greatly simplifies solving the Schrödinger equation and attaining the energy and wavefunction of a chemical system, it does not explicitly account for electron correlation due to its mean field treatment. The correlation energy (E_{corr}) is defined as

$$E_{corr} = E_{exact} - E_{HF} \tag{2.35}$$

where E_{exact} is the exact electronic energy and E_{HF} is the energy calculated with the HF method. The accurate treatment of the correlation energy is essential to calculate interaction energies[128] and by extension critical for the computation of binding free energies.

In the following section, an overview of the second-order Møller–Plesset perturbation theory, an accurate method for calculating electron correlation, is provided.

2.4.2 Second-Order Møller–Plesset Perturbation Theory

In MP2, the idea is to capture electron correlation by double excitations, *i.e.* the contributions between pairs of electrons.[129] MP2 is part of a larger family of methods, namely, perturbation methods.[130] Therein, the electronic Hamiltonian takes the Fock operator from HF theory and adds a perturbation to it:

$$\hat{H} = \hat{H}_0 + \lambda V \tag{2.36}$$

where V is a small perturbation and λ is a dimensionless parameter.

The energy and wave function can be expanded in a hierarchical manner with respect to the perturbation parameter λ

$$E_n = E_0 + \lambda E_1 + \lambda^2 E_2 + \ldots + \lambda^n E_n \tag{2.37}$$

$$\Psi_n = \Psi_0 + \lambda \Psi_1 + \lambda^2 \Psi_2 + \ldots + \lambda^n \Psi_n \tag{2.38}$$

The HF energy is the sum of the zero and first order energies

$$E_{HF} = E_0 + E_1 \tag{2.39}$$

and the correlation energy can then be expressed as the following sum

$$E_{corr} = E_2 + E_3 + E_4 + \ldots \tag{2.40}$$

of which the first term is the MP2 energy.

The MP2 energy is given as

$$E_{MP2} = -\frac{1}{4} \sum_{a<b}^{virt} \sum_{i<j}^{occ} \frac{|\langle ab||ij\rangle|^2}{\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j} \tag{2.41}$$

where a, b denote virtual MOs, i, j denote occupied MOs, ϵ_i is the energy of MO i, and the numerator is equivalent to

$$|\langle ab||ij\rangle|^2 = (ij|ab) - (ij|ba) \tag{2.42}$$

where $(ij|ab)$ is a two electron repulsion integral (see Eq. (2.28)).

The MP2 method has become a widely used approach for calculating electron correlation due to it being one of the most computationally accessible post-HF method and recovering between 80% to 90% of correlation.[129] The MP2 method is used in Chapter 6 when calculating the gas phase interaction energies of protein-ligand systems.

2.4.3 Resolution-of-the-Identity

The Resolution-of-the-Identity (RI) approximation[131] of HF and/or MP2 is used across multiple chapters in this work (Chapters 4 and 6). This section provides an overview of the RI method.

One of the major bottlenecks of HF and MP2 computations is the evaluation and digestion of ERIs. Storing these four-center ERIs requires an impractically high amount of storage and instead they are typically recomputed at every SCF iteration.

Under the RI approximation, these four-center ERIs are approximated with a sequence of matrix multiplications of two- and three-center integrals:[132]

$$(ab|cd) \approx (ab|cd)_{RI} = \sum_{PQ} (ab|P) J^{-1}_{PQ} (Q|cd) \tag{2.43}$$

where a, b, c and d denote primary atomic orbital basis functions, P and Q denote auxiliary basis functions, and J_{PQ} is the auxiliary basis Coulomb inner-product tensor and is a two-center integral.

Due to their lower spatial complexity, the two- and three-center integrals can be computed once and stored in memory, avoiding the repeated calculation of four-center integrals. Furthermore, this RI approximation reduces the pre-factor of the Fock matrix construction step since the matrix multiplications can be performed at near-peak floating point performance and utilise the computational hardware more efficiently.[133]

In the GPU-accelerated Extreme-scale Electronic Structure System (EXESS) quantum chemistry program,[26] these two- and three-center integrals are stored in the GPU memory to avoid slow CPU-GPU data transfers. However, because the two- and three-center integrals are stored in memory, the system size in which RI-HF and RI-MP2 can be applied to is limited. Therefore, the RI approximation is only used in fragmented-based calculations in this work.

In this work, the RI approximation of HF (RI-HF) is employed in Chapter 4 to explore accelerating fragmentation-based initial guesses. On the other hand, RI approximations of both HF (RI-HF) and MP2 (RI-MP2) are used in fragmentation-based interaction energy calculations of Chapter 6.

2.4.4 Density Functional Theory

Thus far, the QM methods described have been limited to wave function-based approaches. Another class of QM approaches are density functional theory (DFT) methods. Unlike HF and MP2 which are wave function-based methods, and thus are functions of the coordinates of all particles in the system, DFT is based on the electron density (ρ_{elec}), which is dependent on the three coordinates in Cartesian space only.[134]

In "pure" DFT methods, also known as orbital-free DFT, the energy of interacting

electrons is computed as a functional of the density.[135] Though this is considered a direct and correct approach, such methods typically exhibit low accuracies in practice due to the lack of accurate approximations of the kinetic energy functional for interacting systems.

To overcome this, Kohn and Sham proposed an alternative approach of dealing with a reference non interacting system instead whose kinetic energy can be exactly computed.[136] This Kohn-Sham approach enabled the practical application of DFT methods. All mentions of DFT henceforth refer to Kohn-Sham DFT.

In DFT methods, it is the Kohn-Sham (KS) equations which are solved[136]

$$\left(-\frac{1}{2}\nabla^2 + \nu^{eff}(\boldsymbol{r})\right)\phi_i = \epsilon_i\phi_i \tag{2.44}$$

where the first term $(-\frac{1}{2}\nabla^2)$ is the kinetic energy operator and the effective potential operator $\nu^{eff}(\boldsymbol{r})$ is given by

$$\nu^{eff}(\boldsymbol{r}) = \nu_{ne}(\boldsymbol{r}) + \nu_H(\boldsymbol{r}) + \nu_{XC}(\boldsymbol{r}) \tag{2.45}$$

$\nu_{ne}(\boldsymbol{r})$ accounts for the nuclear-electron attraction, $\nu_H(\boldsymbol{r})$ is the Hartree or Coulomb potential and $\nu_{XC}(\boldsymbol{r})$ is the exchange-correlation potential which is not formally known. Different DFT methods have different approximations of the $\nu_{XC}(\boldsymbol{r})$ term.

Level	Name	Variables
1	LDA	ρ
2	GGA	ρ, $\nabla\rho$
3	meta GGA	ρ, $\nabla\rho$, $\nabla^2\rho$ or τ
4	hybrid GGA	ρ, $\nabla\rho$, $\nabla^2\rho$ or τ, HF exchange

Table 2.1: Classification of exchange correlation functionals. Higher accuracy is indicated by higher level. LDA - local density approximation; GGA - generalised gradient approximation. τ denotes orbital kinetic energy density.

DFT methods fall into various classes of accuracy as shown in Table 2.1 depending on which variables it accounts for such as the density (ρ), its derivative $(\nabla\rho)$ or its second derivative $(\nabla^2\rho)$. In Chapter 4, two DFT functionals, MN15[137] (meta hybrid GGA) and B3LYP[138–141] (hybrid GGA) are employed.

2.4.5 Molecular Fragmentation

Molecular fragmentation is a technique that involves dividing the molecular system into smaller partitions by breaking bonds and is used across multiple chapters in this work, namely, chapters 3, 4 and 6. This section provides an overview of this concept.

Molecular fragmentation is an effective strategy for tackling scalability and parallelisation issues in QM modelling. This suite of methodologies is based on the premise that chemical interactions are sufficiently localised, allowing a chemical system to be divided into smaller segments known as monomers. To approximate the energy of the entire, unfragmented system, fragmentation approaches incrementally include the effects of larger fragments. These fragments, which encompass interactions among monomers, range from dimers and trimers to larger n-mers.

In fragmentation methods based on the Many Body Expansion (MBE)[142], the energy of the system is obtained as the following sum over fragments

$$E_{MBE} = \sum_{I} E_I + \sum_{I<J} \Delta E_{IJ} + \sum_{I<J<K} \Delta E_{IJK} + ... \tag{2.46}$$

where E_I is the energy of monomer I, ΔE_{IJ} and ΔE_{IJK} are dimer and trimer energy corrections defined as follows

$$\Delta E_{IJ} = E_{IJ} - E_I - E_J, \tag{2.47}$$

$$\begin{aligned} \Delta E_{IJK} =& E_{IJK} - \Delta E_{IJ} - \Delta E_{IK} - \Delta E_{JK} \\ & - E_I - E_J - E_K \end{aligned} \tag{2.48}$$

where E_{IJ} is the energy of a dimer system obtained as the union of monomers I and J, and E_{IJK} is the energy of a trimer system obtained as the union of monomers I, J, K.

The calculations of two-body and higher order terms (ΔE_{IJ}, ΔE_{IJK}, *etc.*) are only performed on fragments that are spatially close together, yielding an asymptotic scaling of $\mathcal{O}(N)$ with system size.[143,144]

The hierarchical nature of the MBE allows it to approximate the total energy to greater accuracy through the systematic inclusion of higher order terms.[145] In addition, the energy calculations of the many body fragments (monomers, dimers, *etc.*) can be performed independently, thereby exposing significant opportunities for exploiting large scale parallelism.[144] In Chapters 3 and 6, MBE is performed up to the two-body/dimer (MBE2) and the three-body/trimer (MBE3) levels.

2.5 Poisson-Boltzmann Surface Area Model

The QM techniques presented in Section 2.4 are used in this work to calculate the gas phase energies. In QM/PBSA workflows, in addition to the gas phase energy component, the solvation energy is also required. This section presents an overview of the PBSA solvation model.

The Poisson-Boltzmann Surface Area (PBSA) model is commonly employed to compute the solvation free energy (G_{solv}) of a molecular system. In particular, the polar and

non-polar contributions of G_{solv} are treated separately:

$$G_{solv} = G_{solv}^{polar} + G_{solv}^{non\ polar} \tag{2.49}$$

where G_{solv}^{polar} and $G_{solv}^{non\ polar}$ represent the polar and non-polar contributions to G_{solv}, respectively.

The computation of $G_{solv}^{non\ polar}$ is relatively simple and been presented in Section 2.3.3 (see Eq. (2.14)).

The evaluation of G_{solv}^{polar} is accomplished by solving the Poisson–Boltzmann equation (PBE). As discussed in Section 1.1 and later demonstrated in Chapter 5, current PBSA software are much too slow to handle the demands for large scale virtual screening. Thus, improving computational performance and efficiency in solving the PBE is essential for expanding the applicability of QM/PBSA methods or other workflows with PBSA integrated in computational drug discovery.

Under physiological ionic strengths relevant to protein-ligand binding, the linearised form of the PBE is employed:[59,146,147]

$$\nabla \cdot \varepsilon(\mathbf{r}) \nabla \varphi(\mathbf{r}) - \varepsilon_{sol} \kappa^2 \varphi(\mathbf{r}) = -4\pi\rho(\mathbf{r}) \tag{2.50}$$

where $\mathbf{r}$ represents solute positions, ρ is the solute charge density, φ is the electrostatic potential and ε is the dielectric distribution function. Here, $\kappa^2 = \frac{8\pi e^2 I}{\varepsilon_{sol} k_B T}$ is the modified Debye–Hückel parameter where I is the ionic strength and ε_{sol} is the solvent dielectric constant.

For molecular systems of varying and arbitrary shapes, the linear Poisson-Boltzmann equation Eq. (2.50) can only be solved numerically. In particular, Eq. (2.50) is typically treated with finite difference methods and forms a system of linear equations, given by[12,148]

$$\begin{aligned}
\frac{q(i,j,k)}{h} = \quad & \\
& \varepsilon_x(i-1,j,k)[\varphi(i,j,k) - \varphi(i-1,j,k)] \\
& + \varepsilon_x(i,j,k)[\varphi(i,j,k) - \varphi(i+1,j,k)] \\
& + \varepsilon_y(i,j-1,k)[\varphi(i,j,k) - \varphi(i,j-1,k)] \\
& + \varepsilon_y(i,j,k)[\varphi(i,j,k) - \varphi(i,j+1,k)] \\
& + \varepsilon_z(i,j,k-1)[\varphi(i,j,k) - \varphi(i,j,k-1)] \\
& + \varepsilon_z(i,j,k)[\varphi(i,j,k) - \varphi(i,j,k+1)] \\
& + h^2 \varepsilon_{sol} \kappa^2(i,j,k) \varphi(i,j,k)
\end{aligned} \tag{2.51}$$

where h is the grid spacing, $i,\ j$ and k index the grids along the x, y and z axes, respec-

tively. The terms $\varepsilon_x(i,j,k)$, $\varepsilon_y(i,j,k)$ and $\varepsilon_z(i,j,k)$ are the dielectric constants between the grid points connecting (i,j,k) to $(i+1,j,k)$, $(i,j+1,k)$ and $(i,j,k+1)$, respectively. The terms $\kappa^2(i,j,k)$, $\varphi(i,j,k)$, and $q(i,j,k)$ denote the Debye–Hückel parameter, the electrostatic potential and charge at grid point (i,j,k), respectively.

Equation (2.51) is applied to an $N_x \times N_y \times N_z$ point lattice, where N_d denotes the number of grid points in the d direction. The result is a system of $N_{grid} = N_x \times N_y \times N_z$ equations with N_{grid} unknowns (φ) to be solved at the grid points.

Conveniently, Eq. (2.51) can be expressed in the following matrix form

$$\boldsymbol{A}\mathbf{x} = \mathbf{b} \tag{2.52}$$

where $\boldsymbol{A}$ is a coefficient matrix comprising the dielectric constants (ε) and the Boltzmann term $(h^2\varepsilon_{sol}\kappa^2)$, $\mathbf{x}$ contains the electrostatic potential (φ) and $\mathbf{b}$ is the constant vector of charges (q) at the grid points.

A range of different solvers have been employed to solve Eq. (2.52) including multigrid and relaxation methods and conjugate gradient.[40] In Chapter 5, a Jacobi preconditioned conjugate gradient solver is employed to solve the linear PBE numerically.

2.6 Artificial Intelligence Optimisers

In Chapter 3, the artificial intelligence (AI) optimisation approaches Bayesian Optimisation (BO) and Genetic Algorithm (GA) are utilised to develop the automated fragmentation method proposed therein. This section provides an overview of the two approaches.

2.6.1 Bayesian Optimisation

In Chapter 3, Bayesian optimisation (BO) is utilised to optimise the weights for the scoring function (Eq. (3.3)). BO is an optimisation approach that focuses on solving the problem:

$$\min_{x \in A} f(x) \tag{2.53}$$

where f is a continuous, expensive objective function and x is the input vector. Of course, this can also be reframed as a maximisation problem in which case the problem appears as:

$$\max_{x \in A} -f(-x) \tag{2.54}$$

In the context of Chapter 3, x contains the weights for each of the terms in the scoring function Eq. (3.3).

BO builds a surrogate model for the objective function using previously sampled points (*i.e.* $x_1, x_2, \ldots, x_k$). Typically a Gaussian Process $(\mathcal{GP})$ is used to model the

surrogate model. A mean function $\mu_0(x)$ and a covariance function or kernel $k_0(x, x')$ define the $\mathcal{GP}$[149]:

$$f(x) \sim \mathcal{GP}(\mu_0(x_{1:k}), k_0(x_{1:k})) \tag{2.55}$$

A radial basis function kernel is widely employed for k_0 where

$$k_0(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) \tag{2.56}$$

where l is the length scale.[150]

Following, the next sampling point (x_{i+1}) is selected by maximising an acquisition function. The most common acquisition function is Expected Improvement (EI), indeed EI is used in Chapter 3. EI is defined as

$$EI(x) = \mathbb{E}[\max(0, f_{best} - f(x))] \tag{2.57}$$

where f_{best} is the best f value observed thus far, $\mathbb{E}$ is the expectation operator.[149] In essence, $EI(x)$ computes the expected value of the improvement from a new sampling point x. By maximising Eq. (2.57), a new x_{next} is selected that is expected to lead to a new minimum/maximum. With the new x_{next}, the corresponding objective function $f(x_{next})$ is evaluated, added to to the current dataset and the $\mathcal{GP}$ model is updated accordingly.

2.6.2 Genetic Algorithm

Chapter 3 presents a new fragmentation scheme based on the optimisation of a scoring function where the genetic algorithm (GA) is employed for the optimisation. This section provides an overview of the GA method.

GA approaches operates on the idea of "survival of the fittest" whereby the genes of the 'fit' individual are passed onto the next generation. Additionally, GA also utilises concepts from the behaviour of chromosomes when cells undergo division, *e.g.* mutation and crossover to create the next generation. Figure 2.8 illustrates a general schematic of the GA and where the crossover and mutation are located within the procedure.

As shown in Fig. 2.8 and Algorithm 2.1, the GA method begins with a population which is a collection of individuals and an individual contains a collection of genes. In the context of Chapter 3, an individual represents a fragmentation scheme and a gene corresponds to a bond. An individual is a binary vector (vector of 0's and 1's) that signals which bonds to break (for fragmentation) and which bonds to preserve (to as not perturb the chemical environment detrimentally).

Next, the fitness of each individual is determined, in the context of Chapter 3, this involves evaluating the corresponding scoring function (Eq. (3.3)) whose optimal weights

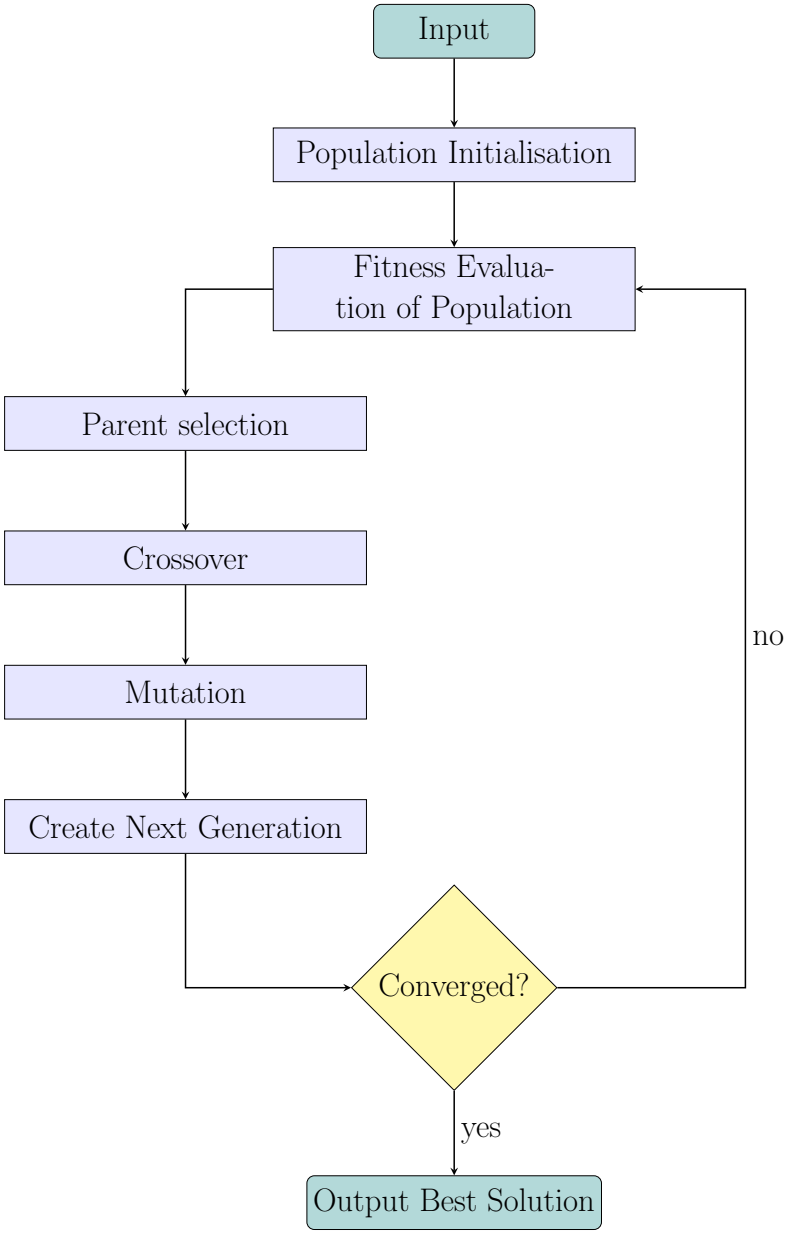


Figure 2.8: General schematic of the genetic algorithm.

Algorithm 2.1 Genetic Algorithm

1: Initialise population P
2: Determine fitness of each individual in P
3: **while** not converged **do**
4: Select parents from population P
5: Apply crossover to parents to generate new population P_{new}
6: Apply mutation to individuals in P_{new}
7: Calculate fitness for each individual in P_{new}
8: $P \leftarrow P_{\text{new}}$
9: **end while**

were determined with BO. Based on the fitness values calculated, parents are selected to generate individuals for the next generation. There are multiple parent selection methods such as tournament, roulette wheel, stochastic universal sampling and rank selection. Individuals not selected do not survive to the next generation. Following, crossover occurs to combine the genetic information of the parents to form the offspring, as well as mutation which further modifies the genetic makeup of the offspring. The next generation of the population contains only the parents and their offspring. This process of parent selection and mating repeats until convergence.

GA was selected as the optimiser for tackling the fragmentation problem in Chapter 3 for two main reasons. First, it is well suited to exploiting parallel computing architecture at scale, which in turn helps reduce execution time. Second, it is particularly adept at identifying global minima in complex combinatorial challenges,[151] such as the fragmentation problem addressed herein, where the aim is to find the optimal combination of edges to cut and leave intact.

Automatic Molecular Fragmentation by Evolutionary Optimisation

3.1 Introduction

Quantum mechanical models provide an accurate description of the binding interactions (*e.g.* hydrogen bonding, many body effects, $\pi \cdots$CH interactions) in protein-ligand systems. However, the computational time required by accurate QM methods increases extremely fast—formally faster than $\mathcal{O}(N^4)$[8,152,153]—with the size of the system. This rapid growth in computational demand severely limits the applicability of these methods to large molecular systems. Additionally, the algorithms fundamental to QM calculations are generally not optimised to leverage the extensive parallelism inherent in contemporary supercomputer architectures, which further complicates this challenge.

Molecular fragmentation has emerged as a successful strategy to overcome such concerns. By dividing the molecular system into smaller partitions, and using expansions such as MBE (Eq. (2.46)), fragmentation techniques reduce the formal complexity of QM methods to $\mathcal{O}(N)$. Furthermore, fragmentation exposes parallelism, enhancing algorithmic parallelisation and opportunity to leverage heterogeneous architectures. Although fragmentation methods offer considerable advantages, they are usually not applicable in a general black-box fashion to medium and large molecular systems. This can be primarily attributed to a dearth of automated fragmentation procedures. Currently, the design of fragments that yield accurate results is typically performed manually, requiring a laborious iterative combination of chemical intuition and trial and error. This not only limits the size of systems that can be accurately fragmented and studied, but also renders the resulting fragmentation schemes largely nontransferable across molecular systems and application studies.

Automated bond-breaking fragmentation algorithms have been developed in conjunction with the Molecular Tailoring Approach,[154,155] Systematic Molecular Fragmentation,[156–158] and the Generalised Energy Based Fragmentation.[159–161] These techniques create fragments from small units like functional groups or non-hydrogen atoms, select-

ing a specific size based on distance criteria (either the number of bonds or spatial distance). However, these fragmentation algorithms generally do not explicitly consider the surrounding chemical environment nor the energetic effects of the bond breaking, as fragment generation is primarily guided by basic distance and connectivity factors.

The main challenge in constructing a high quality fragmentation scheme is the generation of an optimal set of fragments that minimises the fragmentation energy error while retaining a user-defined fragment size. This goal is elusive and remains largely unaccomplished, primarily due to the absence of fragmentation strategies focused on generating high-quality fragments. A key issue with current schemes is their lack of explicit consideration for the types of bonds being broken. It is well-recognised that different sets of fragments, resulting from breaking various bonds, can lead to varied approximations of the final energy value. The severance of different bonds, yielding unique sets of fragments, results in the loss of distinct chemical interactions. This, in turn, leads to different estimates of the final energy of the unfragmented system.

The significance of the nature of bond breaking in molecular fragmentation was highlighted, for example, in a previous study which focused on the application of the Fragment Molecular Orbital (FMO)[162,163] method to DNA molecules. In this study, DNA was fragmented by cutting either the carbon-carbon (C−C) or carbon-oxygen (C−O) bond between the five-carbon sugar and phosphate group, as shown in Fig. 3.1. Considering the close proximity of these bonds, one might anticipate similar energy estimations for the intact system using both fragmentation schemes. However, the calculated energies for the two approaches showed a significant difference, exceeding 18 kJ mol^{-1}.[164]

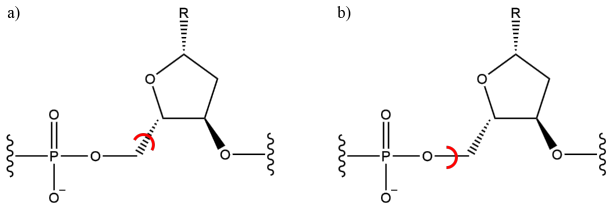


Figure 3.1: The two alternative fragmentation schemes for fragmenting DNA used in[164]. Fragments are formed by either a) breaking the C−C bond or b) breaking the C−O bond.

Thus, the issue at hand raises an important question: *How can one measure the effectiveness of a molecular fragmentation scheme?*

In current methodologies, the efficiency of these schemes is not known beforehand. Instead, their effectiveness is only determined retrospectively. This is done by calculating the system energy with and without fragmentation and comparing the results. However, for large molecular systems comprising hundreds to thousands of atoms, calculating the unfragmented system energy with traditional QM approaches is impractically demanding. The primary aim of fragmentation methods is, in fact, to circumvent

this very challenge. Consequently, employing such a metric for evaluation is neither practical nor reasonable, and necessitates the development of a more feasible alternative approach.

In this Chapter, a novel automatic fragmentation scheme that aims to obtain the optimal sets of fragments for a molecular system is presented. This new approach, named the Quick Fragmentation via Automated Genetic Search (QFRAGS), employs a specialised scoring function to assess fragmentation quality. The scoring function is designed and parameterised to obtain a strong correlation with the energy error of the resulting fragmentation scheme, thereby circumventing the usage of an expensive and generally unusable direct energy error metric. This enables recasting the fragmentation problem as an evolutionary optimisation of the scoring function, which is a rapid, cost-efficient and accurate process.

This Chapter is structured as follows. Section 3.2 begins with describing the datasets of molecular systems QFRAGS was applied to. This is followed by a description of the methodology of the fragmentation scheme, particularly, the scoring function used to describe the quality of fragmentation and the mathematical representation of the molecular system and its fragmentation. Then, the approach used for the optimisation of weights in the scoring function is detailed. Next, in Section 3.3, the algorithms utilised for the optimisation of the scoring function are discussed. The optimised weights are reported in Section 3.4 and these were used in the application of QFRAGS to over 1,000 protein systems. The corresponding fragment sizes generated and their corresponding energetic accuracy are discussed in Section 3.4. To further exemplify the accuracy of QFRAGS, a comparison to three manual fragmentation schemes is presented in Section 3.4. Section 3.5 concludes.

This chapter addresses Aim 1 and reproduces the following published manuscript: **Yu, F.C.**, Galvez Vallejo, J.L. and Barca, G.M., 2024. Automatic molecular fragmentation by evolutionary optimisation. *Journal of Cheminformatics, 16*(1), p.102.

3.2 Materials and Methods

3.2.1 Datasets

The automatic fragmentation algorithm will be applied across a range of biologically significant protein systems.

Figure 3.2 shows the classification of the three protein datasets used in this study. Cumulatively, the datasets comprise 1,100 protein structures obtained from two different sources. A subset of 100 protein structures, included in Dataset 3 as indicated in Fig. 3.2, was extracted from the PDB-Bind database.[165] These structures are characterised by having more than 500 atoms each. Notably, protein systems with less than

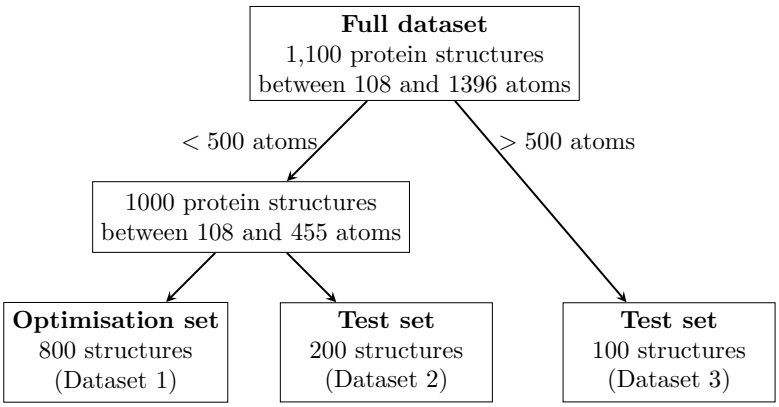


Figure 3.2: Classification of datasets used in this study.

500 atoms are rare in the PDB-Bind dataset. To address the shortage of such systems, existing protein structures in the PDB-Bind dataset were fragmented to generate 1,000 additional systems with less than 500 atoms. This involved severing single $C_\alpha-N$ or $C_\alpha-C$ bonds and valence is restored by appending hydrogens along the axis of the bond cut. Specifically, the coordinates of the hydrogen cap $\mathbf{x}(H)$ is given by

$$\mathbf{x}(H) = \mathbf{x}(i) + \frac{r(i) + r(H)}{r(i) + r(j)} \left(\mathbf{x}(j) - \mathbf{x}(i)\right) \tag{3.1}$$

where $\mathbf{x}$ denotes a Cartesian coordinate, r is the standard covalent radius given by Cordero *et al*,[166] and i, j denote the atoms belonging to the severed bond.

None of the resulting 1,000 systems (< 500 atoms) were derived from structures present in Dataset 3. All datasets are mutually exclusive.

All protein structures were hydrogenated using the PDBFixer software at the default pH of 7.0. All protein structures herein comprise one polypeptide chain and no metal-dependent structures are present within the dataset.

As illustrated in Fig. 3.2, the classification of the datasets involves an initial split of the full dataset based on the size where 500 atoms is the threshold. This threshold distinguishes between structures taken directly from the PDB-Bind dataset as opposed to the generated structures. The dataset of 1,000 generated systems is further divided in two datasets with a 80:20 split. Here, 80% of the structures are used for the optimisation of hyperparameters within the fragmentation algorithm (Dataset 1) and 20% is used to test the application of QFRAGS with the optimised hyperparameters (Dataset 2). The optimisation of the hyperparameters is presented in Section 3.2.4. Similar to Dataset 2, Dataset 3 is also a test dataset but for protein systems with more than 500 atoms. The size distributions of the three datasets are shown in Fig. 3.3.

Structures used for the optimisation of hyperparameters (Dataset 1) comprise a diverse set of protein sequences; 90.4% of protein pairs exhibited pairwise sequence iden-

tity (PID) scores below 20%. On the other hand, 6.3% of structure pairs exhibit PID values greater than 30%. Furthermore, these structures also exhibit a wide range of functionalities including: signalling proteins, structural proteins, toxins, viral proteins, enzymes, DNA/RNA binding proteins, transcription and transport proteins.

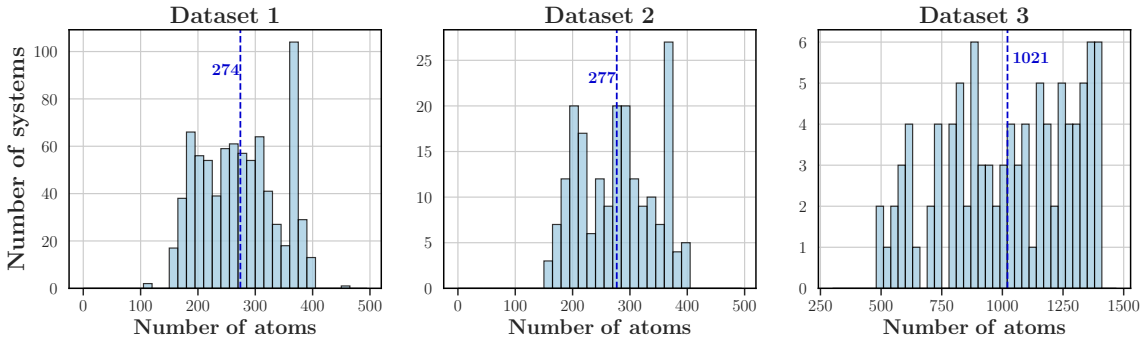


Figure 3.3: Size distribution of systems belonging to the three datasets: Dataset 1, 2 and 3. Averages are indicated by vertical lines and the corresponding values are reported.

In addition, QFRAGS was also applied to 10 glycolipid or lipoglycans systems ranging between 368 and 727 atoms to demonstrate its applicability to systems beyond proteins. All structures were taken from the Human Metabolome Database (HMDB) directly. These structures were selected on the basis of size. Specifically, structures within HMDB were sorted against size and 10 systems were randomly selected belonging in the top 20 largest glycolipids/lipoglycans. Glycolipids and lipoglycans structures were selected to apply QFRAGS to for two main reasons. Firstly, such systems are of particular biological significance as they include structures that form part of cell membranes responsible for structural integrity or modulating signal transduction events, as well as serving as intermediates in the synthesis pathway of glycans where disruptions can lead to congenital disorders of glycosylation. Secondly, unlike proteins which have an intuitive monomeric unit (amino acids), lipoglycans/glycolipids do not and we endeavour to examine the performance of QFRAGS on such systems.

3.2.2 Single Point Hartree-Fock Energy Calculations

In this study, single point energy calculations on molecular systems are conducted to evaluate the effectiveness of the proposed fragmentation algorithm and to fine-tune the hyperparameters utilised in the process. These calculations were consistently carried out at the Hartree-Fock theoretical level, employing the 6-31G* basis set. An overview of the HF method is presented in Chapter 2 Section 2.4.1. Unless specifically indicated, all computations were executed using the Extreme-scale Electronic Structure System (EXESS) quantum chemical software package.[23,24,26,28,167–169]

To assess the accuracy of the proposed fragmentation scheme, the difference between the energy of the total unfragmented system (E_{tot}) and the energy obtained *via*

fragmentation (E_f) is the metric used in this Chapter:

$$\Delta E = E_{tot} - E_f. \tag{3.2}$$

Evaluating ΔE requires performing full system energy calculations (E_{tot}) on datasets of protein systems, each containing hundreds or thousands of atoms. To achieve this within a feasible time frame, an improved initial guess scheme for the density matrix in the SCF procedure was employed instead of the traditional Superposition of Atomic Densities (SAD) approach. Specifically, the MBE initial guess method described in Chapter 4 Section 4.2.3, was used, with fragment sizes of approximately 30 atoms.

The fragmentation-based single point energy calculations (E_f) were performed using two methods: the Many Body Expansion (MBE) (Eq. (2.46)) and the Fragment Molecular Orbital (FMO) approach, both at the dimer (MBE2 and FMO2) and trimer (MBE3 and FMO3) levels. An overview of the MBE approach is presented in Chapter 2 Section 2.4.5. FMO is similar to MBE in that it utilises Eq. (2.46) to recombine fragment energies. However, rather than performing fragment energy calculations *in vacuo* as in MBE, in FMO these are performed in a self-consistent manner with respect to an electrostatic embedding, known also as Coulomb bath or ESP (electrostatic potential), of the surrounding monomers.[162] Furthermore, in the MBE implementation, hydrogen capping is used to restore valence at the sites of bond breaking. Hydrogen atoms are appended to fragments along the axis of the broken bond. On the other hand for FMO, the adaptive frozen orbital (AFO) approach was employed for the treatment of broken bonds. This involves freezing the molecular orbital of the broken bond.[170,171] All FMO calculations were performed using the GAMESS quantum chemical software package.[172]

In dimer calculations, all possible dimers were included and for trimer calculations, all possible dimers and trimers were included.

3.2.3 Methods for Automatic Fragmentation

In this section, the methods underpinning the proposed automatic fragmentation algorithm are described. This includes the representation of a molecular system, the metrics employed to evaluate the quality of fragmentation, and the representation of fragmentation involving bond breaking.

Molecular graph characterisation

As illustrated in Fig. 3.4, in the fragmentation algorithm, a molecular system is represented as a graph where the nodes and edges correspond to atoms and covalent bonds, respectively. Similar representations have been employed across multiple studies that

use graphs to capture the connectivity of molecular systems.[173–175]

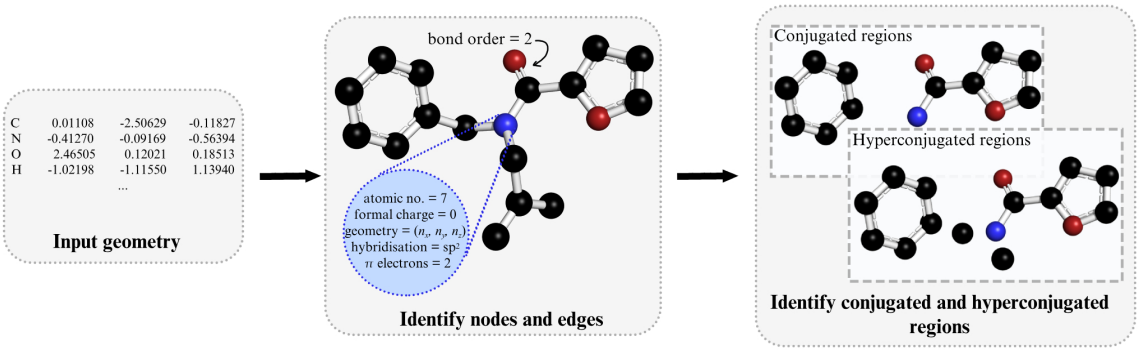


Figure 3.4: Graph representation of an input molecular system. Hydrogen atoms are omitted for clarity. Loner nodes in hyperconjugated regions correspond to hyperconjugated donor/acceptor C−H.

In the molecular graph representation, nodes and edges are assigned distinct attributes to accurately map the molecular system. Each node, representing an atom, is characterised by several attributes: the atomic number, formal charge, number of π electrons, hybridisation state, and its Cartesian coordinates. The attribute for the number of π electrons holds a non-zero value only for atoms that are components of a conjugated system. In such cases, this value corresponds to the number of π electrons that the atom contributes to the system. For instance, the nitrogen atom in a pyrrole molecule possesses two π electrons.

In contrast, bond order represents the sole attribute of an edge. Edges, or bonds, along with their respective bond orders, are determined based on the distances found in the Computational Chemistry Comparison and Benchmark DataBase, which includes experimental bond lengths.[176]

The program not only characterises the nodes and edges in the molecular graph but also identifies regions of conjugation and hyperconjugation. This identification is crucial for understanding how fragmentation might disrupt these molecular features. However, aromatic regions are not considered in this context, as the current implementation is limited to the breaking of single bonds. The rationale behind this limitation is discussed in greater detail in Section 3.3.1.

Conjugated regions identified as part of the molecular graph refer to groups of atoms exhibiting π-conjugation. This effect occurs when there are alternating single and double/triple bonds along a chain of the structure,[177] and π-electrons across the atoms becoming delocalised. To represent this, in the program, a conjugated group is defined as a group of connected nodes where every node has a hybridisation state of either sp^2 or sp.

Hyperconjugation involves the interaction between polarised σ-bonds and nearby

π-orbitals.[178] π-orbitals are found in various forms, including conjugated systems, double or triple bonds, and lone pairs on atoms. Polarised σ-bonds are typically of the form C−X, where X is a hydrogen or a halogen. In a hyperconjugated pair, there are donor and acceptor groups, which can be either π-systems or σ-systems, or a combination of both. However, for a pair to be considered hyperconjugated, it must include one σ-system and one π-system. In the current software implementation, the donor and acceptor groups are limited to being at most three bonds apart. Table 3.1 outlines the specific hyperconjugation donor and acceptor groups identified in this scheme. The conjugated groups that are identified serve as potential donors and acceptors for hyperconjugation.

Table 3.1: Classification of hyperconjugated groups. Corresponding σ/π nature that the program identifies. Hybridisation states and charges are shown on relevant atoms.

Group	Type	Classification
C=C	π	donor/acceptor
C≡C	π	donor/acceptor
C−H	σ	donor/acceptor
C=O	π	acceptor
C−F	σ	acceptor
C−Cl	σ	acceptor
C−Br	σ	acceptor
C−I	σ	acceptor
$C^{+}(sp^2)$	π	acceptor
$C^{-}(sp^2)$	π	donor
$N(sp^3)$	π	donor
$O(sp^3)$	π	donor

Scoring Function

In pursuit of an alternative non-energy-based metric to describe the quality of fragmentation, the following scoring function is employed

$$\begin{aligned} s =& \beta_1 p_{pe} + \beta_2 p_{conj} + \beta_3 p_{hyper} + \beta_4 p_{vol} \\ &+ \beta_5 p_{comp} + \beta_6 p_{vrange} \end{aligned} \tag{3.3}$$

where the penalty factors p_i are designed to account for various chemical and implementation factors. Broadly, these penalty factors fall into two categories. The first category encompasses penalties related to potential energy, conjugation, and hyperconjugation, with the primary objective of maintaining the chemical environment's integrity. The second category focuses on managing fragment size. This includes penalties based on the volume of fragments, the number of fragments, and the range of their volumes, ensuring that the fragments produced closely match the desired target size.

Each penalty p_i is a function that takes a set of broken bonds as input and produces a corresponding penalty value associated with the factor i. The parameters β_i serve as the weights for these penalties p_i. The subsequent text outlines the formulation of each penalty term.

Potential Energy Penalty The p_{pe} penalty is a measure of the change in the potential energy of the system induced by the fragmentation scheme. This is evaluated according to the formula:

$$p_{pe} = \frac{1}{1 + \exp\left(-\frac{\lambda}{\gamma}(\Delta_{pe} - \gamma d)\right)} + \frac{1}{1 + \exp\left(-\frac{\lambda}{\gamma}(-\Delta_{pe} - \gamma d)\right)}. \tag{3.4}$$

The formula for p_{pe} comprises two logistic sigmoid functions that are mirror images of each other and handle positive and negative Δ_{pe} values. The Δ_{pe} term is the difference between the energy of the total unfragmented system (E_{tot}) and the total energy (E_{MBE1}) obtained at the one-body MBE level (MBE1)

$$\Delta_{pe} = E_{tot} - E_{MBE1}. \tag{3.5}$$

Here, both the E_{tot} and E_{MBE1} energies are calculated using the universal force field (UFF).[179,180] This was selected due to its accessibility with parameters available for all atoms in the periodic table[181] as well as its low computational evaluation time.

The values of the parameters λ and d in Eq. (3.4) are 1.963 and 6, respectively. These are defined based on where the sigmoid function has sufficiently approached the lower and upper asymptotes. An arbitrary tolerance of 5% from each asymptote is used to define the lower and upper thresholds, consistent with the definition of "sufficiently approached" employed by McDowall and co-workers.[182] Thus, λ and d were selected by setting the lower and upper threshold values of the positive sigmoid function to correspond to $\Delta_{pe} = 10$ and $\Delta_{pe} = 40$ kJ mol^{-1}, respectively. These Δ_{pe} values the are called the boundary points.

The γ parameter in Eq. (3.4) is a scaling function and is defined as follows

$$\gamma = \frac{\sqrt{N_f} \cdot N_A^{min}}{n_t} \tag{3.6}$$

where N_f is the number of fragments, N_A^{min} is the number of atoms in the smallest fragment, and n_t is the target fragment size. The role of this scaling factor is to modulate the range between the boundary points. If γ is small, the range becomes narrower and the opposite is true for larger γ values.

The UFF employed in the computation of p_{pe} implements simple functional forms,

and may face difficulty in accounting for effects such as conjugation and electronic effects including hyperconjugation.[183,184] Thus, the effects of the fragmentation on conjugation and hyperconjugation are included as separate penalties in the scoring function.

Conjugation Penalty The conjugation penalty (p_{conj}) is defined as

$$p_{conj} = \frac{1}{N_{cs}} \sum_{k}^{N_{cs}} \mathcal{S}\left(\Delta_{conj}^{k}\right) \tag{3.7}$$

where k indexes the conjugated systems that have been disrupted by fragmentation, N_{cs} is the total number of affected conjugated systems, $\mathcal{S}$ is a normalisation function, Δ_{conj}^{k} factors for conjugated system k and is defined as follows

$$\Delta_{conj} = \frac{1}{cs}\left(\frac{1}{N_A}\sum_{i}^{N_A}\frac{N_e^i}{N_A^i} - cs\right). \tag{3.8}$$

Here, N_A is the number of atoms within the conjugated system, with each atom being indexed by i. The term N_e^i denotes the number of π electrons contributed by atom i to the conjugated system. Additionally, N_A^i refers to the aggregate count of atoms in the conjugated system that remains interconnected after fragmentation, specifically in the fragment to which atom i pertains. The terms cs is a conjugation score of the system given by

$$cs = \frac{1}{N_A}\sum_{i}^{N_A}\frac{N_e^i}{N_A} \tag{3.9}$$

For example, consider the case of pyrrole which consists of one conjugated system ($N_{cs} = 1$). In pyrrole, all non-hydrogen atoms participate in conjugation ($N_A = 5$). Each carbon atom contributes one π electron and the nitrogen atom contributes two π electrons from its lone pair. Thus, the conjugation score of pyrrole is

$$cs_{pyrrole} = \frac{1}{5}\left(\frac{1}{5}+\frac{1}{5}+\frac{1}{5}+\frac{1}{5}+\frac{2}{5}\right) = \frac{6}{25} \tag{3.10}$$

The normalisation function $\mathcal{S}$ in Eq. (3.7) has the form

$$\mathcal{S}(\Delta_{conj}) = \frac{1-\exp\left(-\lambda\Delta_{conj}\right)}{1+\exp\left(-\lambda\Delta_{conj}\right)} \tag{3.11}$$

Similar to the normalisation function for the potential energy, the exponent λ is chosen by setting a boundary value to correspond to 5% below the upper asymptote. Specifically, $\mathcal{S}(\Delta_{conj}^{max}) = 0.95$ where Δ_{conj}^{max} is the maximum possible value of Δ_{conj} and arises when every bond in the conjugated system is broken. The value of Δ_{conj}^{max} is obtained as follows

$$\Delta_{conj}^{max} = N_A - 1 \tag{3.12}$$

Hyperconjugation Penalty The hyperconjugation penalty p_{hyper} is calculated using the following formula

$$p_{hyper} = \frac{1}{N_{hs}} \sum_{k}^{N_{hs}} \frac{1}{\gamma_k} \mathcal{S}\left(\Delta_{hyper}^{k}\right). \tag{3.13}$$

Here k indexes the hyperconjugated systems, each comprising a donor and an acceptor group, disrupted by fragmentation. The N_{hs} term represents the total count of these affected systems, γ_k is the bond count between the donor and acceptor in system k, $\mathcal{S}$ is a normalisation function, and Δ_{hyper}^{k} is defined for each hyperconjugated pair k as follows

$$\Delta_{hyper} = \frac{1}{N_d} \sum_{i}^{N_d} \frac{N_e^i}{N_A^i} - \frac{1}{N_a} \sum_{j}^{N_a} \frac{N_e^j}{N_A^j} \tag{3.14}$$

Eq. (3.14) captures the change in electron distribution across the donor and acceptor atoms due to fragmentation. In the first term, N_d denotes the number of fragments containing donor atoms from the hyperconjugated pair. The index i identifies these fragments. N_e^i and N_A^i respectively represent the number of electrons donated and the count of atoms in fragment i. The second term mirrors the first, focusing on acceptor atoms. N_a indicates the count of fragments with acceptor atoms, with j indexing these fragments. N_e^j and N_A^j respectively represent the number of electrons accepted and the count of atoms in fragment j.

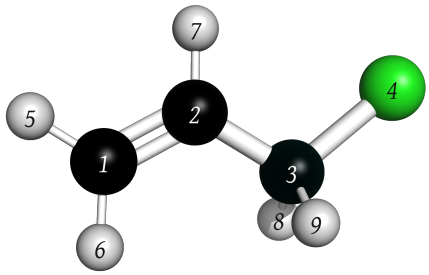


Figure 3.5: Labelled ball and stick model of 3-chloroprop-1-ene. Green, black, white spheres correspond to chloride, carbon and hydrogen atoms, respectively.

To illustrate the meaning of the terms within Eq. (3.14), consider fragmenting the molecule shown in Fig. 3.5 by cutting the bond between atoms 2 and 3. There is only one hyperconjugation pair present, where the donor is the C=C bond (atoms 1 and 2) and the acceptor is the C−Cl bond (atoms 3 and 4). After fragmentation, the two donor atoms remain connected, therefore in the first term of Eq. (3.14), $N_d = 1$ and $N_A^i = 2$.

Here, $N_e^i = 2$ as the C=C bond contributes two π electrons to hyperconjugation and the two atoms (1 and 2) remain connected. Conversely, for the second term in Eq. (3.14), the two acceptor atoms remain connected, leading to $N_a = 1$ and $N_A^j = 2$. Since the bond bridging the donor and acceptor groups together has been cut (bond between atoms 2 and 3), electrons are no longer being donated to the acceptor, resulting in $N_e^j = 0$.

The functional form of $\mathcal{S}$ in Eq. (3.13) is identical to that of Eq. (3.11):

$$\mathcal{S}(\Delta_{hyper}) = \frac{1 - \exp\left(-\lambda\Delta_{hyper}\right)}{1 + \exp\left(-\lambda\Delta_{hyper}\right)} \tag{3.15}$$

The parameter λ was selected by setting $\mathcal{S}(\Delta_{hyper}^{max}) = 0.95$, with Δ_{hyper}^{max} being the maximum value of Δ_{hyper} calculated as

$$\Delta_{hyper}^{max} = \frac{N_e}{N_A^d} \tag{3.16}$$

where N_e is the sum of all the electrons being donated in the hyperconjugated system and N_A^d is the total number of donor atoms in the system.

The penalty terms discussed thus far all relate to capturing the perturbation in the chemical environment. The following subsections provide the formulation of the penalty terms associated with controlling the fragment size.

Volume Penalty The volume penalty (p_{vol}) is defined as

$$p_{vol} = \frac{1}{N_f} \sum_{k}^{N_f} \frac{1}{1 + \exp\left(-14.654\left(\Delta_{vol}\right)^2\right)} \tag{3.17}$$

Here N_f is the number of fragments, k indexes each fragment, and

$$\Delta_{vol} = \frac{1}{N_f} \sum_{k}^{N_f} \left(\frac{V_k - V_{ref}}{V_{ref}} \right) \tag{3.18}$$

where V_k is the volume of fragment k, and V_{ref} is the reference volume, which is determined from the target fragment size as discussed further below. The exponent of -14.654 in Eq. (3.17) was selected based on where the function has sufficiently approached the asymptote value of $p_{vol} = 1$. Specifically, the exponent was selected such that when $|\Delta_{vol}| = 0.5$, $p_{vol} = 0.95$.

The volume of a fragment is evaluated according to the following formula

$$V = \sum_i V_i - \sum_{i<j} V_{ij} \tag{3.19}$$

Both i and j index atoms belonging the fragment. The V_i term is the hard-sphere equivalent atomic volume[185]

$$V_i = \frac{4}{3}\pi\sigma_i^3 \tag{3.20}$$

where σ_i is the van der Waals radius of atom i, and V_{ij} is the overlapping volume between two atoms[185,186]

$$V_{ij} = a_i a_j \exp\left(-\frac{\alpha_i\alpha_j r_{ij}^2}{\alpha_i + \alpha_j}\right)\left(\frac{\pi}{\alpha_i + \alpha_j}\right)^{\frac{3}{2}} \tag{3.21}$$

The amplitude a_i is set to a default value of $2\sqrt{2}$, and r_{ij} denotes the distance between atoms i and j. The α_i term is calculated from σ_i as follows

$$\alpha_i = \pi\left(\frac{3a_i}{4\pi\sigma_i^3}\right)^{\frac{2}{3}} \tag{3.22}$$

The reference volume V_{ref} is computed using the following equation

$$V_{ref} = n_t \cdot \frac{1}{N_A}\sum_{s\in S} N_s V_s \tag{3.23}$$

Here, n_t represents the target fragment size, while N_A denotes the total number of atoms in the molecular system. The set S includes all unique atomic elements present in the molecular system, for instance, Argon (Ar), Carbon (C), Nitrogen (N), *etc.* The variable s is used to index these elements. N_s indicates the total count of atoms with the symbol s, and V_s represents the characteristic volume of an atom with symbol s in the molecular system, defined as follows

$$V_s = \frac{4}{3}\pi\sigma_s^3 - \frac{1}{\|K\|}\sum_{i\in K} V_{s,i} \tag{3.24}$$

In Equation (3.24), the first term calculates the hard-sphere volume of an atom denoted by s. Here, K refers to the set of atoms directly bonded to an atom symbolised by s, with i indexing these neighboring atoms. $V_{s,i}$ represents the overlapping volume between atom s and its neighbors in K. Thus, the second term in Equation (3.24) averages the overlapping volumes between atom s and its adjacent atoms. As an example, consider Fig. 3.6 illustrating the representative volume of oxygen V_O. In this case, the overlapping volumes between two atom pairs (designated as $n_{neigh}^O = 2$) are considered: between atoms 1 and 4, and atoms 4 and 5.

Volume-Range Penalty In the previous discussion on the volume penalty formulation, it is evident that p_{vol} serves as an indicator of the average variation in fragment volumes. This definition implies the possibility of creating a set of fragments with a low p_{vol} value, yet these fragments may vary significantly in size.

To address this issue, the following volume-range penalty is introduced

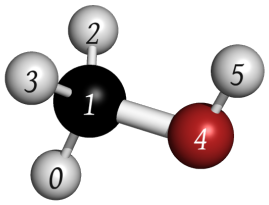


Figure 3.6: Labelled ball and stick model of methanol. Red, black, white spheres correspond to oxygen, carbon and hydrogen atoms, respectively.

$$p_{vrange} = \frac{1}{1 + \exp\left(-\lambda(\Delta_{vrange} - d)\right)} \tag{3.25}$$

where d is preset to -0.25, λ is adjusted to 11.78, and

$$\Delta_{vrange} = \frac{V_{range} - V_{ref}}{V_{ref}} \tag{3.26}$$

with V_{ref} defined as in Eq. (3.23), and V_{range} being the difference between the maximum and minimum fragment volumes.

Number-of-Components Penalty The final component of the scoring function is the penalty term p_{comp}, defined as

$$p_{comp} = \frac{1}{N_f} \tag{3.27}$$

where N_f is the number of fragments the molecular system has been divided into. This penalty term is designed to encourage scenarios with a higher count of fragments, while discouraging situations with fewer fragments. Specifically, p_{comp} seeks to mitigate cases where no fragmentation occurs (*i.e.*, $N_f = 1$), which is otherwise favored due to the small p_{vrange} value. In such instances, $\Delta_{vrange} = 0$ and $p_{vrange} \approx 0$, and this effect is counterbalanced by a high p_{comp} value of 1.

The β_i Weights Each penalty in Eq. (3.3) is weighted by a matching β_i factor. These factors are constrained to be non-negative ($\beta_i \geq 0$) and subject to a normalisation constraint, ensuring their sum equals one ($\sum_i \beta_i = 1$). The importance of these weights lies in mitigating the impact of double counting. For instance, hyperconjugation and conjugation are interrelated chemical phenomena, and their combined penalties can lead to an over-representation of chemical disturbances due to either conjugation or hyperconjugation. Therefore, the β_i weights play a crucial role in moderating the influence of each factor on the overall score, thereby attenuating the effects of potential statistical correlations among penalties. The process of determining the β_i values involves an optimisation procedure detailed in Section 3.2.4.

Representation of Fragmentation and Solution Space

The essence of the automated fragmentation scheme lies in minimising Eq. (3.3) to obtain an optimal set of fragments, each contributing to the best possible score.

This scheme segments a molecular system into fragments by breaking covalent bonds, resulting in edges being either broken (labelled as '1') or unbroken (labelled as '0'). Consequently, a fragmented molecular system is represented as a binary vector, where each element represents an edge in the molecular graph. This binary vector is then used as input for the scoring function in Eq. (3.3). The resulting score reflects the quality of the corresponding fragmentation.

The objective of the fragmentation algorithm is to partition a molecular system in a way that minimizes the scoring function. This involves an optimisation process, as outlined in Section 3.3.1, which minimises the scoring function and yields the ideal set of fragments.

3.2.4 Optimisation of Scoring Function Weights

In this subsection, the methodology used for optimising the weights of the scoring function, $\{\beta_i\}$, is provided.

The optimisation of the $\{\beta_i\}$ values, as applied in Eq. (3.3), is crucial for generating high-quality fragments. These weights quantitatively represent the importance of each penalty term in the scoring function. Suboptimal weightings can lead to an imbalanced scoring function. For instance, excessively high weighting for the volume penalty (p_{vol}) might result in an unduly low weight for the potential energy penalty (p_{pe}), leading to fragments with significant potential energy variations.

To determine the optimal $\{\beta_i\}$ values, an iterative Bayesian optimisation (BO) approach is employed. The weights were fine-tuned using Dataset 1, comprising 800 protein systems with sizes ranging from 108 to 455 atoms. A description of the BO approach can be found in Chapter 2 Section 2.6.1.

The Bayesian optimisation aims to minimise the objective function defined as

$$f = \alpha \frac{1}{n} \sum_i p_{vol}^i + (1-\alpha) \frac{1}{n} \sum_i \mathcal{S}(\Delta E_i) \tag{3.28}$$

Here, p_{vol} represents the volume penalty, and ΔE denotes the energy difference between the total unfragmented system and the MBE2 energy, both computed at the HF/6-31G* theory level. The symbol $\mathcal{S}$ in Eq. (3.28) indicates a normalisation function, analogous to that used for p_{pe}, as previously discussed in Section 3.2.3, with the exception that the boundary points of the sigmoid function correspond to 1 and 4 kJ mol^{-1}. The method for evaluating p_{vol} is also detailed in Section 3.2.3. In the equation, i indexes each protein system in the dataset, n is the total number of protein systems, and α is a

hyperparameter. This function, thus, represents a weighted average of the deviations in fragment volume and energy across the dataset. An α value of 0.5 was chosen to balance the significance of volume and energy equally.

The MBE fragmentation method was selected as it forms the basis of other fragmentation methods such as electrostatically-embedded MBE, generalised MBE and FMO.

The design of the objective function in Eq. (3.28) aims to derive $\{\beta_i\}$ values that guide the scoring function towards producing fragments with minimal MBE2 energy deviations from the complete, unfragmented system, while also maintaining fragment sizes near the desired target. For the optimisation procedure, the target fragment size was set as 50 atoms.

The surrogate model of the objective function was modelled using the Gaussian Process Regressor from the scikit-learn Python package, employing a radial basis function kernel.[150] The noise variance and length scales were set to 1.0. The expected improvement acquisition function guided the selection of subsequent $\{\beta_i\}$ values for sampling.

Figure 3.7 illustrates the optimisation workflow. The initial stage involved data preparation to build the Gaussian Process model. This step included computing the objective function f (Eq. (3.28)) for 16 different sets of $\{\beta_i\}$ values. These initial values were derived through a grid search, ranging from 0.1 to 0.9 in 0.1 intervals, as listed in Table 3.2. As Table 3.2 indicates, the minimum values for both β_{pe} and β_{vol} were set at 0.2, reflecting the anticipated higher significance and value of these factors (energy and volume) compared to others.

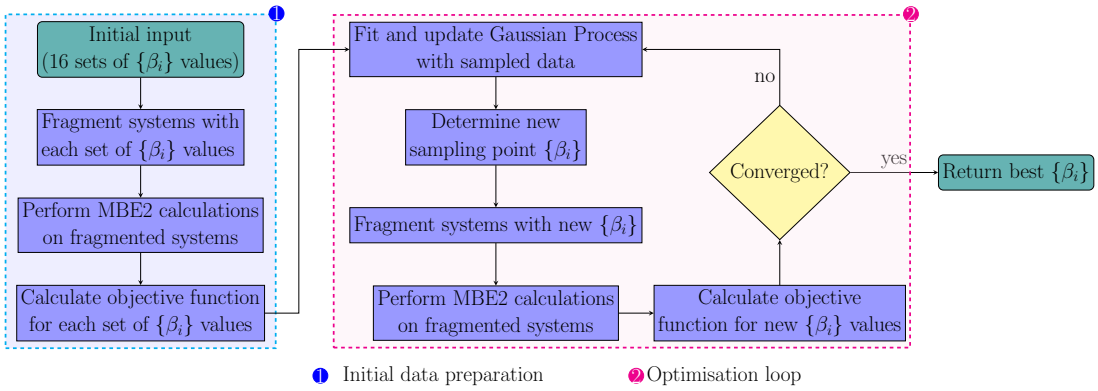


Figure 3.7: Workflow for the optimisation of $\{\beta_i\}$ values. The numbering corresponds to each of the phases: (1) Initial data preparation; (2) Optimisation loop.

The second stage in the optimisation of $\{\beta_i\}$ values includes the optimisation loop where each iteration involves updating the Gaussian Process with the recently sampled data, generating the next set of $\{\beta_i\}$ values to sample and using these to fragment and calculate the corresponding MBE2 energies. The objective function is evaluated using the MBE2 energies and volume penalties across all 800 protein systems in Dataset 1.

Table 3.2: Initial datasets of $\{\beta_i\}$ values.

β_{pe}	β_{conj}	β_{hyper}	β_{vol}	β_{comp}	β_{vrange}
0.2	0.1	0.1	0.2	0.1	0.3
0.2	0.1	0.1	0.2	0.2	0.2
0.2	0.1	0.1	0.3	0.1	0.2
0.2	0.1	0.1	0.4	0.1	0.1
0.2	0.1	0.2	0.2	0.1	0.2
0.2	0.1	0.2	0.3	0.1	0.1
0.2	0.1	0.3	0.2	0.1	0.1
0.2	0.2	0.1	0.2	0.1	0.2
0.2	0.2	0.1	0.3	0.1	0.1
0.2	0.2	0.2	0.2	0.1	0.1
0.2	0.3	0.1	0.2	0.1	0.1
0.3	0.1	0.1	0.2	0.1	0.2
0.3	0.1	0.1	0.3	0.1	0.1
0.3	0.1	0.2	0.2	0.1	0.1
0.3	0.2	0.1	0.2	0.1	0.1
0.4	0.1	0.1	0.2	0.1	0.1

This process was repeated until the minimum objective value remained unchanged for 200 iterations. A total of 645 iterations were performed accordingly.

3.3 Algorithms

This section describes the algorithms employed within QFRAGS for the optimisation of the scoring function (Eq. (3.3)) as well as the overall automated fragmentation algorithm.

3.3.1 Optimisation of Scoring Function

Restriction of solution space and allowed edges

As previously discussed, each specific solution to the fragmentation problem is represented in the form of a binary vector where each entry corresponds to the state of a bond (1 - broken, 0 - unbroken). However, since the aim is attaining fragments of a specific size, edges that when severed generate fragments that are too small can be eschewed from the solution space .

To accomplish this, the solution space is restricted to edges that when cut, do not generate fragments that are smaller than 60% of the target fragment size (n_t). For example, consider the fragmentation of the protein system consisting of MFS-bound Sans CEN2 peptide, with a PDB ID of 2L7T (174 atoms), into fragments containing $\sim$20 atoms. As shown in Fig. 3.8 if the A-B edge is cut, two fragments of size 6 and 168

atoms are generated. Since 6 atoms is much smaller than the target size of 20 atoms, the edge A-B is not considered part of the solution space.

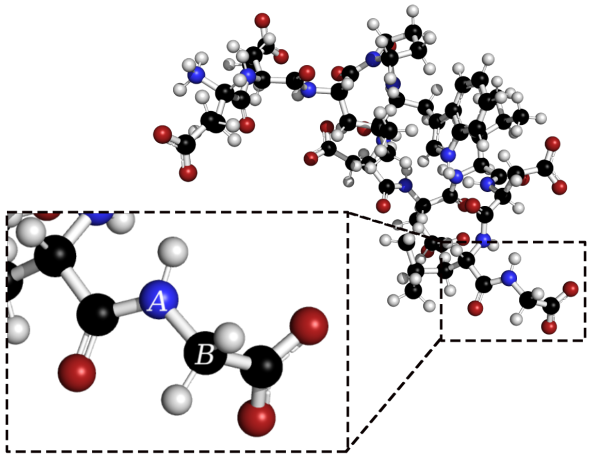


Figure 3.8: Ball and stick model of MFS-bound Sans CEN2 peptide (PDBid 2L7T) displaying an unallowed edge between atoms A and B. See text for more information.

In the present version of the fragmentation code, the solution space is constrained exclusively to single bonds. That is, only bonds with a bond order of one are permitted to be broken. From this point forward, the complete collection of edges within the solution space are referred to as *allowed edges*. In this implementation, all allowed edges are, without exception, single bonds. Note that according to the definitions of conjugated systems used herein, conjugated systems can include single bonds and consequently be disrupted by fragmentation.

Initial Guess

To enhance the optimiser's capability in identifying optimal solutions, a collection of preliminary approximations is supplied. These are instances of fragmentation that represent initial fragment groups.

For formulating these initial guesses, to commence, all permissible edges are eliminated from the molecular graph of the system, essentially breaking all bonds within the solution space. This process results in a group of diminutive fragments, which will be called primitive monomers. The fragments constituting the initial guess are subsequently assembled in a recursive manner by combining these primitive monomers, following the methodologies outlined in Algorithms 3.1 and 3.2.

In Algorithm 3.1, a reference point ref_point_i is required as an input to select a reference primitive monomer (ref_mon) to begin construction of the fragments. The computation of the set of reference points $\{ref\_point_i\}$ is dependent on system size and shape. Specifically, the Euclidean space occupied by the system is partitioned into three-dimensional rectangular intervals along the directions of the principal axes of in-

Algorithm 3.1 Constructing initial guess for fragmentation instance

Require: ref_point_i
1: Initialise empty fragment container F
2: **while** $visited\_nodes \neq$ all nodes **do**
3: $ref\_mon \Leftarrow$ unvisited primitive monomer closest to ref_point
4: $frag \Leftarrow$ build fragment from ref_mon (Algorithm 3.2)
5: Append $frag$ to F
6: Label $frag$ as visited
7: **end while**
8: Convert F to binary vector

ertia (the eigenvectors of the inertia tensor), and the midpoint of these intervals are taken as the reference points, as shown in Fig. 3.9. The inertia tensor was used to ensure that the calculation of the set of $\{ref\_point_i\}$ is invariant to translations, rotations and reflections in the geometry of the structure. Further detail on the computation of the reference points can be found in the Supplementary Information.

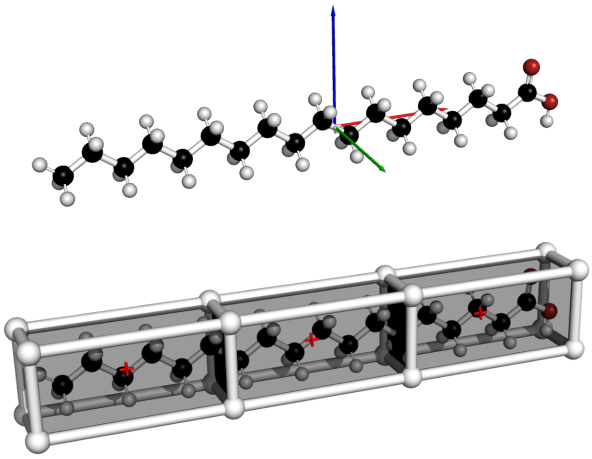


Figure 3.9: Ball and stick figure of steric acid; *top*: displaying the principal axes of inertia (red, green and blue arrows); *bottom*: displaying the three rectangular intervals formed using the principal axes of inertia, and the reference points (red crosses). All rectangular interval edges are parallel to the principal axes of inertia.

Algorithm 3.1 begins by initialising an empty fragment container which will contain the initial set of fragments obtained with ref_point_i. Lines 2 to 7 describe the strategy of forming these fragments. The algorithm monitors the nodes being visited and continues to build fragments until all nodes have been visited. The construction of each fragment begins at the primitive monomer closest to ref_point_i that is unvisited (line 3). A fragment, $frag$ on line 4, is built from ref_mon using Algorithm 3.2. The $frag$ object is a collection of primitive monomers, and this fragment is then appended to the fragment container F. On line 6, the nodes within $frag$ are marked as visited. This process is repeated until all nodes have been visited and the algorithm outputs a binary vector as the initial guess.

Algorithm 3.2 describes the procedure of constructing a fragment from a reference monomer (ref_mon). Two empty containers are initialised on lines 1 and 2. Both Q

and M hold primitive monomers. However, Q represents a queue and M is a container that will contain the set of primitive monomers to form a fragment. Next, on lines 3 and 4 ref_mon is added to both containers. Lines 5 to 16 describe the procedure of constructing the fragment which is a collection of primitive monomers. The algorithm uses a while loop and repeats until Q is empty.

Within each iteration of the while loop, Q is firstly dequeued and the first primitive monomer in Q is assigned to mon_v (line 6). Following, the neighbouring primitive monomers are iterated over (line 7) of mon_v, where mon_w denotes the neighbours and the visitation status of each neighbour is checked. If mon_w has not been visited, it is appended to Q and M (lines 8 to 10). Next on line 11, the algorithm checks the size (number of atoms) of the growing fragment container M and if the size is $\geq$ 90% of the target fragment size (n_t), M is returned; otherwise the while loop continues. Algorithm 3.2 repeats until the fragment size condition (line 11) has been satisfied or until the list of neighbouring monomers has been exhausted and Q becomes empty.

Algorithm 3.2 Building fragment from reference monomer

Require: ref_mon, n_t
1: Initialise empty queue container Q
2: Initialise empty primitive monomer container M
3: Enqueue ref_mon to Q
4: Append ref_mon to M
5: **while** $Q \neq$ empty **do**
6: $mon\_v \Leftarrow$ dequeue Q
7: **for** each monomer neighbour w of mon_v **do**
8: **if** $mon\_w \neq$ visited **then**
9: Enqueue mon_w to Q
10: Append mon_w to M
11: **if** $M.size \geq 0.9 \cdot n_t$ **then**
12: Return M
13: **end if**
14: **end if**
15: **end for**
16: **end while**

Optimiser

The minimisation of the scoring function in Eq. (3.3) is performed using a genetic algorithm (GA). A description of the GA method is provided in Chapter 2 Section 2.6.2.

Within the framework of the GA implementation herein, each individual within the population represents a distinct fragmentation scenario or a set of fragments. The starting population is comprised of initial guesses, originating from a dataset specifically prepared for this purpose, as discussed in Section 3.3.1.

In the implementation, each gene in an individual corresponds to an allowed edge. As described in Section 3.3.1 concerning the solution space, genes can only take on two possible values: 0 and 1. The total number of parents selected for mating is two if the population size is less than or equal to eight, and it is $\lfloor 0.25 \times$population size$\rfloor$ otherwise. Parents are selected according to the tournament selection technique.[187] The crossover

type is a single-point crossover[188] and mutation is random and occurs by replacement.

Algorithm 3.3 Iteration of the genetic algorithm

Require: min_score, $best\_sol$
1: Initialise pop_scores and $next\_generation$ to be empty containers
2: Select parents from previous generation and add to $next\_generation$
3: Create offspring and add to $next\_generation$ (crossover and mutation)
4: **for** each solution i in $next\_generation$ **do**
5: Calculate score and append to pop_scores
6: **end for**
7: $local\_min\_score \Leftarrow \min(pop\_scores)$
8: **if** $local\_min\_score < min\_score$ **then**
9: $min\_score \Leftarrow local\_min\_score$
10: $best\_sol \Leftarrow \text{argmin}(pop\_scores)$
11: **end if**

The GA approach involves an iterative procedure that aims to explore the solution space by allowing fit individuals to mate and pass its genes to the next generation. Algorithm 3.3 describes the procedure of creating the next generation in the GA scheme implemented. The fragmentation algorithm monitors the 'global' individual ($best\_sol$) with the minimum score (min_score). Fit individuals (parents) are chosen from the previous generation (line 2) and are used to create the offspring for the next generation via crossover (line 3). The solution with the minimum fitness in the next generation ($local\_min\_score$) is compared to min_score, and the global individual and minimum score are updated if a solution with a lower score is found (lines 9 to 10 of Algorithm 3.3). This iterative process is repeated until either the maximum number of iterations has been reached or if the minimum score has not changed for more than 50 iterations. A maximum number of iterations of 100 was adopted for all computational experiments.

To guide the optimiser in locating good quality solutions (low score from Eq. (3.3)), the dimer energy (Eq. (3.29)) is utilised to further restrict the solution space throughout the optimisation procedure. The dimer energy correction ΔE_{IJ} is calculated as

$$\Delta E_{IJ} = E_{IJ} - E_I - E_J \tag{3.29}$$

where E_{IJ} represents the energy of the dimer and E_I and E_J are the energies of the monomers. The value of ΔE_{IJ} provides a measure of the energy perturbation when the bond(s) connecting the two monomers in a dimer is/are broken. A force field treatment (UFF) is used for the calculations of the energies (E_{IJ}, E_I and E_J). As an example consider Fig. 3.10, where a C−C bond is broken in the dimer (F_{IJ}) to produce two monomers (F_I and F_J).

If the dimer energy correction corresponding to an edge being cut exceeds a threshold value, this edge is blacklisted and the corresponding bond remains unbreakable in future iterations. The threshold value used is 10 kJ mol^{-1}.

The evaluation of the dimer energy correction for each edge being cut is performed within the scoring function calculation for each individual in Algorithm 3.3. The pro-

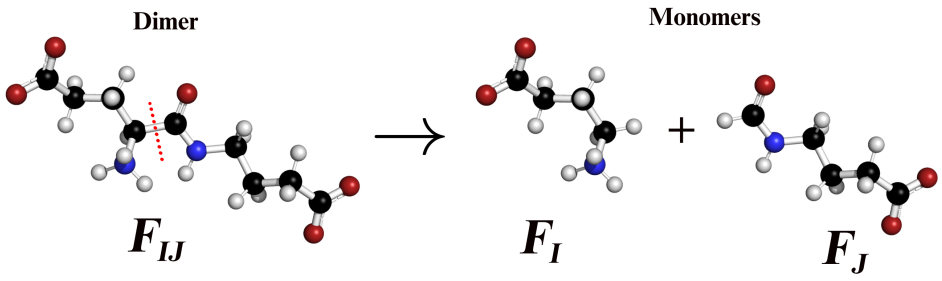


Figure 3.10: Ball and stick figure of two separate monomers (F_I and F_J) and the dimer (F_{IJ}) composing these monomers. Atoms coloured black, blue, red and white correspond to carbon, nitrogen, oxygen and hydrogen, respectively. Red broken line denotes a broken bond.

cedure of blacklisting edges is limited to the first ten iterations of a GA procedure. Otherwise if this continues across the entire optimisation procedure, there is the risk of potentially rendering the set of blacklisted edges to be too large and prevent the optimiser from exploring diverse solutions. By excluding edges associated with large dimer energies from the solution space, the optimiser is steered towards an energetically favourable solution that preserves the integrity of the chemical environment.

3.3.2 Fragmentation algorithm

Dealing with the complex optimisation challenge of optimally dividing a system into N_f fragments, each with approximately n_t atoms, proves arduous for the optimiser. As molecular systems grow, the difficulty escalates. The optimiser struggles to fragment the system in a feasible number of iterations, hindered by the exponential growth in the combinations of bonds.

To mitigate this issue, a recursive fragmentation approach is adopted. With this approach, a GA optimiser instance will need to consider a significantly lower number of potential broken bonds at any given time. In turn, breaking a smaller number of bonds results in a smaller cumulative effect on the score, enabling the optimiser to distinguish better between bonds that lead to low and high energy perturbations. Furthermore, the reduction in the problem size that comes with this recursive strategy also means less degenerate solutions for the optimiser to consider.

In the recursive procedure, the molecular system is initially partitioned into n larger fragments and each of these fragments is then broken up further. This process is repeated until the fragments are sufficiently close to the target fragment size.

This recursive procedure is exemplified in Fig. 3.11, where a 174-atom protein system (MFS-bound Sans CEN2 peptide, PDB ID: 2L7T) is fragmented, aiming for fragments of approximately 20 atoms. Initially, the algorithm splits the system into five larger fragments of 31, 35, 38, 36, and 42 atoms, respectively. These fragments are then

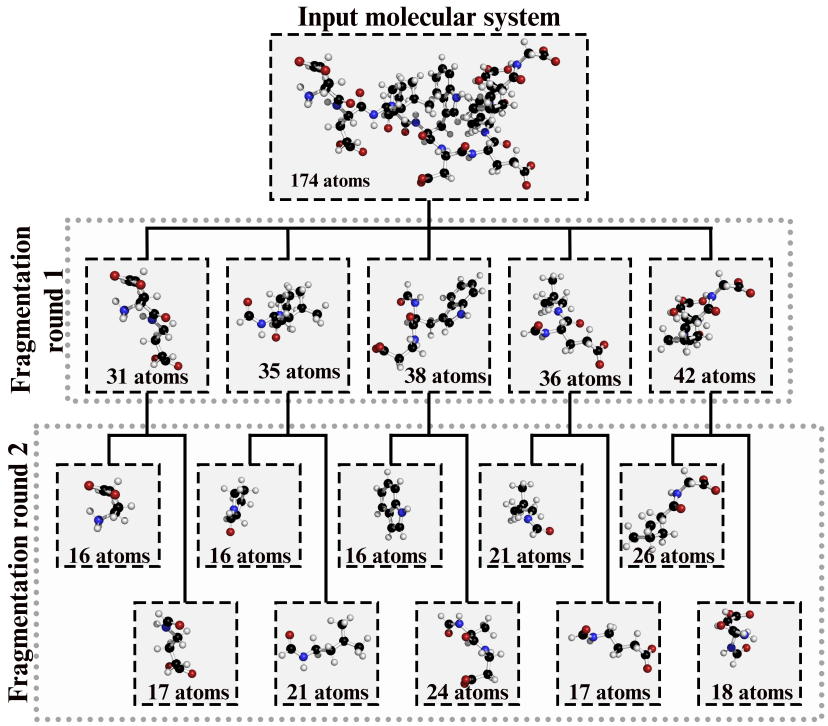


Figure 3.11: Example of the recursive fragmentation scheme with MFS-bound Sans CEN2 peptide (PDBid 2L7T). Target fragment size is 20 atoms. Number of atoms listed for fragments includes hydrogen caps.

further subdivided to achieve fragments nearing the desired 20-atom size. After two fragmentation stages, ten fragments emerge, each averaging approximately 19 atoms.

Figure 3.12 graphically illustrates the final automatic fragmentation algorithm. The process starts by analysing the molecular system, which involves categorising node and edge attributes and identifying conjugated and hyperconjugated areas. Subsequently, the system undergoes recursive fragmentation, incorporating a series of genetic algorithm optimisation steps. Following each fragmentation phase, the resulting fragment sizes ($||s||$) are assessed against the target size (n_t). If a fragment exceeds the target size, it undergoes further recursive fragmentation.

3.4 Results and Discussion

3.4.1 Optimisation of Scoring Function Weights

Figure 3.13 displays the evolution of the minimum value of the objective function f (Eq. (3.28)) over the course of the Bayesian optimisation procedure as described in Section 3.2.4. Within the first five iterations, there is a drastic drop in the minimum value from 0.38152 to 0.30799. Past this, the rate at which the minimum value decreases slows down considerably, indicating a relatively flat optimisation surface. A duration

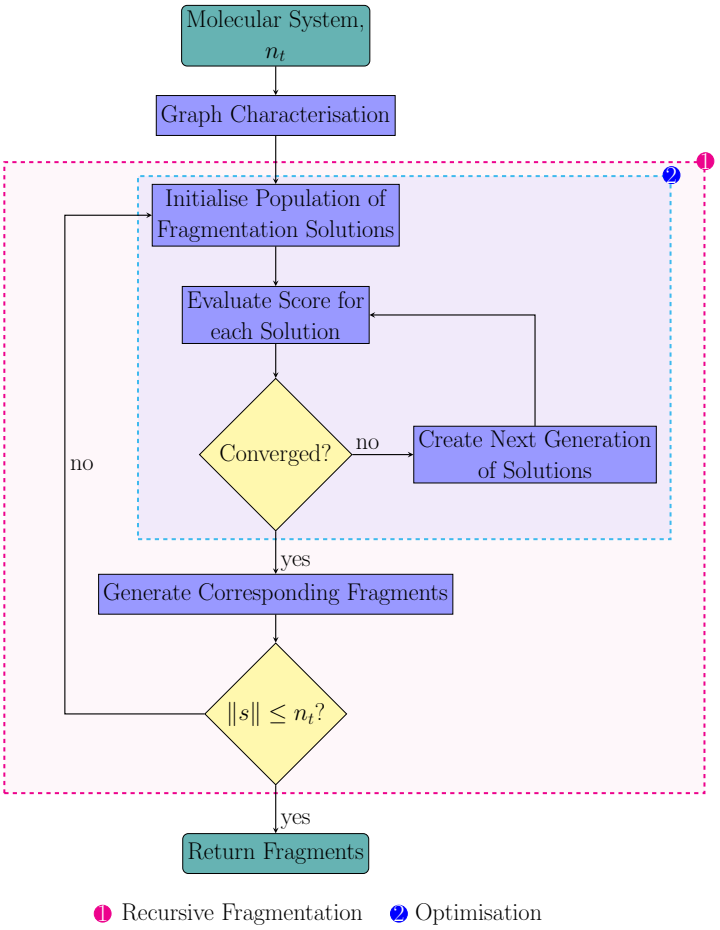


Figure 3.12: The automatic fragmentation algorithm. The numbering corresponds to different sections of the algorithm: (1) Recursive fragmentation; (2) Genetic algorithm.

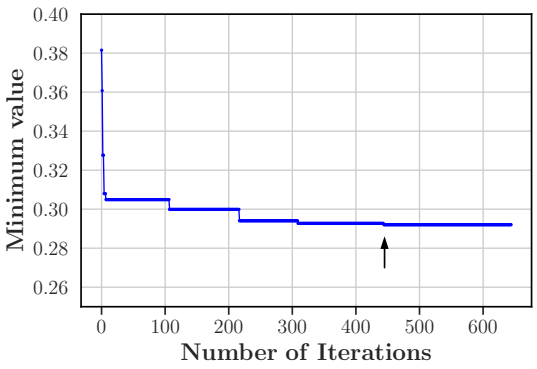


Figure 3.13: Evolution of the minimum value of f (Eq. (3.28)). Arrow indicates the occurrence of the final minimum value of f at iteration number 445.

of approximately 100 iterations were required for the occurrence of the next three minimum values; the final minimum value of 0.29205 (see black arrow in Fig. 3.13 at iteration number 445) occurred 135 iterations after the previous. After 200 iterations of no change in the minimum value, the optimisation procedure was terminated and the corresponding $\{\beta_i\}$ values at iteration number 445 were employed in the fragmentation algorithm.

Table 3.3 shows the set of $\{\beta_i\}$ values that minimise the scoring function (Eq. (3.3)) against the objective function (Eq. (3.28)).

The term carrying the largest weight in the scoring function is hyperconjugation ($\beta_{hyper} = 0.313325$), which is immediately followed by the volume range ($\beta_{vrange} = 0.294074$). As detailed in Section 3.2.3, the six penalty terms in the scoring function are divided into two primary classes: one focusing on maintaining the chemical landscape (encompassing potential energy, conjugation, and hyperconjugation), and the other on managing fragment size (including volume, the number of fragments/components, and volume range). It is noteworthy that the two most heavily weighted factors—hyperconjugation and volume range—belong to these distinct classes. Moreover, the aggregated weights of penalty terms involved in maintaining the chemical environment (0.595084) is greater than that of controlling the fragment size (0.404916). This difference in weight distribution between the two categories implies there is a greater importance to preserving the chemical environment. Later, it is shown that a good balance between the preservation of the chemical environment and partitioning the system into appropriately sized fragments is been achieved with these weights.

Table 3.3: The set of optimal $\{\beta_i\}$ values obtained from Bayesian optimisation.

β_{pe}	β_{conj}	β_{hyper}	β_{vol}	β_{comp}	β_{vrange}
0.135816	0.145943	0.313325	0.109416	0.001426	0.294074

The number of components/fragments exhibits the lowest weight of $\beta_{comp} = 0.001426$. The low weighting for β_{comp} is likely influenced by the inclusion of the volume term (p_{vol}) in Equation (3.3). As discussed in Section 3.2.3, the volume penalty term aims to penalise fragmentation instances where the fragments significantly deviate from the desired target size. Similarly, p_{comp} decreases in value as the number of fragments increases, serving a comparable purpose. These factors both encourage fragmentation, but with the presence of p_{vol}, the impact of p_{comp} diminishes. The comparative magnitudes of these weights, where $\beta_{vol} = 0.109416$ is larger than $\beta_{comp} = 0.001426$, underscores their primary function of reducing the extent of double-counting.

Due to the small magnitude of β_{comp} compared to the other $\{\beta_i\}$ values, a two-tailed *t*-test on the set of 800 systems (Dataset 1) is performed to examine its statistical significance. This involved comparing results obtained with $\beta_{comp} = 0.001426$ and $\beta_{comp} = 0$. For $\beta_{comp} = 0$, the remaining $\{\beta_i\}$ values were normalised to ensure $\sum_i \beta_i = 1$ (listed

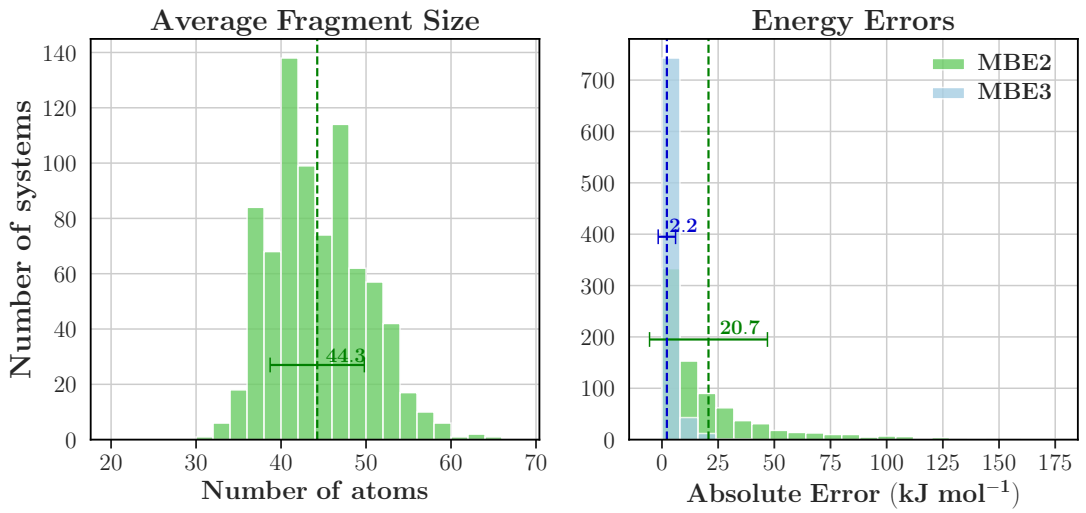


Figure 3.14: *left*: Distribution of the average fragment size of Dataset 1; *right*: Distribution of absolute energy errors at the MBE2 and MBE3 levels of Dataset 1. All energies were calculated at the HF/6-31G* level of theory. Averages are indicated by vertical lines and the corresponding values are reported. Error bars correspond to one standard deviation.

Table 3.4: The set of adjusted optimal $\{\beta_i\}$ values after removal of β_{comp}.

β_{pe}	β_{conj}	β_{hyper}	β_{vol}	β_{vrange}
0.136010	0.146151	0.313773	0.109573	0.294494

in Table 3.4). In particular, the quantity compared in the *t*-test is similar to Eq. (3.28), where each molecular system has a fitness value given by

$$f = \frac{1}{2}p_{vol} + \frac{1}{2}\mathcal{S}(\Delta E) \tag{3.30}$$

where p_{vol}, $\mathcal{S}$ and ΔE are evaluated identical to those in Eq. (3.28). A *t*-value of 0.014 is obtained and is substantially smaller than the critical *t*-value of 1.961 at the $\alpha = 0.05$ level. Consequently, due to the presence of β_{comp} being statistically insignificant at the $\alpha = 0.05$ level, p_{comp} is removed from the scoring function altogether and the weights listed in Table 3.4 is used for the remainder of this Chapter.

Since the weights of the scoring function terms were optimised on Dataset 1, the following text concerns the application of QFRAGS with the optimised $\{\beta_i\}$ values to Dataset 1.

Fig. 3.14 shows the distribution of the average fragment size as well as the energy errors obtained with MBE truncated at the two-body and three-body levels for Dataset 1. The vast majority of systems in Dataset 1 (81.9%) exhibited average fragment sizes ranging between 35 and 50 atoms. Furthermore, 83.0% of systems exhibited average fragment sizes less than the target size of 50 atoms. These resulting fragment sizes are encouraging for the purposes of this work; the standard deviation of 5.5 atoms is relatively

small (approximately 10% of the target fragment size) and the majority of the average fragment sizes do not exceed the target fragment size. Exceeding the target fragment size can be problematic due to the growing size of larger fragments (e.g. dimers and trimers) which can lead to memory and convergence issues in fragmentation-based QM calculations. It is demonstrated later in Section 3.4.2 that the distribution of the average fragment size narrows with larger system sizes (above 500 atoms).

Regarding the accuracy of total energies, Dataset 1 exhibits relatively low error margins. The mean absolute errors (MAE) are 20.7 and 2.2 kJ mol^{-1} for MBE2 and MBE3, respectively. At the MBE3 level, 84.5% of the systems yielded errors smaller than 4.2 kJ mol^{-1}, compared to 29.0% for MBE2. The significant improvement in error rates with MBE3 is expected, as MBE3 accounts for more chemical interactions by incorporating trimers.

The set of $\{\beta_i\}$ values in Table 3.4 is used in the current implementation of the proposed automated fragmentation algorithm. The subsequent section reports the results obtained from applying QFRAGS to both Dataset 2 and Dataset 3.

3.4.2 Application of QFRAGS

Datasets

In this Section, QFRAGS with the optimised $\{\beta_i\}$ values in Table 3.4 is applied to Datasets 2 and 3. The ability of the automated fragmentation procedure to generate fragment sizes close to the input target fragment size is demonstrated as well its accuracy by the comparison of the single point energies obtained with and without fragmentation. The rationale for employing two distinct test datasets is to examine the impact of system size on energy deviations and fragment dimensions.

Fragment Size

Figure 3.3 presents the size distribution of protein systems in Datasets 2 and 3. Dataset 2, with a maximum of 408 atoms, features smaller systems in comparison to Dataset 3, where the largest structure includes 1,396 atoms.

The distributions of average fragment sizes for Dataset 2 and Dataset 3 are shown in Fig. 3.15. In Dataset 2, a predominant proportion (81.5%) of systems display mean fragment sizes ranging from 35 to 50 atoms. Conversely, Dataset 3's distribution is narrower, with 85.0% of its systems having average fragment sizes within the 40 to 50 atom range, compared to only 65.0% in Dataset 2. This variance in distribution patterns is further evident in their standard deviations: Dataset 2 has a higher standard deviation of 5.3 atoms, while Dataset 3's is 3.2 atoms. These differences are attributable to the recursive fragmentation process and the presence of larger molecular systems in Dataset 3. Given that both datasets aim for a target fragment size of 50 atoms, Dataset 3 under-

goes more fragmentation recursions than Dataset 2. Additionally, the larger systems in Dataset 3 offer more possibilities for dividing the system into 50-atom fragments. Consequently, Dataset 3 exhibits a more concentrated distribution, closely aligning with the target fragment size.

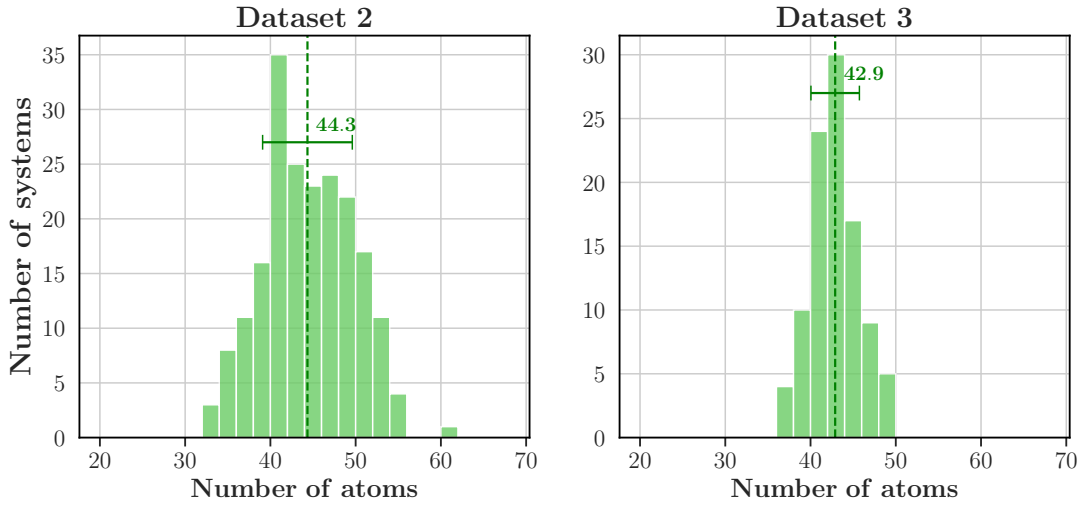


Figure 3.15: Average fragment size distribution of Dataset 2 and Dataset 3. Averages are indicated by vertical lines and the corresponding values are reported. Error bars correspond to one standard deviation.

For both datasets, there is a very small number of molecular systems that, when fragmented, exhibit mean fragment sizes greater than 50 atoms; this is true for 16.0% and 0.0% of systems in Dataset 2 and Dataset 3, respectively. This was also observed for Dataset 1 and the favourable implications of this were discussed earlier in Section 3.4.1.

Single Point Energies

Using the fragments produced by QFRAGS, the total energy of molecular systems in Datasets 2 and 3 was calculated at the HF/6-31G* level using the many body expansion method (Eq. (2.46)). The MBE calculations were truncated at the two-body and three-body levels. Subsequently, energies derived from fragmentation were compared with those obtained from full system (unfragmented) calculations at the same HF/6-31G* level.

Figure 3.16 presents the distributions of absolute errors at the MBE2 and MBE3 levels. For both datasets, a noticeable reduction in the absolute error is observed as the MBE level increases from dimers to trimers. This reduction is exemplified by the change in MAEs when transitioning from MBE2 to MBE3. Specifically, in Dataset 2, the MAE decreases from 20.0 to 2.2 kJ mol^{-1} and in Dataset 3, the MAE reduces from 181.5 to 24.3 kJ mol^{-1}. This improvement in energy accuracy is anticipated and can be

attributed to the inclusion of interaction energies in trimers. These findings align with existing literature on hierarchical fragmentation methods.[189–192]

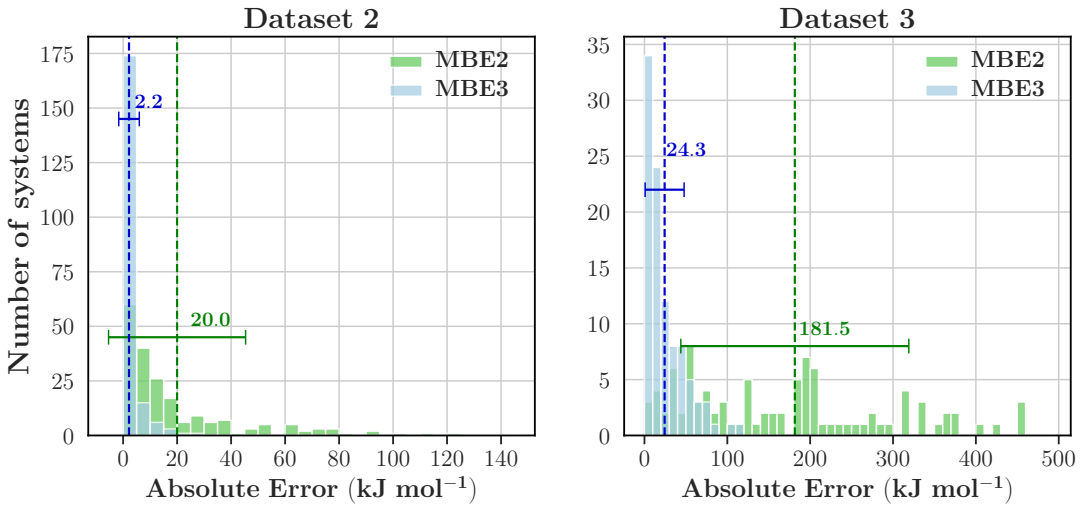


Figure 3.16: Distribution of absolute energy errors at the MBE2 and MBE3 levels for Dataset 2 and Dataset 3. All energies were calculated at the HF/6-31G* level of theory. Averages are indicated by vertical lines and the corresponding values are reported. Error bars correspond to one standard deviation.

Comparing the two datasets, Dataset 2 exhibits much lower errors than Dataset 3. The MAEs of Dataset 3 are 161.5 and 22.2 kJ mol^{-1} greater than those of Dataset 2 at the MBE2 and MBE3 levels, respectively. At the MBE3 level, 84.5% and 16.0% of systems in Dataset 2 and Dataset 3, respectively, achieved errors less than 4.2 kJ mol^{-1}. The higher errors and lower occurrence of accurate results in Dataset 3 are due to the prevalence of larger systems; the average system size in Dataset 3 is 1,021 atoms whereas the average system size in Dataset 2 is 277 atoms (see Fig. 3.3). The same target fragment size of 50 atoms was used to fragment systems in both datasets. With the systems in Dataset 3 being larger than those in Dataset 2, the number of fragments generated in Dataset 3 will be greater than those in Dataset 2. Correspondingly, more bonds are being broken in the systems belonging to Dataset 3, leading to larger absolute errors.

To better understand the system size's impact, the relative error results is included in Fig. 3.17. These errors are calculated by dividing the absolute error by the total electron count in the system. Fig. 3.17 shows the distribution of relative errors for both datasets.

The mean relative error in Dataset 3 at the MBE2 level is 0.051 kJ mol^{-1} per electron compared to the 0.018 kJ mol^{-1} of Dataset 2. Conversely, at the MBE3 level, both datasets exhibit the same relative error of 0.002 kJ mol^{-1} per electron.

Hence, when normalised for system size, at both MBE2 and MBE3 levels, relative errors for the two datasets remain within the same order of magnitude. This contrasts with the absolute errors, where Dataset 3's MAEs for both MBE2 and MBE3 were consistently larger than those of Dataset 2 by an order of magnitude. These findings sug-

gest that, by considering system size, QFRAGS can achieve comparable relative errors across a broad spectrum of system sizes, ranging from 158 to 1,396 atoms.

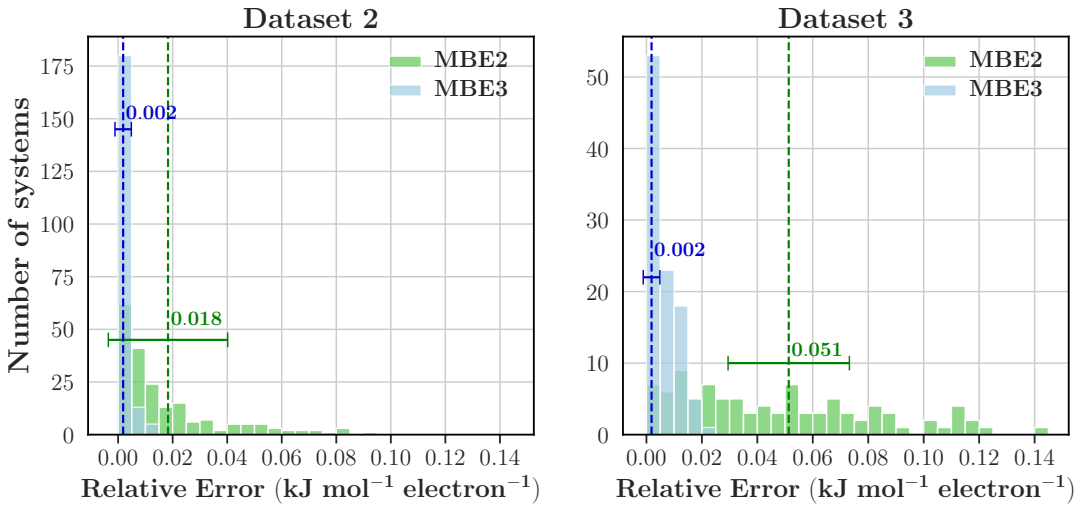


Figure 3.17: Distribution of relative energy errors at the MBE2 and MBE3 levels for Dataset 2 and Dataset 3. Averages are indicated by vertical lines and the corresponding values are reported. Error bars correspond to one standard deviation. See text for definition of relative errors.

3.4.3 Comparison to Manual Fragmentation

To demonstrate the advantage of the proposed fragmentation scheme, the results of QFRAGS are compared on two samples of 20 protein systems randomly selected from Dataset 1 and Dataset 2 to three manual fragmentation approaches specific to protein systems.

The manual fragmentation approaches will be called naive, semi-naive, and non-naive and corresponding fragments are obtained by severing the C−N amide, C_α−N, and C_α−C bonds, respectively. Figure 3.18 illustrates the three different manual fragmentation schemes. These three fragmentation approaches have been explored within literature and it has been consistently demonstrated that the order of increasing accuracy generally follows the order of cutting the C−N amide, C_α−N, and C_α−C bonds.[193–195]

To control for fragment size, as part of the criteria for bond breaking a fragment size of 50 atoms was selected to match the target fragment size used in QFRAGS.

Figure 3.19 shows the distribution of average fragment sizes for the two samples of 20 protein systems using the four distinct fragmentation schemes. For the systems from Dataset 1, the naive and non-naive methods yield fragments averaging between 40 and 48 atoms in size. Similarly, the semi-naive method produces fragments with average sizes ranging from 38 to 49 atoms. In contrast, the QFRAGS approach results in fragments averaging between 36 and 50 atoms. The fragment sizes derived from the three

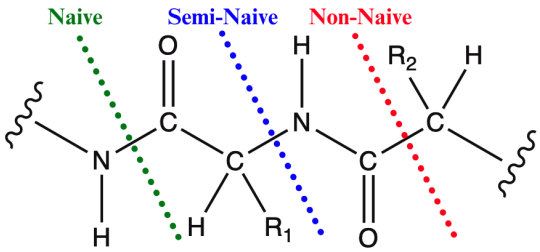


Figure 3.18: Definition of the three different manual fragmentation schemes employed for the 20 protein systems: naive, semi-naive and non-naive. R_1 and R_2 denote arbitrary side groups of amino acids.

manual methods (naive, semi-naive, and non-naive) are more closely aligned with the target size of 50 atoms compared to QFRAGS. A similar outcome can be observed for structures of Dataset 2 where the range of fragment sizes of QFRAGS is larger than those of the manual schemes. Nonetheless, there is a substantial overlap in the average fragment sizes among protein systems across all fragmentation schemes.

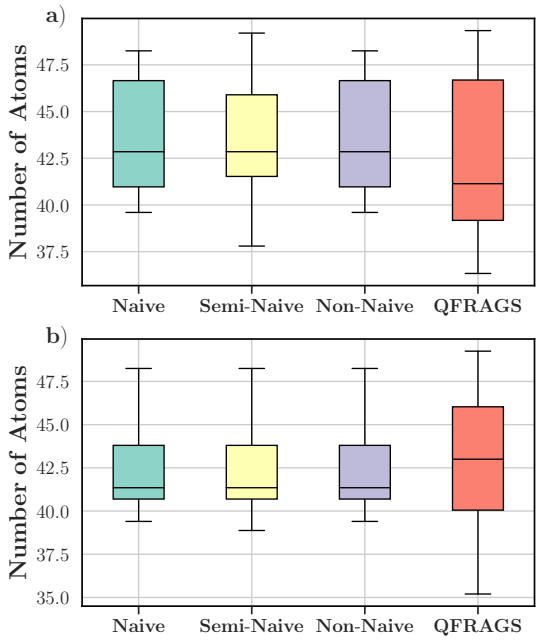


Figure 3.19: Distribution of average fragment size obtained from fragmenting 20 proteins randomly selected from a) Dataset 1; and b) Dataset 2 using the various bond breaking schemes: naive, semi-naive, non-naive and QFRAGS.

Figure 3.20 shows the mean absolute energy errors calculated using two fragmentation methods (MBE and FMO) at the dimer and trimer levels across the four fragmentation schemes for structures of Dataset 1 and Dataset 2. Across all 40 protein systems, the MBE2 and MBE3 MAEs across the four fragmentation schemes are all within the same order of magnitude. The MBE2 MAEs range between 41.8 and 58.5 kJ mol^{-1}, and the MBE3 MAEs range between 3.2 and 9.9 kJ mol^{-1} for systems in Dataset 1. Whilst

for Dataset 2, MBE2 MAEs ranged between 19.4 and 42.5 kJ mol^{-1} and MBE3 MAEs ranged between 2.0 and 4.0 kJ mol^{-1}.

It should be noted that since the weights of the scoring function were trained on systems belonging to Dataset 1 with MBE, there will be some bias in the MBE results from QFRAGS. It is for this reason that systems from Dataset 2 are included. The above results demonstrate the similarity in the behaviour of MBE for systems belonging to and outside of the training set (Dataset 1).

For Dataset 1 structures the FMO errors associated with the non-naive and QFRAGS are consistently an order of magnitude smaller than the naive and semi-naive schemes. Specifically, the FMO2 MAEs of the naive (55.1 kJ mol^{-1}) and semi-naive (74.9 kJ mol^{-1}) are both an order of magnitude larger than those of the non-naive (3.3 kJ mol^{-1}) and QFRAGS (8.5 kJ mol^{-1}). This also holds true for FMO3 where the MAEs of the naive (2.5 kJ mol^{-1}) and semi-naive (1.1 kJ mol^{-1}) fragmentation schemes are greater than those of the non-naive (0.4 kJ mol^{-1}) and QFRAGS (0.4 kJ mol^{-1}) schemes.

Compared to the FMO results of structures belonging to Dataset 1, those of Dataset 2 contrast in two ways. Firstly, the FMO2 MAEs of all three manual fragmentation schemes are an order of magnitude larger than QFRAGS. Secondly, the FMO3 MAEs of semi-naive, non-naive and QFRAGS are all within the same order of magnitude (less than 1 kJ mol^{-1}). However, these differences only occur for the semi- and non-naive manual fragmentation schemes; the behaviour of QFRAGS is consistent across systems in Dataset 1 and Dataset 2.

These differing results between MBE and FMO, specifically, the errors of FMO being generally lower than those of MBE, highlight the importance of the treatment of bond breaking and inclusion of electrostatic potentials in fragmentation methods. MBE fragment calculations do not employ electrostatic potentials and use hydrogen capping to restore valence at the site of broken bonds. The inclusion of hydrogen caps has the potential to perturb the electronic environment and introduce spurious steric effects.[196] Meanwhile, in FMO, fragment energy calculations are performed in the electrostatic potential of surrounding fragments, and furthermore FMO avoids introducing hydrogen caps, and instead the AFO approach is used to treat the broken bonds.[162,197] The effects of such variability between MBE and FMO are best exemplified through the three-body errors of $2LTX_2$ fragmented with the non-naive approach, where the MBE3 error (94.2 kJ mol^{-1}) is two orders of magnitude larger than that of FMO3 (0.2 kJ mol^{-1}).

Furthermore, the MAEs of FMO3 across all four fragmentation schemes are consistently lower than those of MBE3. On the other hand, the FMO2 MAEs are either the same order of magnitude or an order of magnitude less than those of MBE2. Such observations are consistent with the literature on fragmentation methods concerning electrostatic potentials; methods that include electrostatic potentials typically outperform those lacking it,[191,198,199] albeit being more computationally demanding than the

latter.

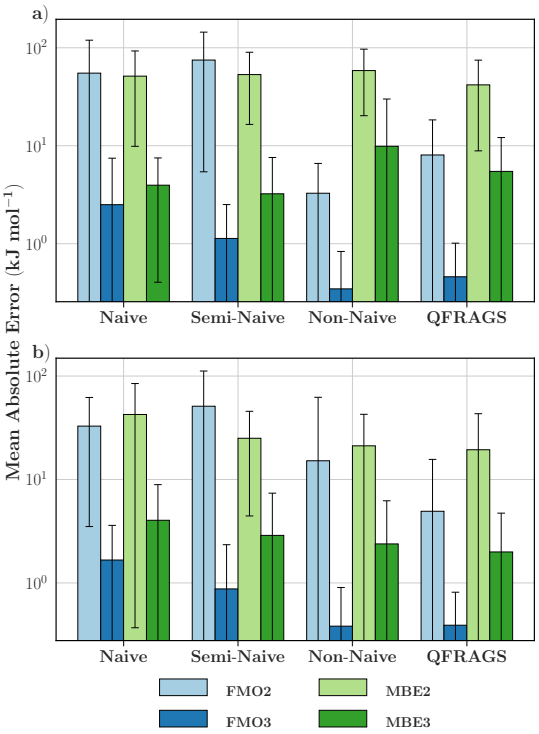


Figure 3.20: Mean absolute energy errors of various bond breaking schemes (naive, semi-naive, non-naive and QFRAGS) with the fragmentation methods FMO2/3 and MBE2/3 levels across 20 protein systems randomly selected from a) Dataset 1; and b) Dataset 2. Error bars correspond to one standard deviation. See text for system description of the various bond breaking schemes.

Furthermore, it is important to recognise that the three manual approaches (naive, semi-naive and non-naive) are suited to protein systems only, whereas QFRAGS possesses no information on the amino acid makeup of the protein systems. On the other hand, the naive, semi-naive and non-naive approaches are specifically tailored to amino acids because these schemes only consider breaking bonds that are found in protein systems. Yet despite this lack of amino acid information, the proposed QFRAGS method is able to achieve MAEs of the same order of magnitude as the three manual fragmentation methods at both the MBE2 and MBE3 levels, outperform the naive and semi-naive schemes with two- and three-body FMO calculations and is comparable to the accuracy of the non-naive scheme with both FMO2 and FMO3.

3.4.4 Application to Glycolipids and Lipoglycans

To demonstrate the applicability of QFRAGS beyond protein systems, it was applied to a set of 10 glycolipid and lipoglycan systems ranging between 368 and 727 atoms, with an average of 455 atoms. Unlike proteins, which are composed of well-defined monomeric units (amino acids), these structures lack an intuitive monomeric unit due

to the varied lipid component and consequently their manual fragmentation poses a challenging task. Consequently, these structures were selected to analyse the performance of QFRAGS.

Table 3.5: Performance of QFRAGS for glycolipid and lipoglycan systems. MBE2/3 errors are reported in kJ mol^{-1}. System name corresponds to the HMDB ID.

System	Average Fragment Size	MBE2 error	MBE3 error
0011945	41.8	-14.6	-0.2
0011957	40.9	7.8	0.4
0011959	41.2	-1.3	0.0
0012117	42.3	-8.6	-0.2
0012121	47.8	3.7	-0.1
0012123	47.1	3.0	0.5
0012124	44.5	0.3	-0.1
0012125	51.8	6.4	-1.0
0012232	48.7	5.0	0.0
0013470	40.4	27.9	0.1

Table 3.5 summarises the results of applying QFRAGS to the glycolipid and lipoglycan systems including the average fragment size and MBE2/3 errors. With the exception of one system, the average fragment size is consistently less than the target fragment size of 50 atoms. In fact, the majority of systems (60%) exhibit average fragment sizes ranging between between 40 and 45 atoms.

Concerning the energy errors, the largest MBE2 error (27.9 kJ mol^{-1}) belongs to largest system (727 atoms) and this error reduces to 0.1 kJ mol^{-1} with the inclusion of trimer energies. Across the 10 systems, the MAE decreases from 7.9 kJ mol^{-1} at the two-body level to 0.3 kJ mol^{-1} at the three-body level. Once again, this decrease in the MAE is ascribed to the inclusion of interaction energies of trimers. Such results highlight the accuracy of QFRAGS even on systems beyond proteins, the systems where the weights of Eq. (3.3) were trained on. With the results above, together, these outcomes demonstrate QFRAGS' capability of generating fragments of a specific size, its accuracy, and its applicability beyond protein systems.

3.5 Conclusion

In this Chapter, an innovative, automated molecular fragmentation approach, QFRAGS, is introduced, characterised by its evolutionary optimisation of a scoring function. The proposed approach hinges on three main innovations.

First, the fragmentation process is fully automated, eliminating the need for manual intervention.

Second, traditional energy metrics, which are impractical for large molecular systems, are replaced by a multi-factor scoring function. This function integrates chemical information and implementation aspects, offering a more feasible and effective alternative for fragmenting complex systems.

Third, the proposed approach employs evolutionary strategies to optimise the scoring function. This actively seeks out fragmentation schemes that indirectly minimise energy discrepancies when compared to the energy of the unfragmented, reference system.

The scoring function's weights were fine-tuned using 800 protein systems, each comprising 108 to 455 atoms. Using the optimised weights, QFRAGS was then applied to over 1,000 protein systems with atom counts ranging from 108 to 1,396 atoms, targeting fragment sizes of 50 atoms. For systems with less than 500 atoms, the mean fragment sizes achieved with QFRAGS varied between 32 and 65 atoms. In larger systems (505 to 1,396 atoms), the average fragment sizes improved, ranging between 37 and 50 atoms. These results show QFRAGS's efficiency in generating fragments that align well with the desired target size.

Using the fragments generated by QFRAGS, total energies were calculated at the two-body and three-body levels using the Many Body Expansion method, with HF/6-31G* as the theory level. The mean absolute errors for systems less than 500 atoms were 20.6 and 2.2 kJ mol^{-1} at the MBE2 and MBE3 levels, respectively. For larger systems (505 to 1,396 atoms), the MAEs increased to 181.5 and 24.3 kJ mol^{-1} at the MBE2 and MBE3 levels, respectively.

Then, a comparison of QFRAGS to three manual fragmentation approaches (naive, semi-naive and non-naive) specific to protein systems was performed on 40 protein structures ranging between 170 and 400 atoms. Total energies were calculated with two fragmentation methods: the Many Body Expansion and the Fragment Molecular Orbital, both at the two-body and three-body levels. All fragmentation strategies generated fragments with similar average sizes close to the target fragment size of 50 atoms. With MBE, the MAEs across QFRAGS and the three manual fragmentation schemes are all within the same order of magnitude. At the MBE2 level, MAEs ranged between 19.4 and 58.5 kJ mol^{-1}, meanwhile at MBE3 level, the MAEs fell between 2.0 and 9.9 kJ mol^{-1}. On the other hand with the fragment molecular orbital method, the accuracy of QFRAGS (FMO2 and FMO3 MAEs of 6.6 and 0.4 kJ mol^{-1}, respectively) are comparable that of the non-naive scheme (FMO2 and FMO3 MAEs of 9.2 and 0.4 kJ mol^{-1}, respectively) at both the two- and three-body levels. Both of these schemes are consistently an order of magnitude less than the corresponding MAEs of the naive (FMO2 and FMO3 MAEs of 43.4 and 2.1 kJ mol^{-1}, respectively) and semi-naive (FMO2 and FMO3 MAEs of 63.0 and 1.0 kJ mol^{-1}, respectively) approaches.

Following, QFRAGS was applied to 10 glycolipid and lipoglycan systems to demon-

strate its applicability beyond protein systems, namely, on structures without explicit monomeric units, and yielded MAEs of 7.9 and 0.3 kJ mol^{-1} at the two- and three-body levels with MBE, respectively.

The results of this study demonstrate the efficacy of the newly proposed automated fragmentation scheme in various aspects. QFRAGS is capable of generating fragments that closely match the desired size. When integrated with MBE and FMO fragmentation methods and applied to protein systems, it achieves an approximation of the total energy that rivals that of manual, non-naive fragmentation. Furthermore, QFRAGS is generalisable to organic systems beyond proteins. Finally, this study corroborates the importance of employing high-quality fragments and carefully selecting the bonds to be broken in molecular fragmentation approaches. Overall, this Chapter presents a new, accurate, automated molecular fragmentation scheme suitable for the application of QM on large molecular systems, advancing strategies to reduce computational cost and enhance algorithmic parallelisability.

Acceleration of Self-Consistent Field Calculations Using Basis Set Projection and Many-Body Expansion as Initial Guess Methods

4.1 Introduction

The research explored in Chapter 3 addresses limitations of existing molecular fragmentation methods in their application to large molecular systems. As described previously, fragmentation methods alleviate the computational burden of QM calculations by partitioning the molecular system into smaller units (*i.e.* fragments). Another avenue of accelerating QM calculations on large molecular systems (without fragmentation) involves improving initial guesses utilised in Self-Consistent Field (SCF) calculations to reduce the iteration count. A description of the SCF procedure is provided in Chapter 2 Section 2.4.1.

The SCF procedure is at the core of electronic structure methods used to solve the Hartree-Fock (HF) and Kohn-Sham equations.

All SCF calculations begin at the same starting point: an initial guess for the density matrix or for the molecular orbitals of the chemical system. The accuracy of the starting guess dictates the SCF procedure's convergence behaviour; the closer the initial guess is to the final solution, the faster the convergence. Additionally, a poor initial guess can lead to convergence failure or convergence to different, undesired solutions.

One of the most commonly adopted initial guess methods is the Superposition of Atomic Densities (SAD).[200,201] In SAD, the initial density matrix assumes a block diagonal structure, assembled using spherically-averaged converged atomic densities. Despite its transferability and ease of implementation, SAD typically faces convergence issues with larger systems. For instance, in a study focused on protein systems, SAD guesses resulted in convergence failures for several structures comprising between 396 and 598 atoms.[202] Additionally, its implementation is often charge neutral,[203] making it

incompatible with many ionic species. Namely, the SAD matrix is not consistent with the charge state of the species with only some implementations allowing users to specify charge states. Furthermore, the SAD method yields a non-idempotent matrix and therefore requires a minimum of two SCF iterations for convergence.

As as alternative to SAD, other initial guess methods have instead focused improving the accuracy of initial guesses. Two families of such methods include the basis set projection (BSP) and molecular fragmentation approaches.

In BSP methods, the density matrix is constructed by projecting the converged density calculated in a smaller basis set into a larger target basis set. In this work the smaller basis set is referred to as the "bootstrapping basis" and the target basis set as the "projected basis".

Fragmentation methods, on the other hand, involve dividing the system into smaller units and assembling the initial guess matrix using the converged densities of these fragments. Multiple studies have reported greater accuracy and fewer iterations associated with BSP and fragmentation methods.[192,202–209] For example, Hegely and Kallay demonstrated that using def2-SVP as the bootstrapping basis set for triple-ζ calculations in BSP led to a decrease in the number of SCF iterations by up to 10 iterations compared to SAD, which had an average SCF iteration count of 20.[208] Additionally, fragmentation methods have been shown to reduce the number of SCF iterations by up to 40% relative to SAD and greater for SAP.[192]

Recently, Ballesteros *et al.* proposed a new fragmentation-based initial guess scheme employing the many body expansion.[192] This MBE-density matrix (MBE-DM) scheme, applied to water cluster systems, exhibited superior performance compared to BSP and SAD, typically requiring 40% fewer SCF cycles. The MBE-DM method was also generalised to the General Many Body Expansion (GMBE) scheme, where overlapping fragments are used to construct the density matrix.[192]

In most studies evaluating initial guess performance, the primary efficiency metric is the number of SCF iterations. However, this measure alone does not capture the computational overhead associated with generating the initial guess itself. Consequently, an improved-guess method that reduces SCF iterations may still result in its overall SCF taking longer than the corresponding SAD-based SCF calculation if its guess-generation step is particularly time-consuming. Thus, in addition to tracking SCF iterations, it is essential to consider the total time-to-solution as a performance metric, encompassing both the time required to produce the initial guess and the time required for the SCF calculation of the target system to converge.

In this Chapter, an investigation on the acceleration of SCF calculations through the use of improved initial guess methods is conducted, specifically focusing on Basis Set Projection and on the Many Body Expansion, in comparison to the standard SAD approach. A detailed benchmark analysis of the time-to-solution for both Hartree-Fock

and hybrid Density Functional Theory calculations is presented, exploring the performance of BSP, MBE, and SAD individually, as well as their combinations. The analysis spans a diverse set of molecular systems, ranging from 42 to 1,107 atoms (400 to 14,386 basis functions), and includes both fully connected covalently bonded systems and non-bonded molecular clusters, offering a detailed evaluation of these methods across a variety of chemical environments.

Unless stated otherwise, all algorithms were implemented in, and tested using the GPU-accelerated Extreme-scale Electronic Structure System (EXESS) quantum chemistry program.[22–29]

This Chapter begins with Section 4.2 where the various initial guess methods explored in this study are detailed. In Section 4.3.2, a description of the molecular systems on which these methods were tested. Sections 4.4.1 and 4.4.2 examine the influence of parameters such as bootstrapping basis set selection, fragment sizes, fragmentation levels, convergence thresholds, and approximate HF methods (resolution-of-the-identity) on the performance of the BSP and MBE methods for various molecular systems. Finally, in Section 4.4.3, the performance of multiple initial guess methods (SAD, BSP, and MBE) are compared using the optimal parameters identified for BSP and MBE. The energy errors, wall-time speedups, and number of SCF iterations required are all analysed. Additionally, a new initial guess method is introduced, Basis Set Projection with Fragmentation (BSPF), which is a hybrid of the BSP and MBE approaches, and assess its performance against SAD, BSP, and MBE. Section 4.5 concludes.

This chapter addresses Aim 2 and reproduces a published manuscript: **Yu, F.C.**, Seidl, C., Palethorpe, E. and Barca, G.M., 2025. Acceleration of Self-Consistent Field Calculations Using Basis Set Projection and Many-Body Expansion as Initial Guess Methods. *Journal of Chemical Theory and Computation, 21*(3), pp.1230-1248.

4.2 Initial Guess Methods

In this section, a description of the four initial guess methods examined in this study are provided: the superposition of atomic densities, basis set projection, and many body expansion-density matrix approaches. Additionally, a novel method is introduced, basis set projection with fragmentation (BSPF), which combines elements of the latter two methods.

A description of the SCF procedure is provided in Chapter 2 Section 2.4.1 to clarify the role of initial guesses in SCF calculations.

4.2.1 Superposition of Atomic Densities

In the SAD method, the initial density is constructed on an atom-by-atom basis. For each atom and basis set, a unrestricted/restricted open-shell HF computation is performed depending on the implementation. The resulting atomic density is spherically averaged and the initial density is then assembled from the converged densities of these atomic unrestricted open-shell HF calculations, resulting in a block diagonal structure. Specifically, the initial density matrix D is the direct sum of the atomic converged densities, taking on the following block diagonal form:

$$D = \begin{pmatrix} D_0 & 0 & \cdots & 0 \\ 0 & D_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_n \end{pmatrix} \tag{4.1}$$

where D_i is the converged density of atom i.

4.2.2 Basis Set Projection

In the BSP method, the initial density is obtained using the converged density matrix in the bootstrapping basis set. In the bootstrapping basis set, an occupied molecular orbital is defined as

$$|\psi\rangle = \sum_i c_i |i\rangle \tag{4.2}$$

where c_i are the coefficients and $|i\rangle$ denote the atomic basis functions. To project $|\psi\rangle$ to the target basis set $|J\rangle$, the following formula is used

$$|\psi\rangle = \sum_J c_J |J\rangle \tag{4.3}$$

where

$$c_J = \sum_{JK} \sum_i \langle J|K\rangle^{-1} \langle K|i\rangle c_i \tag{4.4}$$

where both J and K denote basis functions of the target basis set, i denotes basis functions of the bootstrapping basis set, and $\langle J|K\rangle^{-1}$ denotes elements of the inverse overlap matrix.

4.2.3 Many Body Expansion-Density Matrix

As presented in Section 2.4.5, the many body expansion scheme (Eq. (2.46)) is used to compute the total energy of a fragmented system. Ballesteros and Lao extended the MBE method to form initial guess density matrices[192] using the fragment densities as

follows:

$$D = \sum_i D_i + \sum_{i<j} \Delta D_{ij} + \sum_{i<j<k} \Delta D_{ijk} + \dots \tag{4.5}$$

Here, D_i is the density of monomer i, and the higher-order correction terms ΔD_{ij} and ΔD_{ijk} are defined analogously to Eqs. (2.47) and (2.48):

$$\begin{aligned} \Delta D_{ij} &= D_{ij} - (D_i \oplus D_j) \\ \Delta D_{ijk} &= D_{ijk} - (\Delta D_{ij} \oplus \Delta D_{ik} \oplus \Delta D_{jk}) \\ &\quad - (D_i \oplus D_j \oplus D_k) \end{aligned} \tag{4.6}$$

Here, D_{ij} and D_{ijk} denote the densities of the dimer and trimer systems, respectively, and $\oplus$ denotes the direct sum of matrices.

4.2.4 Basis Set Projection with Fragmentation

A new initial guess method is introduced, named Basis Set Projection with Fragmentation, a hybrid of the MBE and BSP methods.

There are various ways to integrate MBE with BSP. Figure 4.1 outlines the possible combinations of algorithmic stages, with those highlighted in blue representing the BSPF implementation used in this work. Specifically, stages labeled 1 and 2 are excluded from the proposed implementation.

BSPF begins with partitioning the system into fragments and performing SCF calculations on fragments in the bootstrapping basis set.

Stage 1 includes combining the converged fragment densities in the bootstrapping basis to form the initial guess of the full-system and performing an SCF calculation on the latter in the bootstrapping basis. We avoid this step since the fragmentation approach only serves to accelerate the full-system SCF computation in the bootstrapping basis set whereas the main bottleneck is associated with the latter but in the projected basis set.

Rather than stage 1, the converged densities of the fragments in the bootstrapping basis are projected to the target basis set and these are used to assemble the full-system initial guess matrix.

Stage 2 comprises performing SCF on each fragment in the projected basis set and using the converged densities from the bootstrapping basis calculation as the initial guesses. Whilst this is likely to result in a more accurate initial guess of the full-system, it is associated with a notable rise in the overhead of constructing the initial guess due to an extra round of fragment SCF calculations. Furthermore, given that fragments will be of the order of ten atoms and relatively small, the speedup afforded by BSP on the fragment SCF computations in the projected basis set is unlikely to be significant enough to outweigh the reduced SCF timing of the full-system due to the more accurate

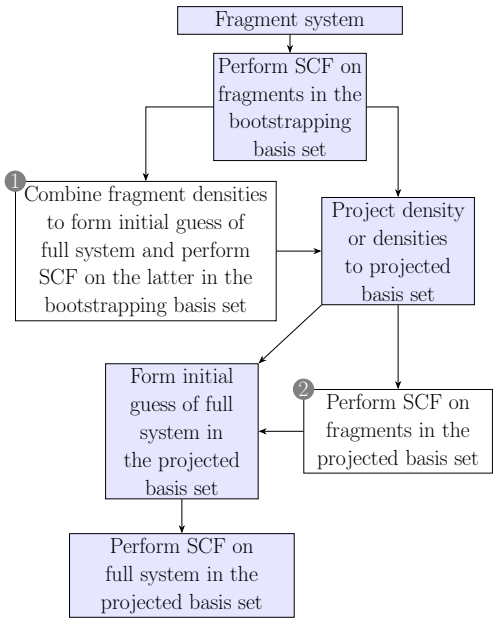


Figure 4.1: Algorithmic stages resulting from the integration of BSP and MBE. Stages highlighted in blue constitute the BSPF algorithm used in this work, while those marked with 1 and 2 are excluded.

initial guess.

With stage 2 excluded, the initial density matrix is formed directly from projecting the converged fragment densities in the bootstrapping basis. BSPF concludes with performing an SCF calculation on the full-system in the projected basis set.

For all BSPF calculations in this work, the optimal parameters obtained for BSP and MBE in Section 4.4.1 and 4.4.2, respectively, are employed.

4.3 Computational Details

4.3.1 Single Point Energies

In this study, single point energy (SPE) calculations on molecular systems are conducted to assess the speedup afforded by different initial guess schemes. Both HF and DFT SPE calculations were performed. An overview of HF and DFT methods can be found in Chapters 2 Section 2.4. For DFT calculations, the B3LYP[138–141] and Minnesota MN15[137] functionals were utilised, and the (99,590) Mura-Knowles[210] quadrature grid was employed along with GauXC's "robust" pruning scheme which matches the default pruning method of Psi4.[211,212] These SPE calculations were performed with a variety of double- and triple-ζ basis sets to demonstrate the generalisability of the results across various initial guess schemes. In particular, the benchmarks include Dunning's correlation consistent basis sets cc-pVDZ and cc-pVTZ,[213,214] the polarisation consis-

tent basis sets pc-1 and pc-2,[215–218] and the Karlsruhe basis sets def2-SVP and def2-TZVP.[219,220]

The direct inversion in the iterative subspace (DIIS) solver was used as the SCF solver for all calculations.[221,222] An SCF calculation on a full-system employing the aforementioned basis sets is considered converged if the DIIS error between two subsequent iterations is less than 10^{-6} a.u. DIIS extrapolations employed a maximum of 10 error vectors from previous iterations. A maximum iteration count of 100 was imposed; systems that failed to meet the convergence criteria with 100 iterations are considered to be not converged. This is especially pertinent to the case of metalloenzymes which are discussed in the next Section. An integral screening threshold of 10^{-10} a.u. was used for all calculations.

All closed-shell SPE calculations were performed using EXESS.[22–29] These include SAD, BSP, MBE and BSPF-based computations. All results were obtained on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center of the United States Department of Energy. All timing results were obtained on an NVIDIA DGX A100 node with 1×64-core AMD EPYC 7763 2.45 GHz, 256 GB of DDR4 memory and 4 NVIDIA Ampere A100 GPUs with each containing 80 GB of High-Bandwidth Memory.

All open-shell unrestricted calculations were performed with Q-Chem[223] version 6.0. These include only SAD and BSP computations. All timing results were obtained on the Gadi supercomputer at the National Computing Infrastructure on a 48-core Intel Cascade Lake node.

To assess the performance of the various initial guess schemes, the following speedup metric is used

$$\text{speedup} = \frac{t_{SAD}}{t} \tag{4.7}$$

where t_{SAD} and t denote wall-times of SAD and non-SAD SPE HF calculations, respectively. These are end-to-end times of the entire calculation inclusive of both the guess construction step and the SCF procedure on the full molecular system.

Superposition of Atomic Densities

The implementation of SAD guesses can vary across different software. In EXESS, the SAD guess is generated by performing an unrestricted open-shell HF computation. The number of unpaired electrons may vary across different systems. Details on the number of unpaired electrons used can be found in Appendix Bn. On the other hand, the SAD guess is obtained by a restricted open-shell HF calculation in Q-Chem. The atomic SAD guesses are pre-tabulated in both Q-Chem and EXESS and therefore, the set of SPE computations performed herein do not include those related to generating them.

Basis Set Projection

Eq. (4.4) defines the projection from one basis set to another. This involves the computation of the mixed overlap matrix for the two bases and is followed by an orthogonalisation step.

In BSP calculations, two parameters are varied: the bootstrapping basis set and the convergence threshold used in the SCF procedure performed with the bootstrapping basis set. The performance of BSP calculations is evaluated using four bootstrapping basis sets, 3-21G, 6-31G,[224–228] STO-3G, and STO-6G,[229,230] and multiple convergence thresholds, 10^m a.u. where $m \in \{-6, -5, -4, -3, -2, -1, 0, 1\}$. The convergence threshold refers to the DIIS error between two successive iterations.

Many Body Expansion-Density Matrix

In this paper, all many body expansion computations are performed using the MBE-DM method, which will be simply referred as MBE for conciseness.

Three parameters in MBE computations are explored: 1) target fragment size (n_t); 2) the convergence threshold for the SCF procedure of the fragments; and 3) the fragmentation level. Six different target fragment sizes are examined, 10, 20, 30, 40, 50, 60 atoms, eight convergence thresholds, 10^m a.u. where $m \in \{-6, -5, -4, -3, -2, -1, 0, 1\}$, and two fragmentation levels (MBE1 and MBE2) to assess their impact on MBE computation performance. The varying convergence thresholds are applied only to the SCF calculations of the fragments.

For MBE2 calculations, two initial guess schemes for the dimer densities are explored: using the traditional SAD guess and employing the converged densities of the constituent monomers, termed MBE2(SAD) and MBE2(MD), respectively.

Additionally, the influence of varying cutoffs for dimer construction in MBE2 is explored. A dimer is assembled if the distance between the centroids of two monomers is less than or equal to the cutoff value. Three cutoffs are explored: 5 Å, 10 Å, and no cutoff, where all possible dimers are included.

In MBE calculations, all SCF calculations on fragments are performed sequentially on a single node (see Section 4.3.1), with initial guesses constructed from converged fragment densities.

To explore accelerating MBE-based calculations further, the resolution-of-the-identity (RI) Hartree-Fock (RI-HF) algorithm in EXESS for calculations of the fragments is employed.[231] RI-HF typically outperforms traditional HF in small systems, making it advantageous for accelerating single-fragment calculations. Therefore, RI-HF is explored for individual fragments required for building the MBE initial guess. A description of the RI approximation is provided in Chapter 2 Section 2.4.3.

RI-HF calculations are limited to Dunning's correlation consistent and the Karlsruhe

basis sets due to the availability constraints of the corresponding auxiliary basis sets.

To employ MBE, a set of fragments is required. An automatic fragmentation approach based on a target fragment size criterion is used, generating fragments close to n_t. Two fragmentation procedures specialised for both fully covalently bonded systems and non-covalent cluster systems are described. The full details of these fragmentation schemes can be found in Appendix B.

4.3.2 Molecular Dataset

Although this thesis focuses on protein systems, the goal of this Chapter is to provide researchers with initial guess tools suited for a range of system types besides proteins. More precisely, the focus is on molecular systems with significance across a range of disciplines such as biology, drug design and materials science. Additionally, a molecular system dataset is sought that exhibits the following attributes:

- Spans a large range of system sizes, specifically, between 50 and 1,000 atoms approximately
- Includes a combination of both covalently bonded and non-bonded systems
- Includes systems that possess neutral and charged moieties
- Includes systems that exhibit various bonding types such as hydrogen bonding, dispersion interactions and π-π stacking

To the best of our knowledge there is no existing datasets that satisfies this criteria and therefore a dataset of molecular structures is constructed herein.

Figure 4.2 summarises the classification of the full dataset of molecular structures used in this study. Structures are divided into two main bonding classes: bonded or cluster. The distinguishing feature between these two classes is the extent of covalent bonds and therefore whether or not bonds need to be severed to generate fragments in the case of the many body expansion implementation. The generation of fragments is described in Section 4.2.3.

Structures belonging to the bonded systems class are categorised into three types: proteins, nucleic acids and carbohydrates. Such systems are of biological relevance, and furthermore, their study is critical towards the design of bio-active drugs. On the other hand, non-bonded systems are grouped into four types, of which two are neutral systems (water and benzene) and two are ionic liquids: 1,3-Dimethylimidazolium dicyanamide ($[C1mim^+][DCA^-]$) and Guanidinium tetrafluoroborate ($[Gdm^+][BF_4^-]$). Such systems play an integral role in the activity and synthesis of drugs, as well as towards the development new materials.[232,233]

Furthermore, there are seven structures belonging to each of the above seven system types of varying size. These structures are approximately of the order 50, 150, 250, 400, 500, 750 and 1,000 atoms, giving a total of 49 systems in the dataset.

Calculations with double-ζ basis sets are performed on the full 49-system dataset. For these computations, the system with the most basis functions (12,019) includes 1,023 atoms. On the other hand, due to memory hardware constraints, single-point energy calculations employing triple-ζ basis sets are limited to 35 of the total 49 systems, excluding structures with approximately 750 and 1,000 atoms. The largest system in this 35-system dataset (509 atoms) includes 14,386 basis functions.

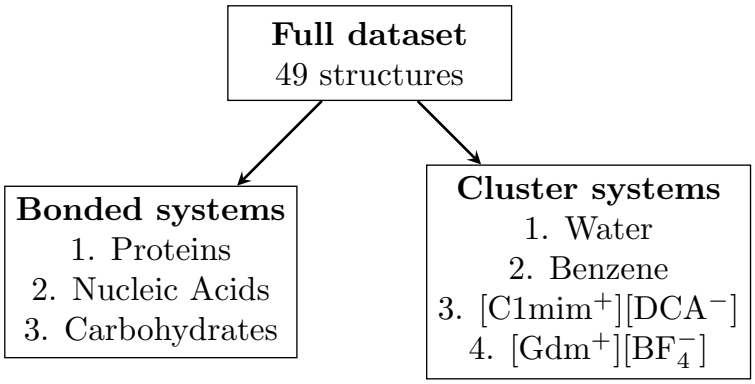


Figure 4.2: Classification of systems used in this study.

All protein and nucleic acid structures were taken directly from the RCSB Protein Data Bank.[234] All carbohydrate structures were obtained from Polysac3-DB.[235] All cluster structures are spherically shaped and were generated with its corresponding unit cell crystal structure. Unit cell geometries were obtained from the Cambridge Structural Database.[236–239] All 49 structures used in this study are available in Appendix B.

Beyond these systems, more complex systems are studied herein including metal atoms, for which SCF convergence is not always guaranteed. Specifically, a set of 20 metalloenzyme structures comprising between 334 and 1,084 atoms was selected. Such structures include metals such as zinc, calcium, sodium and magnesium. All structures were taken directly from the RCSB Protein Data Bank.[234] Double-ζ HF calculations are performed on the full 20-system dataset, where the system comprising the highest basis functions (10,884) includes 1,076 atoms. SPE calculations employing triple-ζ basis sets are limited to 10 of the 20 metalloenzyme systems due to memory constraints. The largest system in this 10-system dataset includes 503 atoms and 10,615 basis functions.

Furthermore, double-ζ calculations for the metalloenzyme structures are limited to def2-SVP and pc-1 as cc-pVDZ is not currently supported for transition metals in EXESS. On the other hand, triple-ζ calculations of the metalloenzymes are limited to def2-TZVP only as cc-pVTZ and pc-2 require g-type functions which are not yet sup-

ported in EXESS.

For the open-shell (triplet) states the original benchmark set of proteins, nucleic acids, carbohydrates, water clusters, benzene clusters, and ionic liquids was used. Due to memory and computational constraints, the molecular dataset is limited to systems with less than 600(300) atoms for double(triple)-ζ computations, and to the methods of HF and B3LYP. Correspondingly, the final dataset of triplet states comprises 35 and 21 systems for double- and triple-ζ basis sets, respectively.

4.4 Results and discussion

In this Section, the effect of parameters on the performance of the BSP and MBE initial guess methods, as detailed in subsections 4.4.1 and 4.4.2, is analysed. Subsequently, in Section 4.4.3, the performance of the BSP and MBE schemes are compared using optimal parameters with the SAD and BSPF initial guesses. All mentions of SCF iteration count refer to the SCF procedure on the full molecular system.

4.4.1 Basis Set Projection

In this section, the findings regarding the influence of bootstrapping basis sets and convergence thresholds on the performance of BSP are presented.

Bootstrapping Basis Set

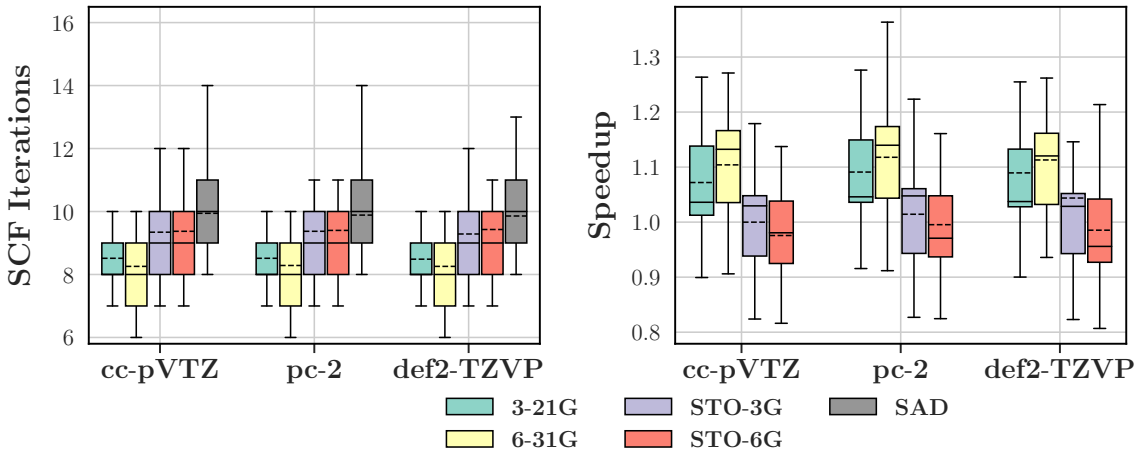


Figure 4.3: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set BSP calculations with various bootstrapping basis sets at the HF level. Average is indicated by broken horizontal line.

Figures 4.3 and B.2 to B.6 show the distribution in the number of SCF iterations and wall-time speedups relative to SAD using different bootstrapping basis sets for HF, B3LYP and MN15 methods, respectively.

For all basis sets and levels of theory, BSP calculations typically demonstrated lower SCF cycle counts compared to SAD. On average, double-ζ basis sets using HF, B3LYP and MN15 methods required at least 17.1%, 3.5% and 21.7% fewer iterations, respectively. Similar observations can be made of triple-ζ basis sets, where iteration counts of HF, B3LYP and MN15 were respectively on average at least 15.7%, 16.3% and 21.2% less. These results are consistent with those of Hégely and Kállay.[208] Furthermore, though this study does not consider augmented basis sets, similar trends in the iteration count can be observed with augmented basis sets as the projected basis.[208]

For all triple-ζ basis sets and all methods (HF, B3LYP and MN15), 6-31G as a bootstrap basis consistently exhibited the highest average wall-time speedups. For example with pc-2 as the main basis and 6-31G as bootstrap, HF and B3LYP BSP calculations displayed mean speedups of 11.8% and 2.6%, respectively. On the other hand, MN15/triple-ζ BSP computations employing the 6-31G bootstrapping basis were typically slower those of SAD (see Figure B.6). While this might suggest that BSP is less efficient for MN15, it is important to note that the speedup metric does not account for system size. As shown in Figure B.7 which illustrates the speedup of BSP calculations at the MN15 level across varying system sizes, the lower speedups are predominantly associated with structures belonging to the smallest class of systems (between 42 and 65 atoms) whereas average speedups up to 20.5% were observed for systems between 480 and 516 atoms.

Concerning the double-ζ basis sets, in terms of wall-time speedups, 3-21G was the best performing bootstrapping basis for HF and MN15, whereas STO-3G was the best for B3LYP. However, only HF exhibited improved wall-time speedups with BSP at all. Specifically, average speedups up to 3.4% could be observed for systems comprising between 996 and 1,107 atoms in HF, whereas in the best case scenarios, BSP computations with B3LYP and MN15 were on average 16.8% and 16.5% slower than SAD, respectively. This is a direct result of the computational cost of constructing the initial guess; the initial guess assembly step constitutes on average at least 24.5%, 34.1% and 39.7% of the total wall-time in HF, B3LYP and MN15 calculations, respectively.

Such results across the double- and triple-ζ basis sets highlight the importance of the bootstrapping basis set selection for the performance of BSP; some may lead to a worse performance than SAD, as in the cases of B3LYP and MN15 for the double-ζ basis sets, whilst others may only exhibit marginal speedups.

All BSP computations thus far utilised a convergence threshold of 10^{-6} a.u., and in the next section we aim to enhance the speedups by raising the convergence thresholds for SCF calculations performed in the optimal bootstrapping basis sets.

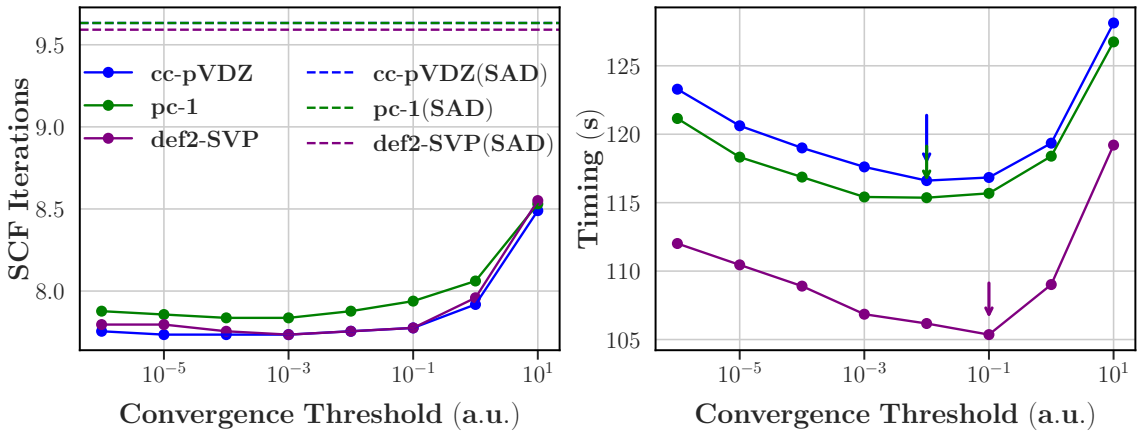


Figure 4.4: Average number of SCF iterations and wall-times of double-ζ basis set BSP calculations with varying fragment convergence thresholds at the HF level. Error bars are omitted for clarity. Arrows indicate the convergence threshold with the lowest average wall-time.

Convergence Threshold

Average number of SCF iterations and wall-times obtained across various fragment convergence thresholds are shown in Figures 4.4, B.9-B.13. In these computations, the bootstrapping basis set is held constant; the optimal bootstrapping bases reported in Section 4.4.1 are employed. The convergence threshold is held fixed at 10^{-6} a.u. for all calculations in the large basis set. Compared to SAD, a decrease in the count of SCF cycles is frequently observed across the majority convergence thresholds. Noticeably, there is a general rise in the number of SCF iterations with increasing (looser) convergence thresholds. For example at the HF level, as the convergence threshold rises from 10^{-6} to 10^{1} a.u. the average iteration count increases by at least 8.9% and 9.4% for double- and triple-ζ basis sets, respectively. Similar trends are also seen at the B3LYP and MN15 levels (see Figures B.10 to B.13). This is largely attributed to larger thresholds allowing for greater error margins, resulting in less accurate initial guesses.

Generally, improved wall-times can be attained with larger (looser) convergence thresholds, such as that exhibited in Figure 4.4. Employing the optimal thresholds leads to only marginal improvements and therefore are not reported here. The full discussion and optimal thresholds can be found in Appendix B.

The results summarised above highlight the importance of the selection of bootstrapping basis sets and convergence thresholds on the performance of the BSP scheme. Increasing the convergence threshold from the default value of 10^{-6} a.u. can yield improved speedups, though marginal. Furthermore, whilst lowering the number of SCF iterations is a primary objective with these non-SAD initial guess schemes, consideration of the computational cost of constructing the initial guess is also critical as we encountered cases where BSP was more computationally costly than SAD.

From the outcomes discussed above, the optimal BSP parameters, *i.e.* bootstrapping basis and convergence threshold, are employed for the calculations reported in Section 4.4.3 where the performance of multiple initial guess approaches are compared.

4.4.2 Many Body Expansion

In this section, the findings regarding the influence of fragment sizes, convergence thresholds and fragmentation levels on the performance of MBE calculations as an initial guess method in comparison to SAD are presented.

Fragment Size

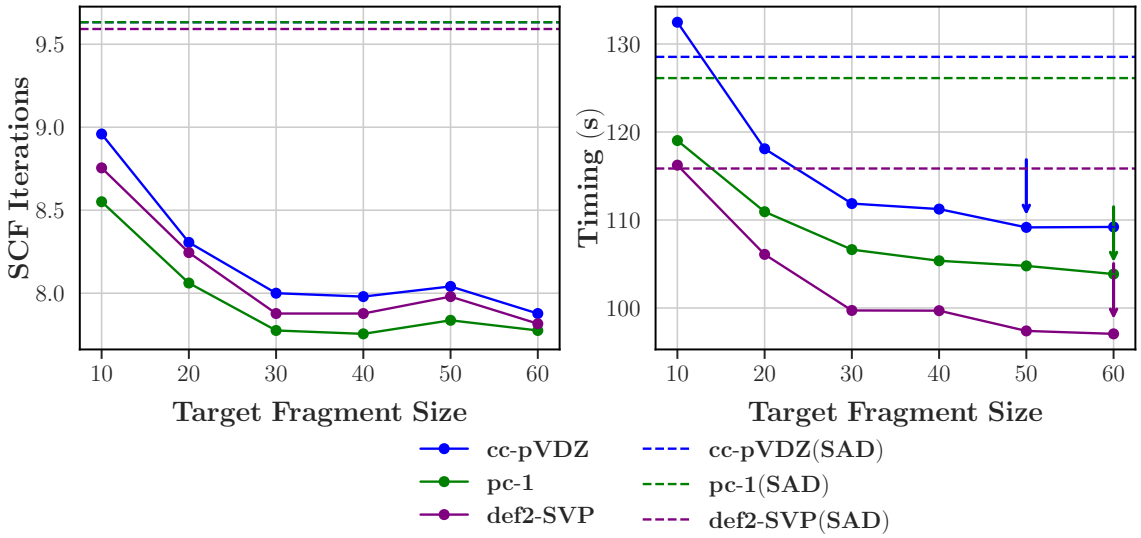


Figure 4.5: Average number of SCF iterations and wall-times of double-ζ basis set MBE calculations with varying target fragment sizes (no. of atoms) at the HF level. Arrows indicate the fragment size with the lowest average wall-time.

Figures 4.5, B.14-B.18 show the variation in the average SCF iteration count and wall-times with respect to target fragment size of MBE1 calculations. All computations were performed with a convergence threshold of 10^{-6} a.u. Across all MBE calculations, all target fragment sizes greater than 20 atoms required fewer SCF cycles than that of SAD. This is also reflected in the wall-time speedups where for example in the HF calculations, with a target fragment size of 30 atoms, average speedups up to 9.2% and 1.6% could be observed for double- and triple-ζ basis sets, respectively.

Generally in MBE computations, the SCF iteration count decreases as the target fragment size increases. Representative examples of this include HF/pc-1 and MN15/cc-pVDZ computations as shown in Figures 4.5 and B.17; the increase in target fragment size from 10 to 60 atoms corresponded to a reduction in the SCF iteration count of 9.1%

and 16.2%, respectively. The decreasing cycle count with fragment size is a direct consequence of fragment size; large fragments capture a greater extent of the chemical interactions, resulting in a more accurate initial guess and requiring fewer SCF cycles.

For double-ζ bases, target fragment sizes with the lowest average wall-times tend to veer on the larger size; the optimal target fragment sizes across HF, B3LYP and MN15 are typically between 40 and 60 atoms. The full list of optimal target fragment sizes is detailed in Appendix B, Table B.3. As shown in Figure 4.6, only HF consistently exhibited improved wall-times relative SAD for double-ζ bases computations; average wall-time speedups with respect to SAD of 5.2%, 11.0% and 10.4% were attained with cc-pVDZ, pc-1 and def2-SVP, respectively.

Conversely, despite improvements in wall-times with larger target fragment sizes and lower iteration counts than SAD, B3LYP and MN15 computations with double-ζ bases using MBE1 as initial guess frequently yielded worse average wall-times than SAD spanning all target fragment sizes, as shown in Figures B.15 and B.17. As shown in Figure 4.6, this translates to speedups less than 1 in multiple cases. This stark contrast between HF and B3LYP/MN15 can be ascribed to the computational cost associated with assembling the initial guess. For example in def2-SVP computations employing the optimal target fragment sizes, the initial guess construction step constitutes a much higher proportion in B3LYP (37.1%) and MN15 (38.4%) calculations than in HF (13.7%). Therefore, once more, we find evidence that lower SCF iteration count does not necessarily translate to improved wall-times, and accounting for the computational cost of assembling the initial guess is critical.

In contrast, for triple-ζ basis set MBE computations, improved wall-times could be consistently observed over most target fragment sizes across all levels of theory, as shown in Appendix B Figures B.14, B.16, B.18. Unlike the double-ζ calculations which favored larger target fragment sizes, the optimal target fragment sizes for the triple-ζ bases do not tend towards any particular size; the optimal target fragment sizes varies depending on the theory and basis set employed. For instance, the optimal target fragment size for MN15 is 10 atoms for all triple-ζ bases, whereas for HF computations the optimal target fragment size was 40, 60 and 30 atoms for cc-pVTZ, def2-TZVP, pc-2, respectively.

Employing the corresponding optimal target fragment size, as indicated in Figure 4.6, improved speedups could be observed for many of the triple-ζ calculation using MBE1 as initial guess. Most noticeably, at the B3LYP level, average speedups of 7.9%, 8.3% and 4.7% were attained with cc-pVTZ, pc-2 and def2-TZVP, respectively.

To reiterate, Figure 4.6 demonstrates the average speedups for varying basis sets and theory levels when optimal fragment sizes are used. The fact that slowdowns are observed in these 'best' case scenarios whilst improved iteration counts are observed, emphasises the importance of considering the computational overhead of constructing

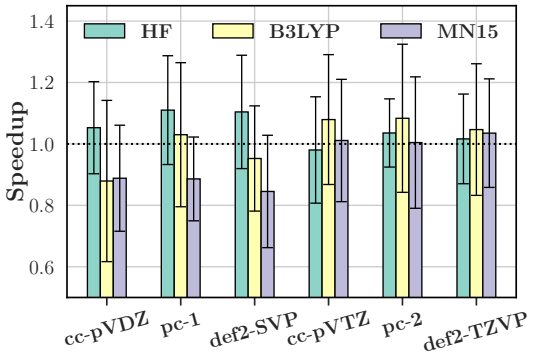


Figure 4.6: Average wall-time speedups of MBE calculations for varying basis sets and theory levels. Optimal target fragment size employed (see Appendix B for full list). Error bars correspond to one standard deviation.

the initial guess. This is particularly important when comparing to previous studies on fragmentation-based initial guesses for SCF acceleration.[192,207,209] Such studies consistently display improved wall-times, accuracy and iteration counts. Thus, such results indicate that whilst acceleration of the final SCF calculation can be achieved (lower SCF cycle count), this can be completely offset by the computational cost of generating the initial guess.

In the following section which analyses the influence of convergence thresholds on the MBE calculations, the optimal target fragment sizes reported above for each respective double/triple-ζ basis set are employed . All MBE computations thus far have employed a convergence threshold of 10^{-6} a.u. for the fragment SCF calculations; as with BSP, we seek to improve the speedups by varying the convergence thresholds of the fragment SCF computations.

Convergence Threshold

The average number of SCF iterations and wall-times with varying fragment convergence thresholds of MBE calculations are shown in Appendix B Figures B.20-B.25. The trends in the SCF cycle count and wall-times seen here in MBE calculations with varying fragment convergence thresholds are similar to those previously observed and explained in Section 4.4.1 for BSP. Due to the improvement in speedups relative to SAD attained with optimal thresholds compared to 10^{-6} a.u. being marginal, these are not reported here and the discussion can be found in Appendix B.

We also explored using an RI-HF approximation to accelerate MBE-based computations further, however, these yielded minimal speedups and are not reported here. The full discussion on employing RI-HF can be found in Appendix B. Instead, in the next Section, the influence of varying fragmentation levels is explored.

Fragmentation Level

In this section, the performance of increasing the fragmentation level to MBE2 is analysed. Specifically, begin with MBE2 computations without dimer cutoffs and then explore the impact of imposing dimer cutoffs. Optimal parameters (fragment size and fragment convergence thresholds) were employed in all MBE2 computations and these are listed in Appendix B.

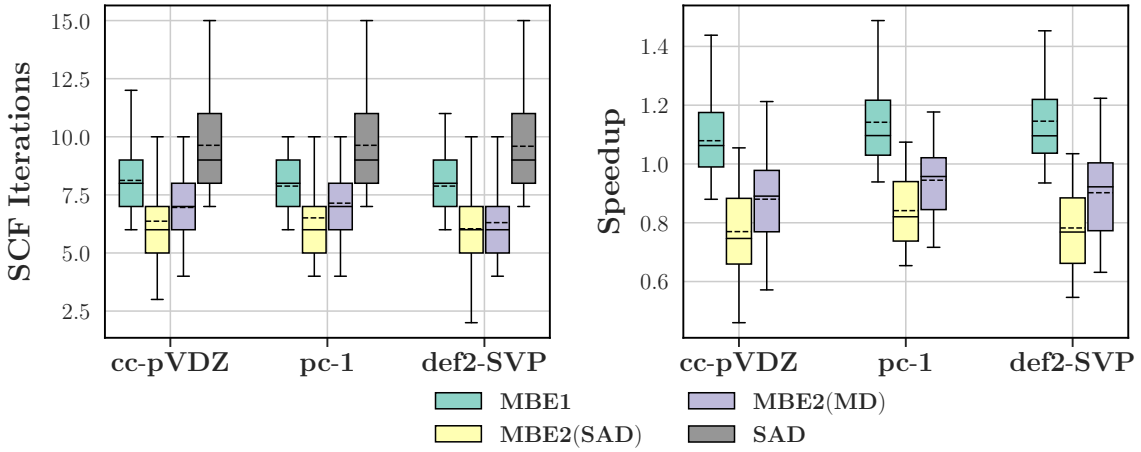


Figure 4.7: Distribution of SCF iterations and wall-time speedups of HF/MBE calculations using double-ζ basis sets at various fragmentation levels. No cutoffs were used for MBE2. Error bars correspond to one standard deviation.

Figure 4.7 and Appendix B Figures B.29-B.33 show distribution of the SCF iteration count and wall-time speedups for MBE calculations at various fragmentation levels across the different theory levels and basis sets.

In many cases, the average number of SCF iterations required in MBE2 computations is less than that of SAD. This is especially true for HF where the reduction in number of iterations is at least 14.1%. Furthermore, all HF/MBE2 computations exhibited fewer iterations than that of MBE1 across all basis sets. MBE2 calculations on average required at least 10.3% and 4.3% less iterations than MBE1 for the double- and triple-ζ basis sets, respectively.

However, exceptions to this includes the case of MBE2(MD) for MN15 computations with double-ζ basis sets in Appendix B Figure B.32, where the SCF iteration count is at least 2.2% higher than that of SAD on average. If the metric used to measure accuracy is the difference between the converged and initial energies, this outcome cannot be explained by the accuracy of the initial guess. As shown in Appendix B Figure B.35, MEB2(MD) attains initial guesses whose energies are closer to the converged value than that of SAD—on average at least 22.8 E_h closer. The higher SCF iteration count of MBE2(MD) may be ascribed to a combination of the fact that MBE2(MD) uses the converged densities of the substituent monomers and the relatively higher convergence

thresholds of 10^{-1} a.u. employed for fragments. The latter 10^{-1} convergence threshold was found to be optimal for MN15 across multiple basis sets (see Appendix B).

Using higher thresholds for monomer densities enables faster convergence than MBE2(SAD). However, this can reduce accuracy, as the resulting dimer density in MBE2(MD) will more closely resemble a simple sum of the monomer densities (*i.e.*, non-interacting monomers) rather than an accurate result of dimer interactions. In contrast, MBE2(SAD) would provide a density matrix closer to actual dimer interactions. This mismatch can lead to over- or under-representations of specific interactions within the system's density matrix, requiring the DIIS algorithm to spend additional iterations correcting these inaccuracies. Consequently, these effects may also question the suitability of using energy differences as a metric to assess the accuracy of initial guess schemes. In the following Section 4.4.3, further evidence for this is provided.

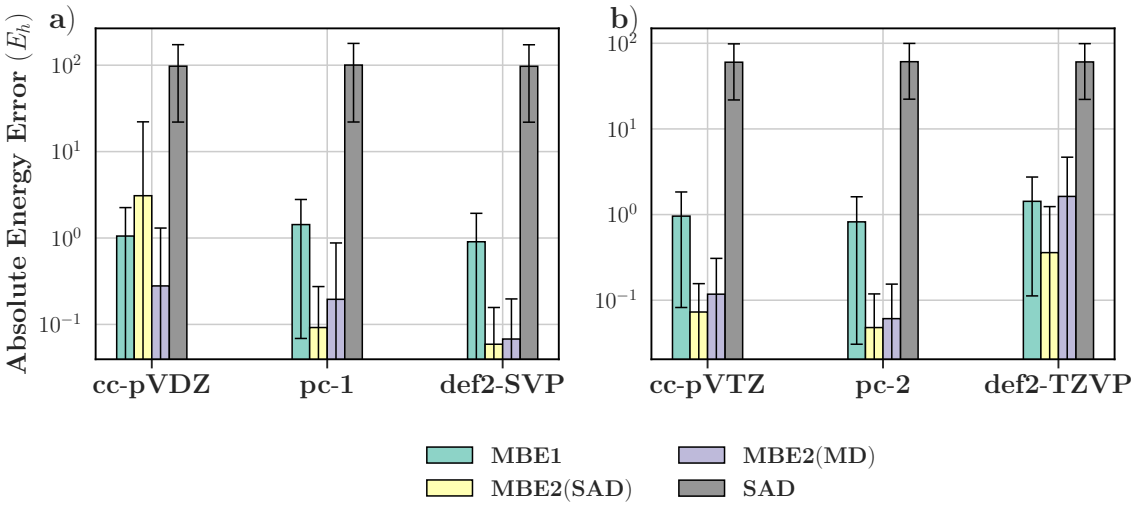


Figure 4.8: Average absolute errors of initial guess energies relative to the converged energies a) double-ζ; and b) triple-ζ MBE computations at various fragmentation levels at the HF level. No cutoffs were used for MBE2. Error bars correspond to one standard deviation.

Regarding energy errors, measured as energy difference between the converged value and that which corresponds to the initial guess, HF/MBE2 calculations routinely exhibit lower errors than that of MBE1 and likely this is responsible for the reduced SCF cycle count observed earlier. Figure 4.8 displays this energy error at the monomer and dimer levels for HF computations. Here, the energy error of both MBE2(SAD) and MBE2(MD) are frequently an order of magnitude less than that of MBE1. For instance in the cc-pVTZ computations, mean energy errors of 0.96, 0.07 and 0.12 E_h were observed for MBE1, MBE2(SAD) and MBE2(MD), respectively. These lowered energy errors attained with higher fragmentation levels is consistent with the existing literature on hierarchical methods: MBE2 captures a greater degree of the interactions present in the original unfragmented system due to the explicit inclusion of dimer in-

teractions.[189–192]

On the other hand, concerning B3LYP and MN15, energy errors are typically of similar orders of magnitude across the fragmentation levels, as shown in Appendix B Figures B.34 and B.35. Additionally, in MN15/double-ζ computations, energy errors of MBE2(MD) are frequently higher than MBE1 and MBE2(SAD) by an order of magnitude. As mentioned earlier concerning the higher SCF cycle count of MN15/doubleζ computations, this is a consequence of the utilisation of monomer densities combined with higher convergence thresholds (10^{-1} a.u.). A similar result is seen for B3LYP/pc-1 in Appendix B Figure B.30, where a convergence threshold of 10^{-1} a.u. is also used for the fragments.

Further, comparing the MBE2(SAD) with MBE1 results of B3LYP and MN15, as displayed in Appendix B Figures B.34 and B.35 energy errors are typically of the same order of magnitude, and even in some cases the errors of MBE2(SAD) exceed that of MBE1. For instance, the MBE2(SAD) calculations of MN15/cc-pVTZ exhibits higher energy errors than MBE1 by 5.53 E_h on average. These outcomes indicate that for hybrid DFT methods such as B3LYP and MN15, inclusion of higher order interactions by increasing fragmentation levels do not necessarily yield more accurate initial guesses as was observed in HF. These results are also consistent with a recent study that demonstrated the divergent behaviour of hybrid DFT methods (containing approximately less than 50% exchange) with fragmentation level (B3LYP and MN15 comprise a 20% and 44% fraction of exact exchange, respectively) due to the self-interacting error of DFT methods.[240]

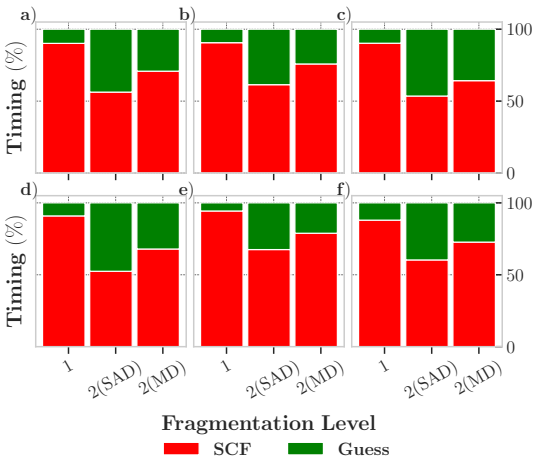


Figure 4.9: Average timing breakdown of MBE calculations at various fragmentation levels for a) cc-pVDZ; b) pc-1; c) def2-SVP; d) cc-pVTZ; e) pc-2; and f) def2-TZVP basis sets at the HF level. No dimer cutoffs were used for MBE2.

The significant discrepancies in the wall-time speedups between MBE1 and MBE2 calculations are predominantly attributed to the significantly higher computational cost of assembling the initial guess with higher levels of fragmentation. Figure 4.9 and Fig-

ures B.36 and B.37 in Appendix B display the variation in the timing distribution with fragmentation levels using the HF, B3LYP and MN15 methods, respectively. Of the three MBE schemes featured, MBE2(SAD) routinely displays the largest time compositions for the guess construction step across all basis sets. A prominent example of this involves the HF/pc-2 calculations where on average the initial guess assembly step constitutes 32.5%, 21.1% and 5.6% of the wall-time for MBE2(SAD), MBE2(MD) and MBE1, respectively. Though replacing MBE2(SAD) with MBE2(MD) reduces the timing contribution of the guess construction step, such improvements remain inferior to the low overhead of the initial guess step in MBE1. Similar outcomes are also attained with B3LYP and MN15 methods.

Due to the substantial variation in guess timing between MBE1 and MBE2 computations being attributed to the presence of dimers, the use of different dimer cutoffs were explored, namely, 5 and 10 Å, to observe whether this could improve the performance of MBE2. However, MBE1 still outperforms MBE2 even with the smallest cutoff of 5 Å. Thus, the findings here are not reported in the main section of this Chapter. The full discussion can be found in Appendix B.

It is important to note that all MBE calculations presented here involved sequential SCF computations of fragments. Although the initial guess construction time for MBE can be reduced by using more parallel processors, in the current implementation, all SCF calculations were performed with the same computational resources. As a result, both the total wall-times and the timing compositions of the guess step for MBE2 are significantly longer than those for MBE1, regardless of the guess method used for dimers or the imposition of dimer cutoffs. Therefore, for this study, MBE1 is the most favorable fragmentation level, andis used in the following section to compare the performance of multiple initial guess approaches.

4.4.3 Comparison Between Initial Guess Methods

In this section, the performance of SCF calculations using multiple initial guess schemes are compared: SAD, BSP, MBE and BSPF. All MBE and BSP results in this Section employ the optimal parameters for fragment size, bootstrapping basis set, convergence threshold, and fragmentation level, as presented earlier in Sections 4.4.1 and 4.4.2. All BSPF computations are applied with the optimal parameters of BSP and MBE.

In Appendix B, figures B.45-B.50 illustrate how the average number of SCF iterations and wall-time speedups vary with system size for different initial guess schemes among the different theory levels and basis sets.

As a general trend, the SCF iteration count grows with system size with the exception of B3LYP and MN15/double-ζ bases calculations where the iteration count remains relatively steady. Of those which exhibit this positive correlation between SCF cycle

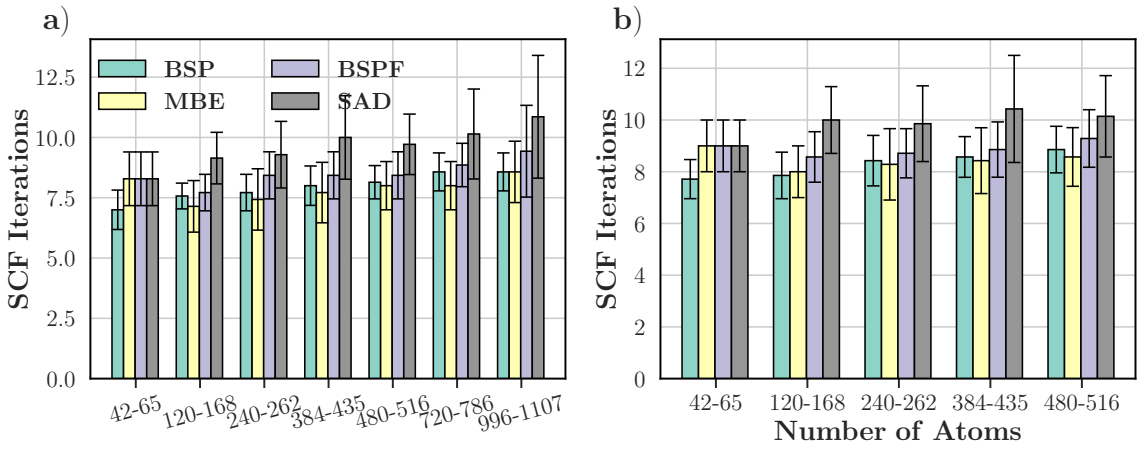


Figure 4.10: Average number of SCF iterations for increasingly large systems employing the a) pc-1 and b) pc-2 basis sets at the HF level. Error bars correspond to one standard deviation.

count with system size, the largest rate of increase is often observed with SAD. For example in HF computations as shown in Figure 4.10, from the smallest to the largest system size, the number of iterations rose by at least 12.7% and 31.0% in SAD computations employing double- and triple-ζ basis sets, respectively. This is a direct result of growing energy errors accompanying larger systems as shown in Appendix B Figure B.51. As system size scales up from the smallest to the largest class, all four initial guess schemes typically experience an increase in the energy error by an order of magnitude. This is most prominently demonstrated in Figure 4.11a) where the energy error rose by 85.45, 7.64, 2.27, and 4.77 E_h in the SAD, BSP, MBE, and BSPF approaches, respectively, in HF/def2-TZVP computations as system sizes increased from 42 to 516 atoms. Similar outcomes can also be observed for the B3LYP and MN15 computations with triple-ζ basis sets.

Within the double-ζ basis calculations, among the three non-SAD approaches, MBE routinely exhibits the highest wall-time speedups. This is most evident in the HF, B3LYP and MN15 calculations employing the pc-1 basis set in Appendix B Figures B.45, B.47 and B.49, where average speedups up to 21.9%, 2.4% and 3.4% were observed, respectively, in the largest class of system sizes (996 to 1,107 atoms). However, it is worth noting that across most systems for B3LYP and MN15, wall-time speedups relative to SAD are rarely observed in the non-SAD initial guess schemes. As discussed earlier in Section 4.4.2, this is due to the computational overhead associated with the initial guess construction step.

Of the three non-SAD approaches, BSPF regularly requires the highest average count of SCF cycles across all theory levels. However, BSPF does not necessarily exhibit the lowest wall-time speedups. For example, in the HF/def2-SVP calculations, BSPF frequently exhibits higher speedups than BSP for systems including more than 65 atoms,

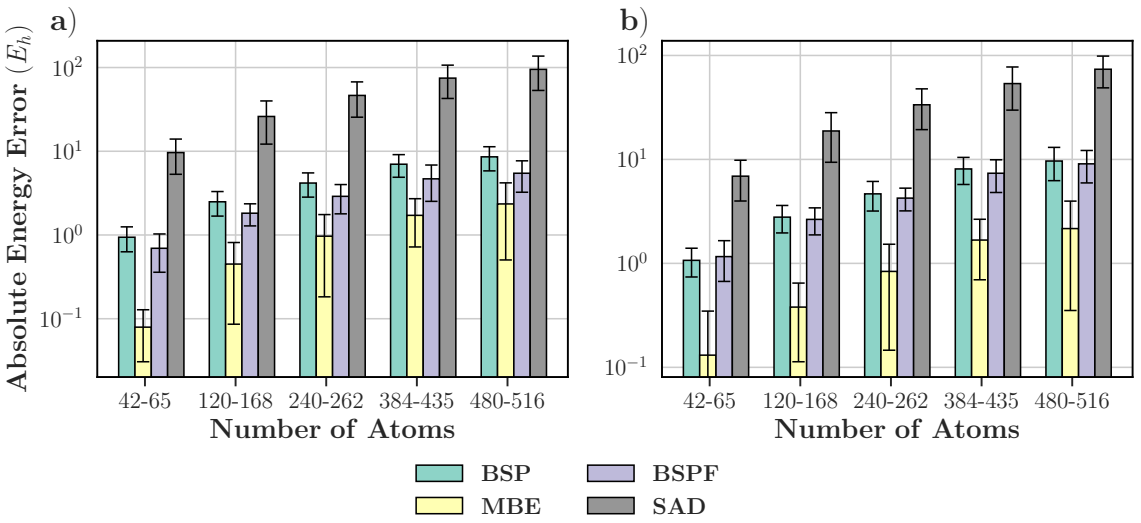


Figure 4.11: Average absolute errors of initial guess energies relative to the converged energies of various initial guess methods for increasingly large systems employing def2-TZVP with the a) HF and b) MN15 methods. Error bars correspond to one standard deviation.

as shown in Appendix B Figure B.45. Similarly, as shown in Appendix B Figure B.48, BSPF displays higher wall-time speedups than MBE in B3LYP/triple-ζ calculations for systems ranging between 42 and 262 atoms. This mismatch between relatively high SCF iteration count with higher wall-time speedups is once again a consequence of the time fraction spent constructing the initial guess of BSPF being consistently less than that of BSP in HF/double-ζ and MBE in B3LYP/triple-ζ (see Fig. 4.12 and Fig. B.56 in Appendix B). For HF, this is most evidently seen in the case of the largest class of systems (between 996 and 1,107 atoms) where the guess time fraction of BSP (8.2% for def2-SVP) is an order of magnitude greater than that of BSPF (0.5% for def2-SVP). On the other hand for B3LYP/def2-TZVP, the initial guess assembly step constituted 17.1% and 32.1% of the wall-time in BSPF and MBE calculations, respectively, for systems up to 144 atoms.

Within the triple-ζ basis set computations spanning all theory levels, it can be observed that among the three non-SAD initial guess methods as featured in Appendix B Figures B.46, B.48 and B.50, BSP frequently requires the fewest SCF cycles and achieves the highest average wall-time speedups. However, when comparing this to the corresponding energy errors shown in Appendix B Figures B.51, B.52 and B.53d) to f), BSP consistently displays the highest average energy errors. Despite this, the energy errors across all three non-SAD methods are typically of the same order of magnitude. For example, as shown in Figure 4.11b), the mean energy errors in MN15/def2-TZVP computations between BSP and MBE varied from 0.93 to 7.49 E_h, while deviations between BSP and BSPF ranged from -0.09 to 0.73 E_h. Although BSP shows significantly larger energy errors than MBE and BSPF, it performs best in both SCF cycles and wall-time

speedups.

These results indicate that the proximity of the first iteration energy, *i.e.*, the energy corresponding to the initial density guess, to the final SCF energy does not necessarily lead to fewer SCF iterations for convergence. When different initial guess methods yield energy errors within the same order of magnitude at the first iteration, there is unlikely to be a correlation between these energy errors and the number of iterations required for convergence. Notably, the energy errors of SAD are typically at least an order of magnitude larger than those of MBE, BSP, and BSPF, suggesting that an improvement of at least one order of magnitude in energy errors is associated with noticeable performance enhancements.

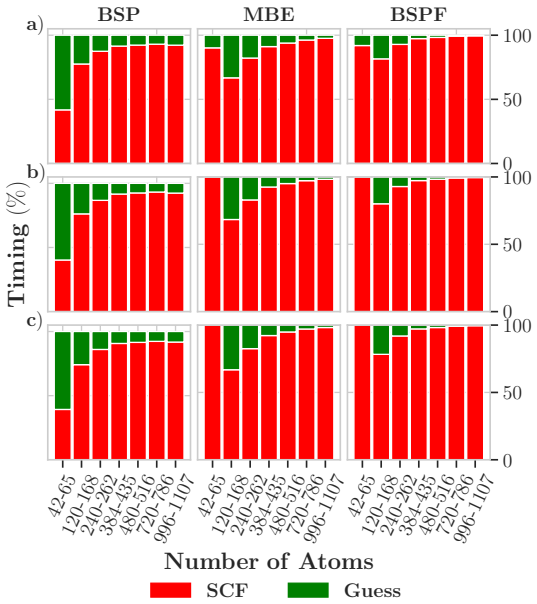


Figure 4.12: Average timing breakdown of the BSP, MBE, BSPF initial guess approaches with varying system sizes employing the a) cc-pVDZ; b) pc-1; and c) def2-SVP double-ζ basis sets at the HF level.

Beyond energy errors, it is also valuable to consider the computational cost breakdown between the SCF procedure and the initial guess construction step as system size increases. The timing breakdown of BSP, MBE, and BSPF with varying system sizes with the different theory levels and basis sets are shown in Fig. 4.12 and Figures B.54-B.58 in Appendix B, respectively. Larger systems generally show a lower timing contribution from the initial guess construction step across all three non-SAD initial guess approaches. For instance, in HF/def2-SVP calculations, as system size increased from 120 to 1,107 atoms, the BSP and BSPF methods experienced decreases in guess timing by 17.6% and 21.3%, respectively. A similar trend is likewise observed with B3LYP and MN15 methods.

These outcomes suggest that employing non-SAD initial guesses becomes increasingly worthwhile for larger systems.

4.4.4 Application to Difficult-to-Converge Systems

Metalloenzymes

In this subsection, the initial guess schemes are applied to molecular systems where achieving SCF convergence is particularly challenging, such as those containing metals, where convergence is often difficult to guarantee. Specifically, the optimal parameters of BSP and MBE as reported in Sections 4.4.1 and 4.4.2 are applied to a set of metalloenzyme systems to examine their impact on SCF convergence.

Metalloenzymes play a crucial role in a wide range of biological, geochemical, and industrial processes, with metal ions being essential for their catalytic activity. To explore the effectiveness of the initial guess approaches, 20 metalloenzyme structures were selected. These structures contained metals such as zinc, calcium, sodium, and magnesium, with sizes ranging from 334 to 1,084 atoms

This section is constrained to HF only as application of the four initial guess schemes (SAD, BSP, MBE, BSPF) to DFT calculations on metalloenzymes yielded results with no discernible difference; many systems failed to converge, and this likely exacerbated by the relatively large system sizes and the lowered HOMO-LUMO gaps associated with metals.

The 20-metalloenzymes data set consists of a diverse range of protein sequences. Specifically, 90.1% and 5.1% of pairs of these structures have pairwise sequence identity (PID) scores below 20% and between 20% and 30%, respectively. Only 4.7% of pairs exhibited PID scores greater than 30%.

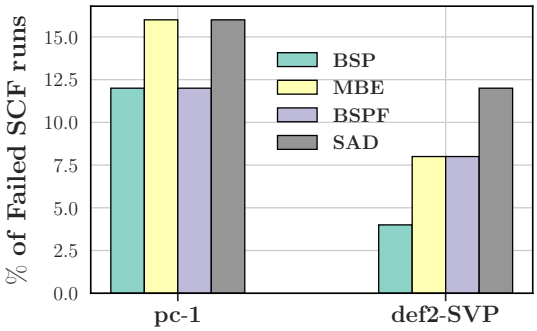


Figure 4.13: Percentage of HF calculations that failed to reach SCF convergence in the metalloenzyme dataset.

Figure 4.13 shows the percentage of systems that failed to converge with the varying initial guess schemes for the double-ζ basis sets. For pc-1, MBE exhibits the same failure rate as SAD (16.0%), whereas this is lower for both BSP and BSPF (12.0%). For def2-SVP, SAD has the highest convergence failure rate (12.0%) and this decreases with the adoption of MBE (8.0%), BSPF (8.0%) and BSP (4.0%).

For both pc-1 and def2-SVP, SAD consistently exhibits the highest average SCF it-

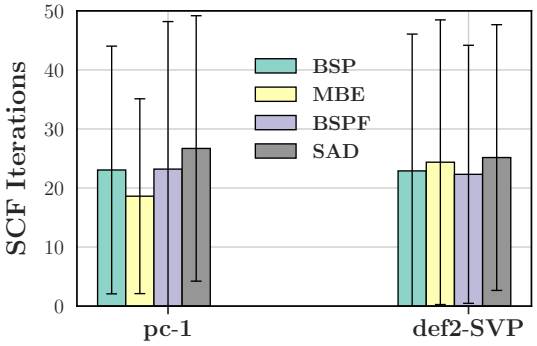


Figure 4.14: Average number of SCF iterations of double-ζ basis set calculations for the metalloenzyme systems.

eration count, as shown in Fig. 4.14. The average number of iterations can decrease by as much as 30.3% with MBE and 11.3% with BSPF for the pc-1 and def2-SVP basis sets, respectively. This is also reflected in the wall-time speedups shown in Fig. 4.15, where all three non-SAD methods exhibit lower mean wall-times than SAD. Furthermore, the initial guess approaches with the highest wall-time speedups correspond to those which exhibited the fewest SCF iteration count, which are MBE for pc-1 (16.4%) and BSPF for def2-SVP (13.8%).

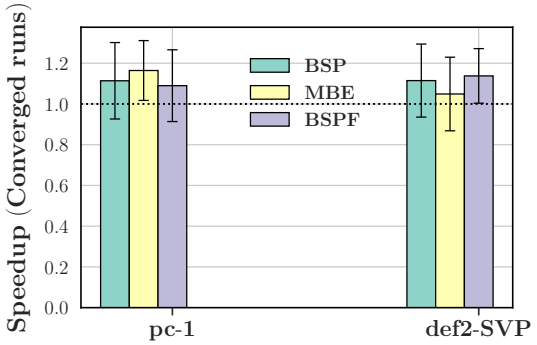


Figure 4.15: Average wall-time speedups of double-ζ basis set calculations for the metalloenzyme systems.

On the other hand, for the triple-ζ basis set where the dataset comprises 10 metalloenzymes, all 10 systems converged across all four initial guess schemes for def2-TZVP.

Scheme	SCF Iterations	Speedup
SAD	19.2	-
BSP	14.5	34.9%
MBE	20.2	5.0%
BSPF	16.4	18.7%

Table 4.1: Average SCF iteration count and wall-time speedups of various initial guess schemes for def2-TZVP. Speedups are relative to SAD.

Table 4.1 lists the average SCF iteration count and wall-time speedups of the various

initial guess approaches obtained for def2-TZVP calculations. In particular, BSP and BSPF exhibit 24.5% and 14.6% less iterations than SAD on average. Conversely, MBE has a higher average iteration count by a marginal 5.2% compared to SAD, although MBE still results in a 5.0% wall-time speedup relative to SAD. This is mostly a result of the iteration count being skewed by two systems, both comprising zinc, requiring a high number of iterations for convergence (more than 30 iterations). When these two systems are excluded, the average iteration count is 14.0, 9.6, 12.3 and 11.5 for SAD, BSP, MBE and BSPF, respectively.

All non-SAD schemes displayed lower average wall-times than SAD. In particular, among the non-SAD initial guesses for def2-TZVP, BSP exhibits the best performance in terms of both speedup (34.9%) and SCF iteration count.

These results highlight the value of non-SAD initial guess schemes for HF. Not only do they reduce the number of SCF iterations and improve wall-time efficiency, but they also enable convergence in HF calculations on challenging systems, such as metalloenzymes, where a SAD initial guess would often lead to converge failure.

Open-Shell States

In addition to metal-based systems, another group of systems of interest where SCF convergence is especially challenging are those with open-shell states. Due to their unpaired electron(s), open-shell systems and states exhibit unique properties distinct from closed-shell systems such as paramagnetism, enhanced reactivity and photochemical sensitivity, which can be exploited across many fields including reaction chemistry, biology, materials science. Due to their importance and convergence difficulties in SCF, the potential of non-SAD initial guesses on enhancing the convergence of triplet states was explored, using the original benchmark set of proteins, nucleic acids, carbohydrates, water clusters, benzene clusters, and ionic liquids.

The comparison is limited to basis set projection with SAD due to the ambiguity of assigning multiplicities to fragments. Optimal bootstrapping basis detailed in Section 4.4.1 sets were employed, and a constant convergence threshold of 10^{-4} a.u. was used for the SCF calculations in the bootstrapping basis set as this is a constraint within Q-Chem.

Figure 4.16 displays the percentage of systems that failed to converge with SAD and BSP initial guess schemes schemes in HF and B3LYP calculations on triplet states. In HF, only pc-1 calculations showed any improvement in convergence rates when using BSP, though this improvement is marginal (2.3%). BSP consistently exhibits higher rates of convergence failures in both HF and B3LYP methods. In the worst case (HF/cc-pVTZ), a difference in the failure rate of 67.0% was observed. The source of failure invariably occurred in SCF computations in the projected basis set; all SCF computations in the bootstrapping basis sets converged successfully. The occurrence of failed SCF calcula-

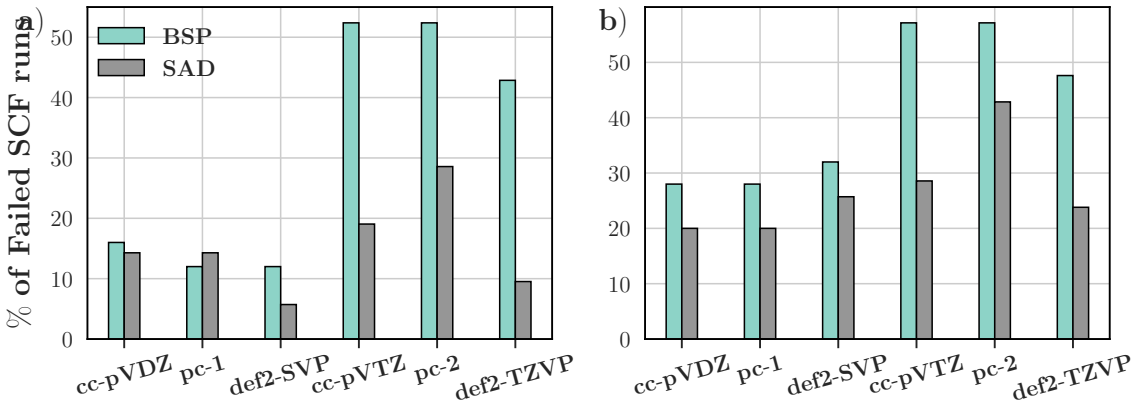


Figure 4.16: Percentage of a) HF and b) B3LYP calculations that failed to reach SCF convergence of triplet states using BSP and SAD initial guess schemes.

tions is likely attributed to the multiple close-lying electronic states that is often characteristic of open-shell systems and this is exacerbated when using larger basis sets such as the projected basis sets. Thus, such results indicate that BSP does not necessarily improve the rate at which successful convergence can be achieved for open-shell systems, and specifically for triplet states.

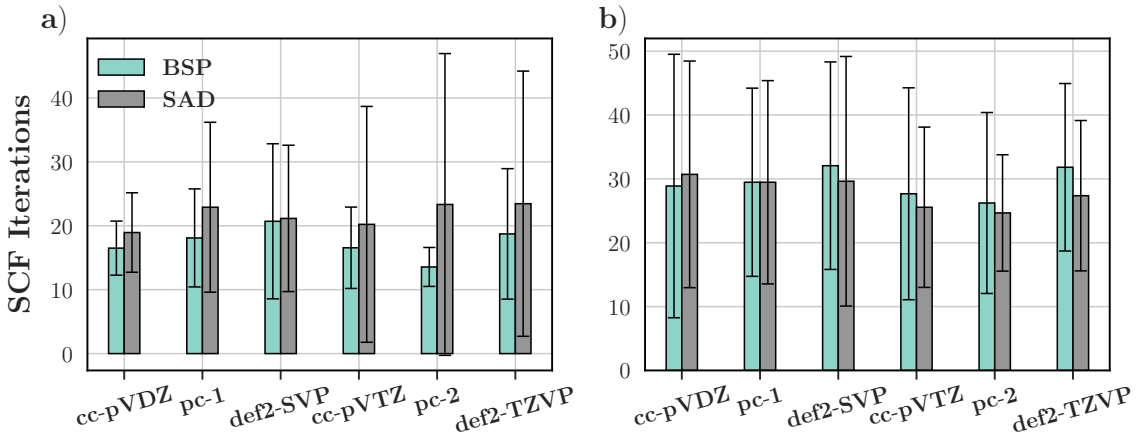


Figure 4.17: Average number of SCF iterations of a) HF and b) B3LYP open-shell triplet calculations employing BSP and SAD initial guess schemes.

Considering the set of converged systems only, Figure 4.17 illustrates the SCF cycle count for the BSP and SAD schemes of HF and B3LYP methods. In HF computations across all basis sets, BSP continually requires lower iteration counts. This is especially prominent in pc-2 computations where the mean SCF iteration count of BSP is 41.9% less than that of SAD. These lower counts of SCF cycles of BSP in HF are also reflected in the higher SAD to BSP wall-time ratios as listed in Table 4.2. For computations involving double- and triple-ζ basis sets, average speedups of up to 20.3% and 80.3% could be observed. This large speedup value of 80.3% is attributed to one carbohydrate system which converged in 11 and 86 iterations using the BSP and SAD schemes, respectively.

	HF	B3LYP
cc-pVDZ	1.13	1.29
pc-1	1.20	1.16
def2-SVP	1.02	1.17
cc-pVTZ	1.03	0.97
pc-2	1.80	1.11
def2-TZVP	1.48	0.92

Table 4.2: Average ratio of SAD to BSP wall-times of open-shell triplet calculations for varying basis sets and theories.

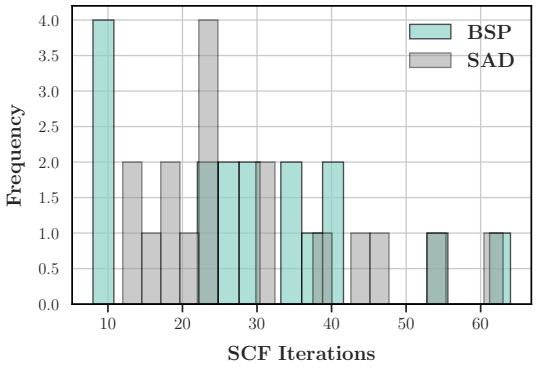


Figure 4.18: Distribution of SCF iteration count of B3LYP/pc-2 computations employing the BSP and SAD initial guess schemes for triplet states.

Removal of this reduces the average speedup observed to 8.3%.

In B3LYP calculations, BSP routinely requires a greater number of SCF iterations than SAD, especially in def2-TZVP calculations where the mean iteration count of BSP is 16.3% higher than SAD. Unsurprisingly, these higher SCF iteration counts observed are reflected with higher average wall-times relative to SAD as tabulated in Table 4.2. In particular, the mean wall-time ratios of SAD to BSP for B3LYP/cc-pVTZ and B3LYP/def2-TZVP are 0.97 and 0.92, respectively, demonstrating that BSP does not yield any significant speedup relative to SAD for B3LYP for these triple-ζ bases.

On the other hand, B3LYP/pc-2 exhibits an average speedup of 11.3% despite BSP showing a higher SCF count than SAD as displayed in Figure 4.17. This is predominantly due to the fact that SCF cycle counts are not normalised whereas speedups are. Considering the distribution of SCF iterations as displayed in Figure 4.18, the reason for the 11.3% speedup is made apparent. BSP attains a higher occurrence of lower iteration counts than SAD; iteration counts of $\sim$10 are observed with BSP but not SAD. This explanation can also be extended to that of B3LYP/def2-SVP where BSP exhibits a mean speedup of 17.2% despite higher SCF iteration counts relative to SAD.

In addition to differences in wall-times and SCF iteration counts, the energy values in which BSP and SAD converge to can vary significantly for HF calculations. Figure 4.19 captures the distribution of the converged energy differences between SAD and

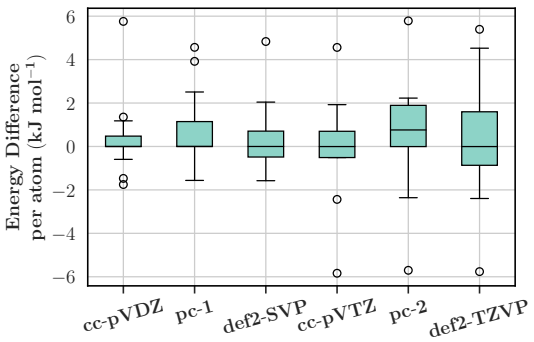


Figure 4.19: Distribution of the differences in the converged energy values between SAD and BSP normalised for system size using the HF method. Energy difference = $E_{SAD} - E_{BSP}$.

BSP normalised for system size by dividing by the number of atoms. Normalised energy deviations ranging between -5.8 and 5.8 kJ mol^{-1} atom^{-1} are observed across the basis sets although the vast majority of these are aggregated around zero. Therefore, whilst BSP can reduce the wall-time of SCF calculations relative to SAD, this also comes with the caveat that the resulting converged energy value may be distinct to that of SAD and the magnitude of this difference can grow with system size. For B3LYP, energy differences are significantly smaller and mostly negligible, with the highest deviation of 0.6 kJ mol^{-1} atom^{-1} attained.

Taken together, the outcomes of this section demonstrate that BSP does not decrease the failure rate of convergence of triplet states, in fact, the opposite is observed. However, for those that do converge with BSP, improved wall-time speedups can be routinely observed relative to SAD, especially for HF, but this is not always the case for B3LYP. Furthermore, different initial guess schemes can result in convergence to different triplet states, particularly in HF, and rarely with B3LYP.

4.5 Conclusion

In this Chapter, a detailed analysis of the performance of basis set projection and many body expansion method is performed to improve the density initial guess in SCF calculations across a diverse set of molecular systems. The performance of these methods is assessed by consideration of time-to-solution (wall-times), the number of SCF iterations and energy errors.

The influence of multiple parameters on BSP (bootstrapping basis set and convergence threshold) and MBE (fragment size, convergence threshold, RI-HF approximation, and fragmentation level) was examined. The results indicate that the number of SCF iterations and the wall-times for BSP calculations are sensitive to the choice of the bootstrapping basis set. Most BSP calculations showed improvements in SCF cycles and

wall-times compared to the standard initial guess (SAD) for HF. However, adoption of BSP routinely lead to higher overall wall-times for B3LYP and MN15 in double-ζ basis set computations. Reduction in wall-times with B3LYP/MN15 were only observed when employing triple-ζ basis sets.

Additionally, raising the convergence threshold from 10^{-6} up to 10^{1} a.u. led reductions in SCF cycles and timings for both BSP and MBE methods in HF, B3LYP and MN15, though these improvements were marginal.

For MBE, optimal target fragment sizes generally depended on theory level and basis sets used; double-ζ tended to favor larger fragments whereas smaller fragments were typically optimal for triple-ζ bases. Marginal speedups were also observed by employing higher convergence thresholds (up to 10^{-1} a.u.) in fragment SCF calculations. Similar to BSP, MBE computations with B3LYP and MN15 frequently exhibited higher wall-times than SAD when employing double-ζ bases. Adopting the RI-HF method for fragment calculations could accelerate these MBE-based calculations further in HF, although only minimally. When increasing the fragmentation level, MBE2 consistently resulted in decreased number of iterations and energy errors in HF. However, due to the self-interaction error of DFT methods, the opposite was seen (higher SCF cycle counts and energy errors) in B3LYP and MN15 MBE2 computations. Nevertheless, across all HF and DFT methods, MBE1 consistently exhibited superior performance over MBE2.

With optimal BSP and MBE parameters, the performance of SAD is compared with three non-SAD initial guess methods: BSP, MBE, and BSPF—a hybrid of the two. Across all basis sets and theories, these non-SAD methods typically required fewer SCF iterations and showed improved wall-time speedups compared to SAD.

Among the non-SAD approaches, BSP displayed the highest wall-time speedups for triple-ζ basis sets, while MBE predominantly showed the greatest speedups for double-ζ basis sets.

The correlation between SCF cycle count and energy errors of the initial guess were also explored. BSP, MBE, and BSPF typically exhibited energy errors an order of magnitude lower than SAD, accompanied by fewer SCF iterations and greater wall-time speedups. However, improvements in energy errors do not always correlate with performance improvements; when energy errors are of the same order of magnitude, slight accuracy gains do not necessarily result in fewer SCF iterations or shorter wall-times.

Furthermore, the application of the various initial guess schemes on a set of difficult-to-converge metalloenzyme and open-shell (triplet) states was examined. For metalloenzyme systems, it was demonstrated that use of these non-SAD initial guess schemes can achieve convergence in cases where SAD would lead to convergence failures. However for triplet states, the contrary was observed where the convergence failure rate of BSP was higher than that of SAD in both HF and B3LYP computations. Despite this, mean speedups of up to 80.3% and 29.3% could respectively be attained with HF and

B3LYP methods in cases where convergence was successful. Further, in HF calculations, utilisation of differing initial guess schemes could lead to the convergence to different triplet states.

Thus, the following recommendations are made: 1) For closed-shell systems, MBE1 and BSP for double- and triple-ζ basis set calculations, respectively, are suggested as these yield the greatest wall-time speedups. However, the suggestion for double-ζ basis sets applies to HF only. 2) For HF calculations of metal-based systems, any of the three non-SAD initial guess schemes can decrease the occurrence of convergence failures and enhance wall-time speedups. 3) For triplet states, we do not advise the use of BSP over SAD to reduce convergence failures in both HF and DFT computations. Although wall-time speedups can be attained, users should be careful when adopting BSP since convergence is not always successful and may lead to convergence to a distinct triplet state from that of SAD, particularly with HF.

The outcomes of this study highlight the importance of considering the computational cost of assembling the initial guess when assessing the efficacy of an initial guess approach. The relative computational cost of non-SAD initial guess methods reduces with increasing system sizes, indicating that more sophisticated non-SAD initial guess schemes become progressively worthwhile for larger systems.

Overall, the results of this study provide a comprehensive comparison of the performance between multiple initial guess methods with varying parameters, offering a useful reference guide towards the acceleration of SCF calculations through improved initial guesses.

Although this thesis largely focuses on protein systems, and indeed the findings of this Chapter have highlighted the improved speedups attained with non-SAD approaches on such structures, the performance of these initial guess techniques have been analysed across a diverse set of systems including water, benzene, ionic liquids, nucleic acids and carbohydrates. Such results emphasise the wide applicability of these initial guess schemes on chemical systems in general, not only those of protein systems.

High-Performance GPU-accelerated Solution of the Linear Poisson-Boltzmann Equation

5.1 Introduction

Realising computationally feasible Quantum Mechanical / Poisson-Boltzmann Surface Area (QM/PBSA) workflows for large scale virtual screening involves at least two components, namely, fast computation of both the gas phase (QM) energies and solvation energies (*via* PBSA). Both Chapters 3 and 4 explored techniques towards realising the application of QM methods on large molecular systems such as proteins for its accurate prediction of gas phase energies. Besides QM, the polar component of the solvation energy poses a significant bottleneck in the QM/PBSA workflow due to its computational demand and the lack of fast Poisson-Boltzmann solvers that leverage the parallelism afforded by modern GPUs. To address this gap, this Chapter addresses Aim 3 and presents a new GPU-accelerated linear PBE solver suitable for large scale PBSA-based workflows.

In end point methods such as QM/PBSA, the solvation energy term ΔG_{solv}

$$\Delta G_{solv} = G^{PL}_{solv} - G^{P}_{solv} - G^{L}_{solv} \tag{5.1}$$

is taken as the difference in Gibbs free energy of solvation between the protein-ligand complex (PL), and that of the protein (P) and of the ligand (L) in isolation. Thus, the evaluation of ΔG_{solv} for a single protein-ligand complex involves at least two expensive PBSA computations, namely, on the P and PL structures, since these comprise the protein structure typically comprising thousands of atoms.

Several linearised PBE solvers have been developed for utilisation in PBSA workflows. Popular PBE solver software includes DelPhi,[35–39] Adaptive Poisson-Boltzmann Solver (APBS),[44] and AMBER's *pbsa* module.[241] However, these implementations predominantly leverage CPUs and thus fail to exploit the massive parallelism afforded by

modern Graphics Processing Unit (GPU)-based computing architectures.

Given that end point methods such as QM/PBSA (or MM/PBSA) involve hundreds to thousands of PBE calculations due to the molecular dynamics simulation step-often sampling protein-ligand conformations every 10 ps across simulations of at least 1 ns—this limitation severely restricts the scope of large scale computational drug discovery efforts.[242–246]

Recognising the need for high-performance GPU-accelerated PBE solvers, several GPU implementations have been proposed. These include AMBER's CUDA-enabled *pbsa.cuda* module,[12,40] a GPU-accelerated direct-sum boundary integral method employing a generalised minimum residual (GMRES) solver,[41] and a third-party GPU adaptation of DelPhi.[42,43] Some of these GPU implementations have been reported to demonstrate substantial performance advantages, with DelPhi achieving approximately $10\times$ speedups and AMBER's *pbsa.cuda* up to $50\times$ compared to their CPU-based counterparts.[12,42]

Nonetheless, significant shortcomings remain. The third-party DelPhi GPU adaptation is no longer maintained or compatible with contemporary high-performance computing infrastructure, and the direct-sum boundary integral GPU method is not publicly available as a distributable package. On the other hand, AMBER's *pbsa.cuda*, the most widely used GPU-accelerated PBE solver for molecular systems, exclusively supports single-precision arithmetic and can lead to non-negligible deviations (as large as -20 to 10 kcal mol^{-1}) from double-precision results[12] when paired with the commonly used convergence threshold of 10^{-3}. Though these can be reduced significantly with a tighter threshold of 10^{-6}, the availability of double precision implementations remains important for maintaining and testing accuracy. Furthermore, AMBER's *pbsa.cuda* relies on linear algebra libraries to handle specialised matrix data structures and Krylov solvers. While such libraries provide higher-level interfaces and portability between different architectures, due to their general purpose, they tend to under-perform compared to tailored implementations, especially when it comes to exploiting the specific sparsity patterns exposed by the PBE problem which are detailed in Section 5.2.3.

To address these limitations and provide the community with a robust and accurate GPU-accelerated solution, this work introduces a novel double-precision GPU implementation of the linearised PBE within the Extreme-scale Electronic Structure System (EXESS) quantum chemical software package.[22–25,27–29,34,169,247] The proposed solver's accuracy and performance is rigorously evaluated against the established CPU-based DelPhi solver and the GPU-accelerated AMBER *pbsa.cuda* module, demonstrating superior computational speed and numerical precision on large scale protein-ligand systems containing thousands of atoms.

This Chapter is structured as follows. Section 5.2 provides the numerical framework and describes the proposed GPU-accelerated implementation of the linearised Pois-

son–Boltzmann solver. Section 5.3 details the computational setups, including software parameters and molecular datasets used for benchmarking. Section 5.4 presents comprehensive accuracy and performance analyses comparing the new method presented herein to the DelPhi and *pbsa.cuda* implementations. Finally, Section 5.5 summarises the findings and concludes the study.

5.2 Theory and Algorithms

5.2.1 Poisson-Boltzmann Method

This section details the proposed approach for numerically solving the linear Poisson–Boltzmann equation. An overview of the PBSA method has been provided in Chapter 2 Section 2.5.

In this implementation, a Jacobi preconditioned conjugate gradient (CG) algorithm is used to to solve the set of linear equations, Eq. 2.51 (represented in matrix form as Eq. (2.52)). The Jacobi preconditioned CG solver was selected as it has demonstrated stable convergence at varying thresholds compared to other solvers such as multigrid and relaxation methods.[40]. Furthermore, compared to other approaches—such as incomplete Cholesky preconditioners, smoothed-aggregation-based algebraic multi-grid preconditioners or no preconditioner—the Jacobi preconditioner has been shown to exhibit superior computational efficiency in GPU-based implementations of the CG solver.[12]

In the Jacobi preconditioned approach, a preconditioner matrix is introduced to improve convergence of the CG algorithm, modifying Eq. (2.52) as follows

$$\boldsymbol{M}^{-1}\boldsymbol{A}\mathbf{x} = \boldsymbol{M}^{-1}\mathbf{b} \tag{5.2}$$

where $\boldsymbol{M}$ is the Jacobi preconditioner matrix and is taken as the diagonal of $\boldsymbol{A}$. Therefore, the computation of its inverse ($\boldsymbol{M}^{-1}$), which is required for CG, is trivial and relatively inexpensive. The details of solving Eq. (5.2) are given in the following section.

5.2.2 Conjugate Gradient Algorithm

This section presents an efficient GPU-accelerated algorithm to solve the linear Poisson-Boltzmann equation.

Algorithm 5.1 shows pseudocode for the implementation of the Jacobi preconditioned CG solver. The notation $\|.\|$ denotes the L_2 norm. All variables in bold denote vectors (*e.g.*, $\mathbf{r}$), whilst those which are both italicised and in bold denote matrices (*e.g.*, $\boldsymbol{M}$). All remaining variables are scalar quantities. The N_{iter} parameter represents the

maximum number of iterations and τ is the convergence threshold, which are both user-defined quantities.

Algorithm 5.1 The Jacobi preconditioned conjugate gradient algorithm.

1: $\mathbf{r} \Leftarrow \mathbf{b} - \boldsymbol{A}\mathbf{x}$
2: $\mathbf{p} \Leftarrow \boldsymbol{M}^{-1}\mathbf{r}$
3: $iteration \Leftarrow 0$
4: **while** $iteration < N_{iter}$ or $\|\mathbf{r}\| > \tau$ **do**
5: $\quad \alpha \Leftarrow (\mathbf{r} \cdot \boldsymbol{M}^{-1}\mathbf{r})/(\mathbf{p} \cdot \boldsymbol{A}\mathbf{p})$
6: $\quad \mathbf{x} \Leftarrow \mathbf{x} + \alpha\mathbf{p}$
7: $\quad \mathbf{r}_{next} \Leftarrow \mathbf{r} - \alpha\boldsymbol{A}\mathbf{p}$
8: $\quad \beta \Leftarrow (\mathbf{r}_{next} \cdot \boldsymbol{M}^{-1}\mathbf{r}_{next})/(\mathbf{r} \cdot \boldsymbol{M}^{-1}\mathbf{r})$
9: $\quad \mathbf{p} \Leftarrow \boldsymbol{M}^{-1}\mathbf{r}_{next} + \beta\mathbf{p}$
10: $\quad \mathbf{r} \Leftarrow \mathbf{r}_{next}$
11: $\quad iteration \Leftarrow iteration + 1$
12: **end while**
13: **return** $\mathbf{x}$

Algorithm 5.1 begins by calculating the residual vector $\mathbf{r}$ (line 1) and uses this to calculate the direction vector $\mathbf{p}$ (line 2) which directs the next guess of $\mathbf{x}$ later on line 6. The $iteration$ variable is initialised to zero just before the while loop and serves as the iteration counter. Within the while loop, the first step involves calculating the step size α (line 5) to obtain the new guess of $\mathbf{x}$ (line 6). Next, the residuals are updated (line 7) and the gradient corrector factor β is computed (line 8) to update the new search direction on line 9. Finally, the residual $\mathbf{r}_{next}$ is stored as $\mathbf{r}$ (line 10) to prepare for the residual computation in the next iteration and the counter variable $iteration$ is incremented. The while loop terminates when either $||\mathbf{r}||$ is less than the convergence threshold τ or when N_{iter} is reached.

In the proposed implementation, $\boldsymbol{A}$ is assembled and stored on the GPU to reduce high-latency host-device data transfers. Specific details on the implementation of $\boldsymbol{A}$ are described in Section 5.2.3. Lines 1 and 2 of Algorithm 5.1 are performed on the GPU device. The while loop (lines 4 to 12) control logic is executed on the host CPU. Lines 5 to 10 are all executed on the GPU, with each implemented as comprising one or more GPU kernels.

The proposed GPU implementation of the Jacobi preconditioned CG algorithm which solves the linear PBE for molecular systems differs from previous implementations, namely, AMBER's *pbsa.cuda*[12] in at least two ways. First, the proposed implementation does not utilise linear algebra libraries such as CUSP and Thrust for specialised matrix data structures (diagonal) and Krylov solvers such as the CG routine. This choice to forgo such libraries was intentional since the sparse matrix-vector multiplication kernel presented herein is memory bound. Dense matrix-vector multiplications are memory bound,[248] and therefore sparse matrix-vector multiplications are even more so. Conse-

quently, having a tailored implementation enables us to achieve better memory access patterns and exploit the memory bandwidth of the GPU over libraries which are suited for general purposes. Furthermore, the new PBE solver is implemented in double precision instead of in single precision. Later, in Section 5.4, it is demonstrated that despite the extra computational burden of executing the algorithm in double precision floating point arithmetic, the proposed GPU implementation in EXESS exhibits lower wall times than AMBER's *pbsa.cuda* GPU implementation in single precision.

5.2.3 Sparse Matrix Treatment

In Eq. (2.52), the matrix $\boldsymbol{A}$ exhibits a sparse seven-banded structure where only the main diagonal and the three diagonals above and below are nonzero[148] as displayed in Fig. 5.1. For conciseness, each diagonal i is denoted as D_i.

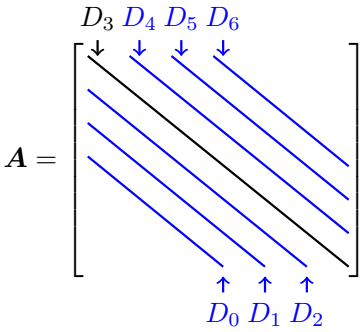


Figure 5.1: Seven-banded sparsity structure of $\boldsymbol{A}$. Main diagonal D_3 is shown in black. Corresponding directions of each diagonal D_i are listed in Table 5.1.

Non-zero entries outside of the main diagonal correspond to dielectric constants between a grid point and one of its adjacent grid points in the positive and negative x, y and z directions. Table 5.1 summarises the direction each diagonal D_i corresponds to.

i	Direction
0	Negative z
1	Negative y
2	Negative x
3	Main diagonal
4	Positive x
5	Positive y
6	Positive z

Table 5.1: Directions represented by each diagonal D_i and the corresponding offsets.

In the proposed implementation, only these seven diagonals of $\boldsymbol{A}$ are stored on the GPU, forgoing storing the full N_{grid}^2 elements of $\boldsymbol{A}$. Each diagonal D_i is stored as an array and the matrix-vector multiplication step is performed on the GPU which exploits

this sparse structure of $\boldsymbol{A}$. This allows the $\boldsymbol{A}\mathbf{p}$ contractions that are prevalent in Algorithm 5.1 to scale as $\mathcal{O}(n)$ as opposed to the $\mathcal{O}(n^2)$ scaling for a typical full matrix-vector multiplication, where n refers to the matrix-side dimension.

Algorithm 5.2 Sparse matrix-vector multiplication routine.

1: **for** $i \Leftarrow$ `Index` $< N_{grid}$ **do**
2: **for** $k = 0$ to 6 **do**
3: $b(i) \Leftarrow b(i) + D_k(i)\, p(i + \delta_k)$
4: **end for**
5: $i \Leftarrow i + N_{threads}$
6: **end for**

Algorithm 5.2 outlines the general structure of the sparse matrix-vector multiplication employed. Specifically, it evaluates the contraction $\mathbf{b} = \boldsymbol{A}\mathbf{p}$. Here, δ_k refers to an offset for the $\mathbf{p}$ vector to target the pertinent values. The parameter $N_{threads}$ refers to the total number of threads launched, and `Index` is a unique index assigned to each thread to indicate where to begin execution. The latter is computed as the sum of the thread index and the product of the block dimension and block index.

Each thread operates on Algorithm 5.2 and contributes to distinct entries, denoted by i, of the output vector $\mathbf{b}$. Thus, there are no possibilities for race conditions across threads, allowing the algorithm to fully exploit the massive parallelism afforded by GPUs without the need for thread synchronisation.

This sparse matrix-vector multiplication routine and other modifications of it (*e.g.* with subtraction) are used repeatedly in the CG workflow. For example, the product $\boldsymbol{A}\mathbf{p}$ on line 5 as well as $\boldsymbol{A}\mathbf{x}$ on line 1 in Algorithm 5.1.

Thus, the acceleration of the CG routine presented herein is attributed to two main features: the exploitation of the sparsity pattern of matrix $\boldsymbol{A}$ to reduce the formal complexity of matrix-vector contractions, and a synchronisation-free algorithm that efficiently leverages the parallelism of GPUs to further accelerate such contractions.

5.2.4 Dielectric Mapping

In this section, a description of the dielectric mapping adopted in the proposed PBE solver is provided, which improves on the approximate approaches employed in DelPhi and AMBER, as it provides a formal exact mapping.

The dielectric mapping involves the classification of grid points, specifically, whether they are inside or outside of the solute molecular surface. In DelPhi, an approximate iterative scheme is implemented and is based on the midpoints between a pair of neighbouring grid points.[9] Therefore, the classification of a grid point is dictated by the location of its six neighbouring midpoints. The simple cases are grid points whose midpoint neighbours are unanimously all within the surface ('in') or outside the surface

('out'). Grid points whose set of six midpoint neighbours is not entirely 'in' or 'out' undergo an iterative procedure. Let us denote these grid points as boundary grid points (BGPs). The algorithm iterates on a container comprising BGPs, and updates the status of its midpoint neighbours based on their distances from the solvent-accessible surface (SAS). If the distance of the midpoint to the SAS is less than the probe radius, it is marked as 'out' and 'in' for the inverse case. The definition of SAS is provided in Chapter 2 Section 2.3.3 (Figure 2.6).

With these updated classifications of the neighbouring midpoints, the status of the grid point is checked. If the grid point is no longer classified as a BGP, it is removed from the BGP container and the procedure continues to the next BGP. The algorithm converges when there is no change in the BGP container between two iterations.[9]

On the other hand, AMBER's *pbsa.cuda* has multiple schemes to handle the assignment of dielectric constants across the dielectric interface.[10] Though AMBER includes a scheme identical to that used by DelPhi, the level set function-based dielectric model[10] was selected as it was found this to be the optimal dielectric model among the available options. A detailed justification is provided in Appendix C.

In this level set function-based approach, a level set function is used to implicitly characterise the molecular surface. Therein, the BGPs are identified where the level set function is zero. The algorithm begins by labeling grid points as either 'solvent' or 'solute' where 'solute' grid points have positive level set values and 'solvent' grid points have negative set values. BGPs therefore correspond to the region where the level set function is zero, specifically, the intersection of a boundary grid edge and the molecular surface. A quadratic function is then used to model the level set function near this region. This function is spanned by three points, two that define the boundary grid edge, and a third point obtained by extending one of these two points by a grid spacing $\pm h$ in one of the x, y or z directions. The intersection point (BGP) corresponds to a root of the quadratic function.[10]

In contrast, the new implementation presented herein employs a method that directly computes the distance between the grid point and the solvent-excluded surface (SES) to distinguish whether or not it lies outside the molecular surface. Algorithm 5.3 describes the scheme employed to classify grid points. Here, the r_{probe} parameter denotes the solvent probe radius. The algorithm begins by iterating across all grid points and initialising the variable `inside_ses` as `False`. Then it iterates across atoms, calculating the distance d_j between each grid point and the current atom. If d_j is less than the atom's van der Waals (vdW) radius r_i, the point is classified to lie within the SES (lines 4 to 8). In the proposed implementation, the three-dimensional space is partitioned into boxes and the set of atoms iterated over in Algorithm 5.3 refer to atoms belonging to the box grid point i is located in. In this way, only atoms within the vicinity of grid point i are considered and the two for loops (lines 3 to 9 and lines 10 to 26) do not grow

Algorithm 5.3 Classification of grid points.

Require: r_{probe}
1: **for** each grid point i **do**
2: `inside_ses`$_i \Leftarrow$ `False`
3: **for** each atom j **do**
4: $d_j \Leftarrow$ distance between grid point i and atom j
5: $r_j \Leftarrow$ vdW radius of atom j
6: **if** $d_j < r_j$ **then**
7: `inside_ses` $\Leftarrow$ `True`
8: **end if**
9: **end for**
10: **for** each atom j **do**
11: **for** each atom $k > j$ **do**
12: $d_j \Leftarrow$ distance between grid point i and atom j
13: $d_k \Leftarrow$ distance between grid point i and atom k
14: $r_j \Leftarrow$ vdW radius of atom j
15: $r_k \Leftarrow$ vdW radius of atom k
16: **if** $(d_j < r_j + r_{probe})$ or $(d_k < r_k + r_{probe})$ **then**
17: $C \Leftarrow S_j \cap S_k$
18: **if** $C \notin \emptyset$ **then**
19: $d_C \Leftarrow$ distance between grid point i and C
20: **if** grid point $i \in \Omega_{jk}$ and $d_C > r_{probe}$ **then**
21: `inside_ses`$_i \Leftarrow$ `True`
22: **end if**
23: **end if**
24: **end if**
25: **end for**
26: **end for**
27: **end for**

with system size.

Next, the algorithm moves to the case of grid points located within the SES but not within the spheres spanned by the vdW radius of any atom. As an example to illustrate this, consider the shaded purple area in Fig. 5.2 located between two atoms j and k. To determine whether a grid point lies in this region, pairs of atoms are iterated over (lines 10 and 11), and the distance between grid point i and each of the two atoms is calculated (lines 12 and 13).

Next, these distances are used to check whether the grid point i lies within the "expanded" spheres of atoms j and k (lines 16). An "expanded" sphere has a radius equal to the sum of the vdW radius of the atom of interest and the solvent probe radius r_{probe} (see circles outlined in dashed lines in Fig. 5.2). It is important to note that Fig. 5.2 is an example case, however, in general, the purple region corresponds to the region inside Ω (shown red in Fig. 5.2) but outside both spheres spanned by the vdW radius and within the "expanded spheres". In this way, it checks for whether the grid point i lies within the "expanded" spheres of atoms j and k.

A curve C on line 17 is defined as the circle formed from the intersection of the two "expanded" spheres (S_j and S_k) corresponding to atoms j and k. If C exists, that is, the two "expanded" spheres of atoms j and k do overlap, it proceeds to computing the distance (d_C) between grid point i and C.

In addition to all the checks mentioned above, two final checks are performed (line 20) to determine whether grid point i is located in the purple region of Fig. 5.2. First, it checks if d_C is greater than r_{probe} and grid point i lies within a triangular region denoted

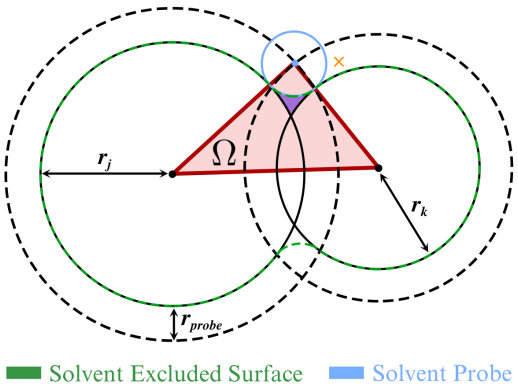


Figure 5.2: 2D representation of regions relevant to the dielectric mapping procedure. Circles outlined in dashed black lines correspond to "expanded" spheres with radius equal to the sum of the vdW of the atom and the probe radius. Ω denotes the triangular region shaded in red. Orange cross represents a counter case described in the text. This diagram is not to scale.

as Ω in Fig. 5.2 and Algorithm 5.3. The reasoning for this is best illustrated through Fig. 5.2 where d_C is the distance between grid point i and the center of the solvent probe which is located on C–this point is denoted as p. Since the probe has radius r_{probe}, to be located in the purple area, $d_C > r_{probe}$ must be true. However, the criterion of $d_C > r_{probe}$ alone is not sufficient to ensure that p lies within the purple area. For example, let us consider the case of a grid point located at the orange cross on Fig. 5.2; this point satisfies the criterion of $d_C > r_{probe}$ but is evidently not in the purple region or SES. Therefore, the additional criterion that a grid point i must also be in Ω is included to confirm its location in the SES.

Later, in Section 5.4, it is demonstrated that although this more computationally costly but exact dielectric mapping method is employed, the proposed PBE solver exhibits superior computational performance over DelPhi and AMBER's *pbsa.cuda*.

5.3 Computational Details

5.3.1 Molecular Dataset

In this section, a description of two molecular datasets is given, on which PB calculations will be performed using the three different software DelPhi, AMBER, and EXESS to assess both accuracy and computational efficiency.

Minnesota Dataset

To validate the accuracy of the PB algorithm presented in this work, it is first applied to structures taken directly from the Minnesota Solvation Database (MN12).[249] The Min-

nesota Dataset comprises 3,037 experimental solvation energies spanning 790 solutes (charged and neutral) and 92 solvents. Since the application focuses on protein-ligand systems solvated in water, only a subset of MN12 where the solvent is water is considered, which includes over 500 solutes. Figure 5.3 shows the distribution of the structures. The average solute size in the dataset is 15.7 atoms.

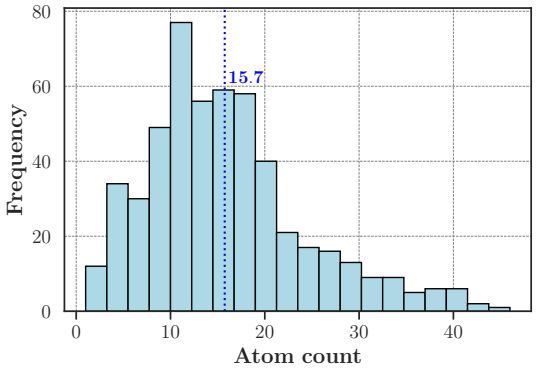


Figure 5.3: Distribution of system sizes in the Minnesota dataset. Average size is indicated by a vertical blue line and the corresponding value is listed.

Protein-Ligand Dataset

The main benchmark dataset consists of a range of biologically relevant protein-ligand complexes. Such systems were extracted directly from the protein-ligand benchmark dataset curated by Hahn *et al.*[250,251] The final dataset employed in this study comprises 335 protein-ligand complexes, with ligands bound to 13 distinct target proteins. Table 5.2 provides an overview of the protein-ligand dataset in terms the target proteins, the corresponding number of ligand complexes, and their total number of atoms including hydrogens.

The selected set of protein-ligand systems was chosen for its relevance to drug development. For instance, the overexpression of myeloid cell leukemia 1 (MCL1) inhibits apoptosis and promotes tumor cell survival[252]. Inhibition of tyrosine kinase 2 (TYK2) is a promising strategy for treating inflammatory diseases, including inflammatory bowel disease[253].

Additionally, the dataset has been curated by Hahn *et al.* with the aim of containing only protein-ligand structures that exhibit large dynamic ranges, high quality structures, with an Iridium score of at least 'mildly trustworthy', and sourced from biophysical assay conditions.[251]

With these structures from Hahn *et al.* equilibrium MD simulations in explicit solution are performed to account for the presence of explicit solvent (water) molecules, as detailed in the following section.

Target	No. of ligands	No. of atoms[a]
CDK2	10	9031
CDK8	30	10456
EG5	27	5473
HIF2A	37	3579
MCL1	25	2448
P38	29	5636
PDE2	21	5496
PFKFB3	32	7259
PTP1B	22	4821
SHP2	24	8363
SYK	44	4457
thrombin	21	4627
TYK2	13	4652

[a]Number of atoms and includes hydrogens.

Table 5.2: Target proteins and the corresponding number of ligands in the protein-ligand-benchmark dataset used in this study.

5.3.2 Protein-Ligand Structure Preparation

In this section, the details on the equilibrium molecular dynamics (MD) simulations are provided. These simulations are performed on the protein-ligand systems presented in Section 5.3.1 as part of the structure preparation. A detailed description of MD simulations is presented in Chapter 2 Section 2.2.

GROMACS 2023.2 was used for all the MD simulations with three-dimensional periodic boundary conditions. MD simulations were performed on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center of the United States Department of Energy.

Protein-ligand complexes were first solvated in a rhombic dodecahedron water box with buffer width of 15 Å. Counterions (Na^+ and Cl^-) were added to neutralise the system. The AMBER ff03 force field was employed in all simulations for the protein.[254] Ligands were parameterised with the second generation general AMBER force field (GAFF2).[255,256] ChElPG charges[257] computed with Q-Chem[223] at the B3LYP/cc-pVTZ level were used for ligands. The short-range non-bonded interactions were computed for atom pairs within 10 Å. The particle mesh Ewald (PME) method was employed to model long range forces, with fourth-order cubic interpolation and 1.2 Å grid spacing. All covalent bonds involving hydrogen atoms were constrained with the LINCS method.[258]

Next, two minimisation routines were employed: 1) First, all backbone carbon and nitrogen atoms were restrained using a strength of 1000 kJ $mol^{-1}nm^{-2}$, and 5000 steepest descent steps were performed; 2) Second, all atoms were optimised without any constraints for 3000 steps of steepest descent and 2000 steps of conjugate gradient min-

imisation. After minimisation, each system was heated from 0 to 300 K in the *NVT* ensemble over a period 100 ps and then relaxed for 100 ps in an *NpT* ensemble MD simulation with a temperature of 300 K and pressure of 1 atm. Temperature was controlled by the Bussi-Donadio-Parrinello thermostat,[259] and the pressure was controlled using the Berendsen barostat[260] in the *NpT* simulations. The time constant for the temperature and pressure coupling was kept at 0.2 ps and 2 ps, respectively. The final protein-ligand structure from the *NpT* equilibrium trajectory was used for PB calculations whose details are provided in the next section. The selection of the number of minimisation steps, the solvers used for the latter, as well as the timing durations of the heating and equilibrium steps is consistent with standard protocol used across numerous studies in this field.[66,242–244,261–263]

5.3.3 Poisson-Boltzmann Calculations

In this study, the accuracy and computational performance of the proposed PBE solver against the CPU-based DelPhi and GPU-based AMBER *pbsa.cuda* software are compared. DelPhi employs OpenMP to utilise the multi-core capability of CPUs and was selected to serve as the CPU benchmark. On the other hand, AMBER's *pbsa.cuda* will serve as the GPU benchmark. DelPhi 8.5.0[38] and AMBER 22.0[264] were used. An overview of the two parallel programming models OpenMP and CUDA are provided in Chapter 2 Section 2.1.2.

Although the solvation energy comprises the polar and non-polar contributions as shown in Eq. (2.49), only the polar term is considered as this study concerns solving the linear Poisson-Boltzmann equation. For conciseness, henceforth, all mentions of solvation energies (G_{solv} and ΔG_{solv}) in this chapter will refer to the polar component only unless stated otherwise.

Unless stated otherwise, the following parameters were held constant across all PB computations with DelPhi, *pbsa.cuda* and EXESS: a temperature of 298 K for the MN12 dataset and 300 K for the protein-ligand dataset, an ionic strength of zero, a ratio of the solute dimension to the lattice dimension of 0.5, dielectric constants of $\varepsilon_{solvent} = 78.54$ (water) and $\varepsilon_{solute} = 1$, a grid spacing of 0.4 Å, a solvent probe of 1.4 Å, Bondi van der Waals radii were used for atomic radii.[265] The boundary conditions were computed from the sum of the Debye-Hückel potentials of all charges.

The Jacobi preconditioned CG solver was adopted for *pbsa.cuda*. A convergence threshold of 10^{-3} was used for both *pbsa.cuda* and EXESS as they are defined similarly, computed based on the L_2 norm of the residual potential vector.[12] On the other hand, a "tighter" threshold (10^{-4}) was used for DelPhi as the convergence depends on the root mean square (RMS) change of potential instead of the L_2 norm. Maximum number of iterations in EXESS and AMBER were both set to 10000. All calculations converged

within the maximum iteration count. Number of iterations in DelPhi is determined automatically *in situ*. All other parameters were set to their default values. Representative input files for DelPhi and AMBER have been provided in Appendix C.

Hardware and Performance Measurement

All PB calculations were performed on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center of the United States Department of Energy. All timing results for DelPhi were obtained using the program's multicore CPU parallel capabilities on a single 64-core AMD EPYC 7763 CPU. All timing results for GPU-based solvers (*pbsa.cuda* and EXESS) were obtained on a single NVIDIA A100 40 GB GPU.

To assess the performance of EXESS against DelPhi and AMBER on the protein-ligand dataset, the following speedup metric is employed

$$\text{speedup} = \frac{t}{t_{EXESS}} \tag{5.3}$$

where t_{EXESS} and t denote wall times of PB calculations performed with EXESS and DelPhi/AMBER, respectively.

To account for the different hardware utilised, the energy consumption of the three different software are compared. Energy consumptions were measured using Linaro Forge's Performance Report which uses Cray HSS energy counters to monitor power usage on the Perlmutter supercomputer.

For each protein-ligand system, three separate PB computations are required according to Eq. (5.1): (1) the bound protein-ligand complex, (2) the isolated protein, and (3) the isolated ligand. The timings and energy consumptions presented in this work correspond specifically to the PB calculations performed on the bound protein-ligand complexes.

For all structures belonging to the Minnesota dataset and ligands in the protein-ligand dataset (described in Section 5.3.1), ChElPG charges were used in all PB calculations. All ChElPG charges were calculated at the B3LYP/cc-pVTZ level using Q-Chem[223] version 6.0 with a convergence threshold of 10^{-8} a.u. Partial charges from the AMBER ff03 force field were used for protein atoms.[254]

Non-Polar Contributions

For the Minnesota dataset, the non-polar component of G_{solv} is included in addition to the polar component obtained from solving the PBE in order to provide a fairer comparison against experimental G_{solv} values. The non-polar contribution is computed with Eq. (2.14) and the values used for γ and β are 2.26778 kJ mol^{-1} nm^{-2} and 3.84928 kJ mol^{-1},

respectively, consistent with those used in the AMBER package.[59]

The solvent accessible surface area (SASA) is computed once per system in the Minnesota dataset and these values are used in the comparison across all three software in Section 5.4.2.

The SASA computation is approximate and is calculated as the sum of atomic contributions. This contribution is calculated first by discretising the surface of the atomic sphere using a Fibonacci lattice[266] where the number of discretisation points per atomic sphere is $2n_f + 1$ with $n_f = 1000$. Following, the proportion of points not found in any other atom sphere are counted. This proportion is then multiplied with the surface area of the atomic sphere – this provides the individual atomic contribution to the SASA.

5.3.4 Dielectric Mapping Validation

To validate the proposed dielectric mapping scheme as described in Section 5.2.4, a method that detects misclassifications in the dielectric model by comparing the distance between a given point and the solvent accessible surface (SAS) is utilised. The validation method is as follows.

First, a discretisation of the SAS is computed and stored. This discretisation is computed by discretising the surface of the "expanded" sphere of each atom. The discretisation of each "expanded" atomic sphere is done using a Fibonacci lattice approach.[266] Here, $2n_f + 1$ (before removal of points) points per atom are generated. Next, any generated point that is within the "expanded" sphere of any other atom is removed from the discretised set.

This discretised SAS is used to compute the distance of a point to the SAS and thus verify the correctness of the classification of a grid point as within (*i.e.* 'solute') or outside the SES (*i.e.* 'solvent'). This procedure is carried out as follows. First, note that the two easy cases are: if a point is outside of every "expanded" atomic sphere, it is certainly outside the SES and is given a nominal distance of infinity; and if a point is inside any atomic sphere whose radius is spanned by the atom's vdW radius, then is it certainly inside the SES (*i.e.* 'solute') and is assigned a nominal distance of a fixed negative number. Next, a naive search across the remaining points on the discretised SAS is performed, recording the minimum distance to any of these points. If this minimum distance is greater than or equal to the probe radius, this point is classified to be inside the SES and assign it the same nominal distance value as the other points inside the SES. If the minimum distance is less than the probe radius, then this point is classified to be outside the SES (*i.e.* 'solvent') and assign its distance value to be equal to the minimum distance just recorded.

This validation procedure was performed on 50 randomly selected systems from the Minnesota set and two protein systems: MCL1 and HIF2A. $n_f = 20,000$ and $n_f = 1,000$

are used for the Minnesota systems and proteins, respectively. Due to this validation procedure being relatively expensive, this analysis is limited to the two smallest protein systems and uses a smaller n_f. The different dielectric mapping used across the three software are compared. Using this validation method and the said n_f discretisation parameter, the number of points misclassified across the different dielectric models are reported in Section 5.4.1.

5.4 Results and discussion

5.4.1 Dielectric Mapping Validation

In this section, an analysis of the dielectric mappings used across DelPhi, AMBER and EXESS is presented. The validation method presented in Section 5.3.4 is employed.

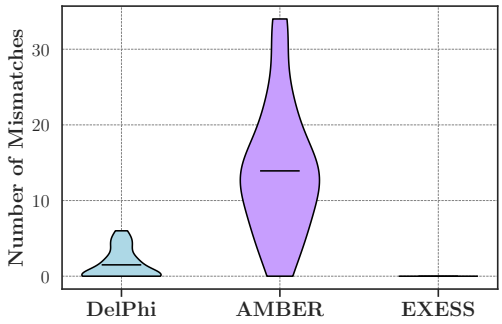


Figure 5.4: Distribution of the total number of mismatches of a point's label between the dielectric mapping schemes employed in each software with the validation method (Section 5.3.4) across the 50 randomly selected systems from the Minnesota set. All systems exhibit 0 for the dielectric model employed in EXESS. Averages are indicated by black horizontal lines.

Figure 5.4 displays the distribution of the total number of mismatches between the dielectric mapping scheme with the validation method as detailed in Section 5.3.4 on the MN12 systems, namely, the number of points where the validation and dielectric model disagreed. Across all 50 systems, the minimum number of points incorrectly labeled is consistently zero with EXESS. The same cannot be said for either of DelPhi or AMBER. AMBER exhibits the largest range (0 to 34) whereas DelPhi ranges between 0 and 6.

Such an outcome is not surprising given that the dielectric model utilised in EXESS directly calculates the distance between a grid point and the SES. On the other hand, the midpoint- and level set-based methods employed by DelPhi and AMBER, respectively, are more approximate schemes and therefore are accompanied by a larger number of mismatches.

These mismatches become accentuated for larger systems comprising thousands of atoms such as proteins. As shown in Table 5.3, which lists the number of misclassified points in the dielectric models for the MCL1 (2,448 atoms) and HIF2A (3,579 atoms), DelPhi and AMBER exhibit significantly higher misclassifications than EXESS; DelPhi and AMBER are accompanied with tens (43 to 55) and hundreds (216 to 299) of mismatches, respectively, significantly higher than that of EXESS which remains zero.

Protein	DelPhi	AMBER	EXESS
MCL1	43	216	0
HIF2A	55	299	0

Table 5.3: Number of misclassified points in MCL1 and HIF2A by the dielectric mapping schemes used across all three software. Averages are indicated by black horizontal lines.

Such results not only validate the dielectric model employed in EXESS but also serves to highlight the superior accuracy of the proposed dielectric model over the approximate schemes used in DelPhi and AMBER. Furthermore, despite employing this more computationally burdensome dielectric mapping scheme, it is later demonstrated that the proposed approach displays superior wall times than either of DelPhi or AMBER's *pbsa.cuda* on practical application protein systems. Thus, having validated the proposed dielectric model, the latter is applied to PB calculations in EXESS in the next section.

5.4.2 Poisson-Boltzmann Calculations

In this section, the accuracy and computational performance of the proposed algorithm in comparison to DelPhi and AMBER's *pbsa.cuda* are compared. Section 5.4.2 begins by verifying the accuracy of the proposed PBE solver on the Minnesota dataset which includes systems with less than 50 atoms, before its application to the protein-ligand dataset comprising systems with thousands of atoms.

Minnesota Dataset

In this section the accuracy of the proposed PBE solver on the Minnesota dataset is verified before proceeding with its application on the large protein-ligand systems.

Figure 5.5 shows the distribution of solvation free energy errors obtained with the different test programs relative to experimental results. For completeness, the G_{solv} obtained including the non-polar contributions is also displayed. The non-polar contributions included are exactly the same across the three test programs to isolate the effect of the polar contribution.

Considering only the polar contributions, all three programs exhibit similar G_{solv} errors; the mean absolute error (MAE) only varies between 13.7 and 14.8 kJ mol^{-1}. On

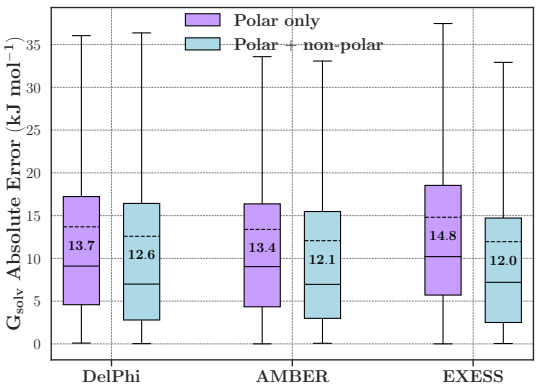


Figure 5.5: Distribution of absolute errors between experimental and predicted G_{solv} values for the Minnesota dataset using only the polar component (purple) and both polar and non-polar components (blue). Mean absolute errors are indicated by dashed horizontal lines and their values are listed.

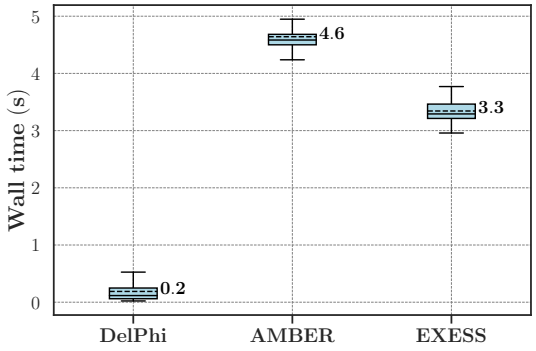


Figure 5.6: Distribution of PB calculation wall times on the Minnesota dataset. Mean timings are indicated by dashed horizontal lines and their values are listed. Timings for DelPhi were obtained on a single 64-core AMD EPYC 7763 CPU and timings for AMBER and EXESS were obtained on a single NVIDIA A100 40 GB GPU.

the other hand, with the addition of the non-polar contribution, a reduction in the MAE is observed across all three software; the MAE decreases by 1.1, 1.3 and 2.8 kJ mol^{-1} in DelPhi, AMBER and EXESS, respectively.

These discrepancies between the predicted energies across the different software can be ascribed to the different dielectric mapping employed across all software, yielding varying coefficient matrices $\boldsymbol{A}$ of Eq. (2.52).

Nevertheless, the purpose of this section is to verify the accuracy of the proposed approach and the comparable accuracy of between EXESS, AMBER and DelPhi has been demonstrated.

Figure 5.6 compares the wall-time distributions obtained using DelPhi, AMBER, and EXESS. The mean wall times for AMBER (4.6 s) and EXESS (3.3 s) are approximately an order of magnitude higher than that of DelPhi (0.2 s). This result is unsurprising and is likely attributable to the fixed overheads accompanying GPU implementations (*i.e.* AMBER and EXESS) including device initialisation, kernel-launch latency, and host–

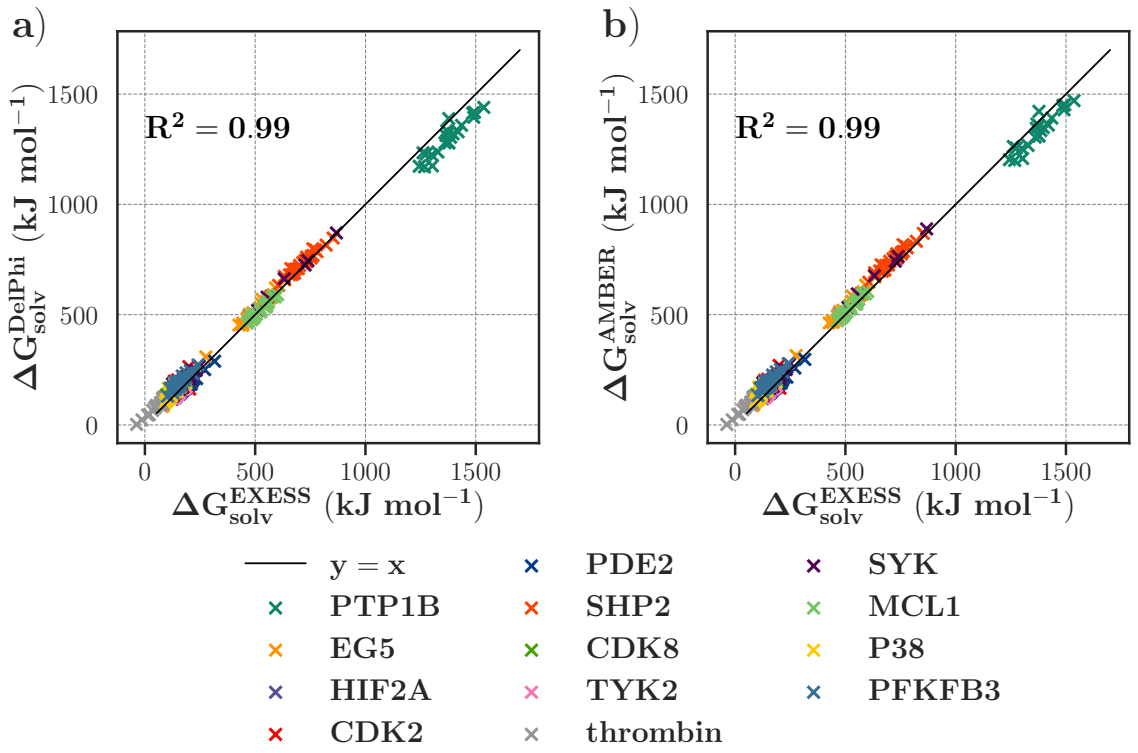


Figure 5.7: Correlation between ΔG_{solv} of protein-ligand systems attained with EXESS and those of a) DelPhi; b) AMBER. Energies are calculated according to Eq. (5.1).

device communication, as well as the limited parallel workload associated with the very small molecules belonging to the Minnesota dataset (average size of 15.7 atoms), as shown in Fig. 5.3. However, as demonstrated in the following section, for larger protein-ligand complexes, the computational benefit gained by leveraging GPU acceleration significantly outweighs these overhead costs.

Having established the accuracy and baseline performance of the proposed PBE solver, the next section involves its application to large protein systems.

Protein-Ligand Dataset

In this section, the performance of EXESS relative to DelPhi and AMBER's *pbsa.cuda* module are evaluated by comparing their solvation energies (Eq. (5.1)) and computational timings.

Unlike in the previous section (5.4.2), which compared calculated results against experimental data, no experimental benchmarks are used here. This is because only experimental values of ΔG_{bind} are available in the literature, which include terms such as ΔE and ΔS (see Eq. (2.12)). This study, however, focuses exclusively on the polar contribution to the solvation free energy.

Figure 5.7 shows the correlation of the protein–ligand solvation energies computed with EXESS against those obtained from DelPhi and AMBER. Two primary observa-

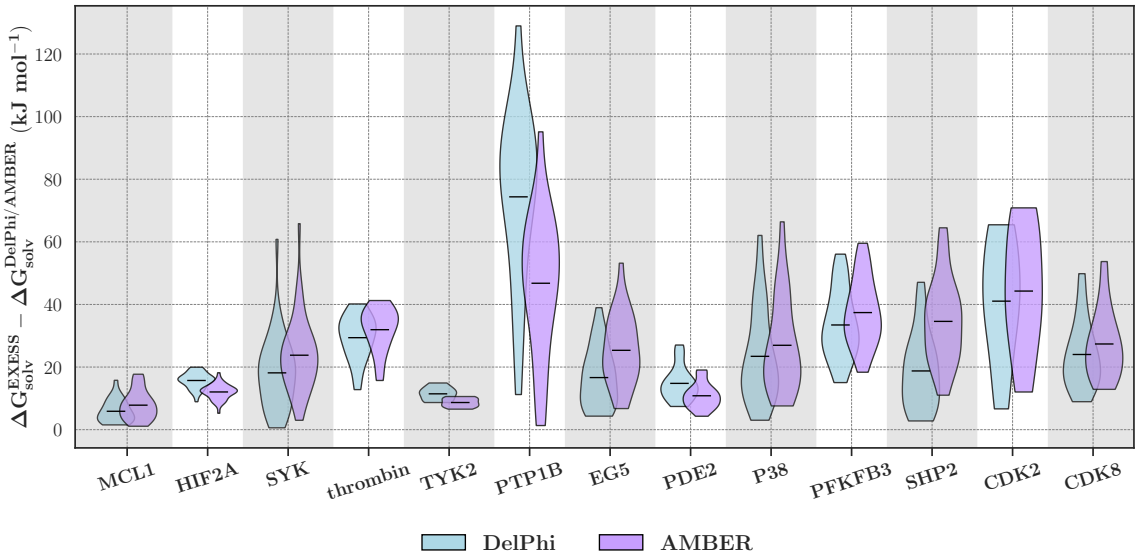


Figure 5.8: Distribution of absolute solvation free energy differences relative to EXESS for each target protein system. Target proteins are sorted in order of increasing size. Averages are indicated by black horizontal lines.

tions emerge. First, there is generally good partitioning of the energies with respect to the protein targets. For instance, the PTP1B solvation energies are all clustered together where $\Delta G_{solv} > 1100$ kJ mol^{-1} irrespective of the program employed (EXESS, DelPhi, and AMBER).

Second, across all 335 protein–ligand systems, there is a strong correlation between ΔG_{solv} values from EXESS and those from DelPhi or AMBER, with nearly perfect Pearson correlation coefficients relative to both DelPhi ($R^2 = 0.99$) and AMBER ($R^2 = 0.99$).

The similarity in the correlation between EXESS, DelPhi, and AMBER are also reflected in the solvation free energy deviations shown in Fig. 5.8. Across all 13 target protein systems, the distribution of the ΔG_{solv} deviations between both DelPhi and AMBER are noticeably similar. A salient example of this is the TYK2 system whereby the average deviation in ΔG_{solv} between EXESS and AMBER is only 2.8 kJ mol^{-1} lower than that of EXESS and DelPhi.

On the other hand, PTP1B exhibits the greatest absolute difference in the ΔG_{solv} deviations between EXESS with DelPhi and AMBER. As shown in Fig. 5.8, the average ΔG_{solv} deviation between EXESS and DelPhi is 27.6 kJ mol^{-1} higher than that of EXESS and DelPhi. Though these absolute energy deviations are large, the relative deviation with respect to the ΔG_{solv} values themselves is small. As observed from Fig. 5.7 PTP1B exhibits the highest ΔG_{solv} values across all 13 protein targets where all ΔG_{solv} are greater than 1100 kJ mol^{-1}. Compared to the ΔG_{solv} values calculated with EXESS, these energy deviations in Fig. 5.8 amount to an average difference of 3.4% and 5.4% with AMBER and DelPhi, respectively.

The underlying cause of these discrepancies can be attributed to multiple factors,

some of which have been discussed in Section 5.4.2 including the different algorithms used between DelPhi (SQR) and AMBER's *pbsa.cuda*/EXESS (CG) as well as the different dielectric mappings employed. Discrepancies between DelPhi and AMBER have been reported within the literature previously.[11,267] For example, one study reported differences ranging from 8.7 to 49.0 kJ mol^{-1} between solvation energies attained with DelPhi and AMBER on protein systems containing between 279 and 2803 heavy atoms.[11]

Similar results are observed between DelPhi when compared to AMBER; the correlation between the solvation free energies calculated with DelPhi and AMBER, which is shown in Appendix C Fig. C.1, is nearly identical to Fig. 5.7b.

Having shown the similarity in the accuracy of the ΔG_{solv} values calculated across all three software, the computational performance of PB calculations in EXESS is examined before its comparison to AMBER and DelPhi.

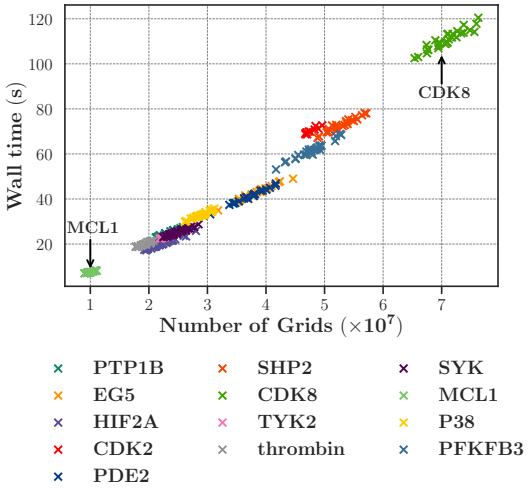


Figure 5.9: Effect of grid size on wall times of PB calculations with EXESS.

Figure 5.9 displays the wall times of PB calculation with EXESS with the corresponding number of grid points. Generally, a positive association between the grid point count and wall times is observed. Such outcome is not surprising since the grid point count dictates the size of the coefficient matrix $\boldsymbol{A}$ in Algorithm 5.1. Also of notice is the correlation between the number of grids and protein size. Larger proteins such as CDK8 (10,456 atoms) exhibit greater number of grids (mean of 7.09×10^7 grids) and wall times (greater than 100 s) compared to smaller proteins such as MCL1 (2,448 atoms) which comprise lower number of grids (mean of 1.00×10^7) and reduced wall times (less than 20 s).

Next, the computational performance of EXESS against DelPhi and AMBER's *pbsa.cuda* are compared. Figure 5.10 presents the distribution of wall times for PB computations using DelPhi, AMBER's *pbsa.cuda*, and EXESS. As shown in Fig. 5.10, EXESS consistently outperforms both AMBER and DelPhi, often achieving wall times an order of magnitude lower across a number of the target systems. This is particularly evident in

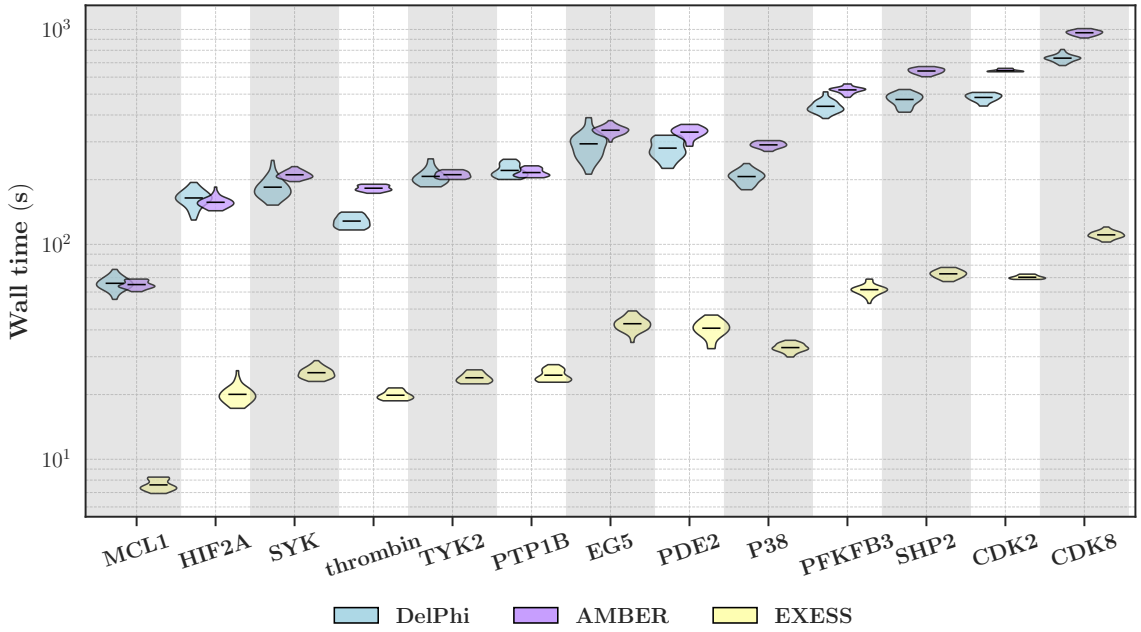


Figure 5.10: Distribution of PB calculation wall times on the protein-ligand dataset using different software for each target protein system. Target proteins are sorted in order of increasing size. Averages are indicated by black horizontal lines. Timings for DelPhi were obtained on a single 64-core AMD EPYC 7763 CPU and timings for AMBER and EXESS were obtained on a single NVIDIA A100 40 GB GPU.

the lower panel of Fig. 5.10, which highlights protein–ligand complexes ranging from 5,473 to 10,456 atoms. For these systems, all EXESS PB computations complete in under 125 seconds (see Fig. 5.9), whereas DelPhi and AMBER exhibit wall times of at least 179 and 270 seconds, respectively.

In contrast, no definitive trend emerges when comparing AMBER and DelPhi directly. In some cases, DelPhi is faster—for example, for the PDE2 system, average wall times were 280.4 seconds with DelPhi and 332.9 seconds with AMBER. However, across most protein targets, the PB wall times for AMBER and DelPhi are comparable.

A notable and apparent discrepancy arises when comparing the AMBER timings observed in this work with those reported by Qi *et al.*[12] Despite using the same solver (the Jacobi-preconditioned conjugate gradient), similar grid spacing, and convergence thresholds, Qi *et al.* reported GPU PB calculation times of the order of 1 second for similarly sized protein systems. In contrast, the measurements herein frequently exceeded 100 seconds. To clarify on this aspect, it is important to reiterate that the timings reported herein correspond to the full runtime of each program. Specifically, the discrepancy in wall times is due to the timings reported by Qi *et al.*[12] comprise only the solver routine, excluding the time spent setting up data to be transferred and executed on the GPU. Profiling confirms that a substantial portion of AMBER's *pbsa.cuda* code–specifically the grid initialisation step–is not offloaded to the GPU (see Appendix C

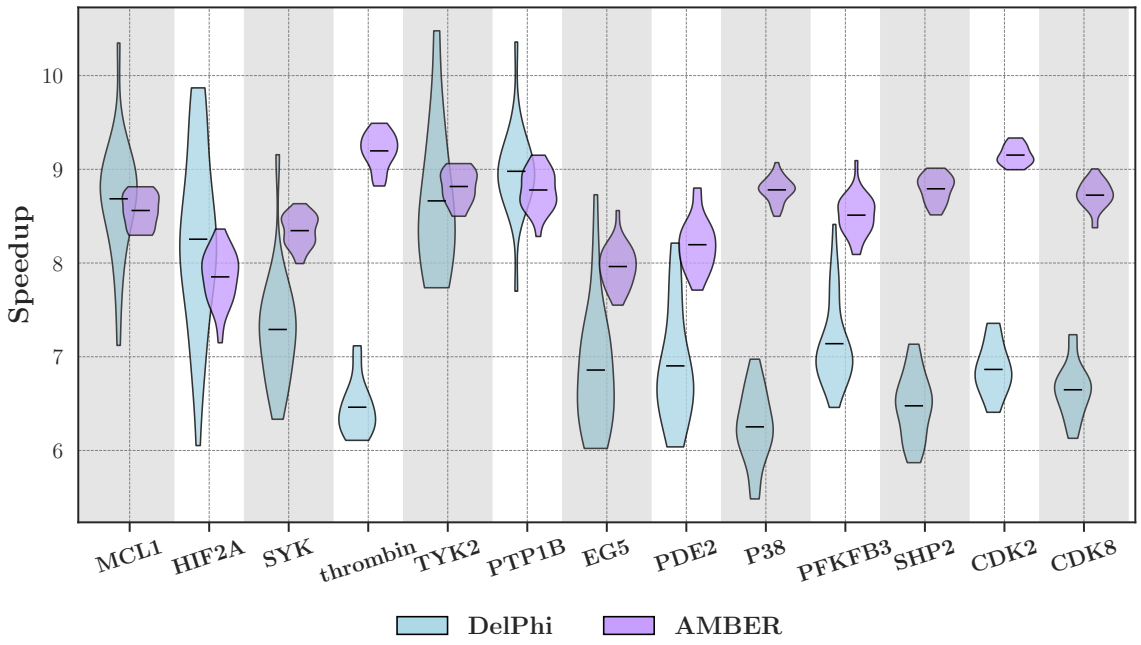


Figure 5.11: Distribution of wall time speedups of EXESS relative to DelPhi and AMBER for each target protein system. Speedups are calculated according to Eq. (5.3). Target proteins are sorted in order of increasing size. Averages are indicated by black horizontal lines. Timings for DelPhi were obtained on a single 64-core AMD EPYC 7763 CPU and timings for AMBER and EXESS were obtained on a single NVIDIA A100 40 GB GPU.

Fig. C.2). In EXESS, by contrast, this step is GPU-accelerated. Detailed profiling data provided in Appendix C show that only a small fraction of AMBER's total runtime is spent executing on the GPU, helping to clarify the source of the discrepancy.

The significant differences in wall times between EXESS and DelPhi/AMBER translate to substantial speedups for EXESS. Figure 5.11 displays the distribution of the speedups of EXESS PB calculations relative to DelPhi and AMBER. Across the majority of the protein target systems, EXESS exhibits a speedup greater than $7\times$ over AMBER and $5\times$ over DelPhi. This is especially important in the three largest protein targets—SHP2, CDK2, and CDK8—each comprising between 8,363 and 10,456 atoms which exhibit the highest wall times (see Fig. 5.10). In these cases, EXESS achieved speedups of 5.9 to 7.4$\times$ over DelPhi and 8.4 to 9.3$\times$ over AMBER. On average, across all 335 systems, EXESS delivers a 7.3$\times$ speedup over DelPhi and a 8.5$\times$ speedup over AMBER, based on wall time measurements.

It should be again emphasised that the wall times reported herein were obtained on a single structure per protein-ligand system as mentioned in Section 5.3.2. This is in contrast to large scale PBSA-based workflows where PB computations are often performed on hundreds or thousands of structures extracted from an MD trajectory per protein-ligand system. Thus, further highlighting the discrepancy in the computational

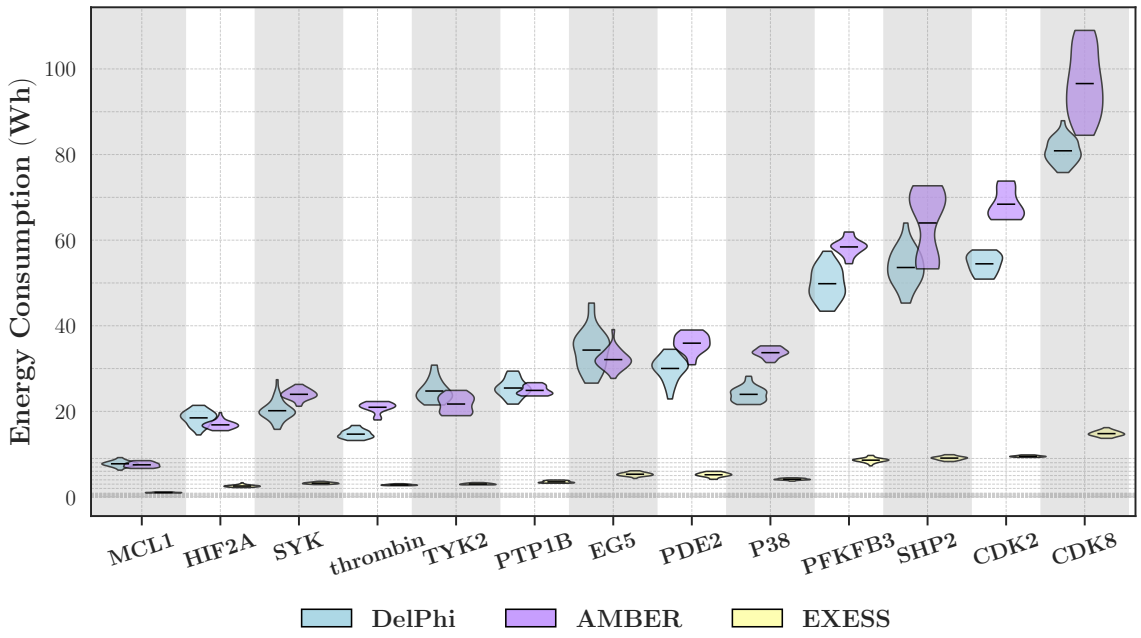


Figure 5.12: Distribution of energy consumption across DelPhi, AMBER and EXESS for each target protein system. Target proteins are sorted in order of increasing size. Averages are indicated by black horizontal lines.

expense between EXESS and both DelPhi and AMBER's *pbsa.cuda*, which when coupled with MM/PBSA or QM/PBSA studies on practical application protein-ligand systems proves to be extremely computationally costly.

It is important to account for the differences in hardware used to perform the PB calculations: DelPhi was executed on CPUs, whereas both AMBER and EXESS utilised GPUs. To enable a fairer comparison of computational efficiency across platforms, the energy consumption across each software are compared.

Figure 5.12 displays the energy consumption across all three software. Across all systems, both DelPhi and AMBER consistently exhibit higher energy consumptions than EXESS. Energy consumptions varied between 0.9 and 21.5 Wh across all PB calculations with EXESS whereas computations with DelPhi or AMBER required between 6.3 and 109.0 Wh.

Collectively, these results underscore the strong computational performance of EXESS's double-precision GPU-based PB solver. Compared to DelPhi, a multi-core CPU implementation, and AMBER's single-precision GPU code *pbsa.cuda*, EXESS achieves superior timing and energy efficiency while maintaining accuracy comparable to both DelPhi and AMBER.

5.5 Conclusion

In this Chapter, a novel GPU-accelerated solver implemented within the Extreme-scale Electronic Structure System (EXESS) quantum chemical software package is introduced, specifically designed for efficiently solving the linearised Poisson–Boltzmann equation (PBE). This implementation directly addresses key limitations of existing PBE solvers commonly employed in computational drug discovery, offering substantial improvements in computational speed, numerical accuracy, and energy efficiency. The proposed PBE solver utilises a Jacobi-preconditioned conjugate gradient algorithm that efficiently exploits the inherent sparsity of the discretised PB linear system, and is implemented without reliance on external linear algebra libraries. Furthermore, the presented solver incorporates a rigorous and formally exact dielectric mapping scheme, and operates entirely in double-precision arithmetic to ensure numerical robustness and precision, particularly when applied to large molecular systems typical of contemporary drug-design applications.

Extensive benchmarking of the proposed solver's performance and accuracy was conducted against established PBE solvers—DelPhi (CPU-based, double-precision) and AMBER's GPU-accelerated *pbsa.cuda* module (single-precision)—using two datasets: the Minnesota solvation database and a diverse set of 335 biologically relevant protein–ligand systems.

The validation on the Minnesota dataset demonstrated comparable accuracy to both DelPhi and AMBER. MAEs in calculated solvation free energies, considering the polar contribution alone, were 14.8 kJ mol^{-1} (EXESS), 13.7 kJ mol^{-1} (DelPhi), and 13.4 kJ mol^{-1} (AMBER). Inclusion of non-polar contributions further improved MAEs for EXESS (12.0 kJ mol^{-1}), DelPhi (12.6 kJ mol^{-1}) and AMBER (12.1 kJ mol^{-1}).

For large scale protein–ligand systems (2,448 to 10,456 atoms), EXESS demonstrated excellent consistency with both DelPhi (squared Pearson correlation $R^2 = 0.99$) and AMBER ($R^2 = 0.99$) in the solvation energies computed.

Performance benchmarking revealed substantial computational advantages for EXESS, achieving average wall-time speedups of $7.3\times$ over DelPhi and $8.5\times$ over AMBER across the entire protein–ligand dataset. When comparing energy consumptions, EXESS maintained a clear advantage, demonstrating superior energy efficiency compared to both CPU-based DelPhi and GPU-based AMBER. Average energy consumptions of 5.5, 33.1 and 38.1 Wh were observed with EXESS, DelPhi and AMBER, respectively.

Furthermore, though this GPU-accelerated PBE solver is developed for the purpose of water-solvated systems, namely protein-ligand systems, it can easily be extended to other types of solvents including hexane, ethanol or toluene—common solvents used in manufacturing and synthesis industries— by simply changing the input parameters (*e.g.* solvent dielectric constant).

Overall, the EXESS GPU solver provides a robust, double-precision, computationally efficient tool for solving the PBE in QM/PBSA workflows. Its enhanced performance and accuracy significantly extend the applicability of QM/PBSA approaches to large scale and high-throughput computational drug discovery efforts, ultimately facilitating faster, more reliable simulations of biomolecular complexes. Thus, in the next Chapter, this solver is combined with QM methods to construct a computationally feasible QM/PBSA workflow.

Predicting Binding Affinities of Protein-Ligand Systems with a QM/PBSA workflow

6.1 Introduction

Chapters 3, 4 and 5 have each explored techniques towards the realisation of Quantum Mechanical / Poisson-Boltzmann Surface Area (QM/PBSA) workflows. Specifically, Chapters 3 and 4 provide techniques to accelerate the calculation of QM gas phase energies whilst Chapter 5 focuses on accelerating the evaluation of solvation free energies. Putting these techniques together, this chapter addresses Aim 4 by performing a benchmark analysis on a proposed QM/PBSA workflow.

The adoption of any method at large scales (*i.e.* for virtual screening) necessitates systematic study of its performance across many target systems in order to identify limitations and to optimise protocols. Previously, due to the steep computational costs of QM and PBSA methods, QM/PBSA workflows have been unfortunately largely limited to single protein target studies.[66–69] With advancements in the field of QM[21–31,33,157,162,221,222,268–273] and PBSA[9,12,38,40–43] outside of this work, in addition to the techniques presented herein in Chapters 3 to 5, we are nearing the practical application of QM/PBSA workflows at large scales. Therefore, to provide the community with a comprehensive performance baseline of such approaches, a benchmark analysis of a proposed QM/PBSA workflow is performed on a set of diverse protein-ligand systems spanning nearly 200 protein-ligand systems. To the best of our knowledge, this work marks the first study where a QM/PBSA workflow has been applied across multiple target structures spanning hundreds of protein-ligand systems.

This Chapter begins with Section 6.2 by detailing the various components of the proposed QM/PBSA workflow including the quantum chemical and solvation models utilised as well as the dataset the performance evaluation is conducted on. In Section 6.3.1, the fragmented quantum chemical treatment is validated before its application to a set of 198 practical application protein-ligand systems which is presented in

Section 6.3.2. Therein, the influence of multiple parameters on the predicted binding affinities by QM/PBSA are considered, namely, starting structure, fragmentation level, solvation model. Its performance is also compared to other approaches including AFE, scoring functions, and other end point methods. Finally, Section 6.4 summarises the findings and concludes the study.

6.2 Computational Details

6.2.1 Molecular Dataset

In this section, a description is provided of the datasets employed herein for the validation the quantum chemical treatment as well as to assess the performance of the QM/PBSA workflow.

Gupta Dataset

To validate the fragmentation approach and the choice of basis set employed for the proposed quantum mechanical treatment, single point energy calculations were performed on structures taken directly from Gupta *et al.*[274] which will be referred to as the "Gupta dataset". Unless stated otherwise, all SPE computations are conducted at the Second-Order Møller–Plesset Perturbation (MP2) level of theory using the resolution-of-the-identity (RI)[275] approximation with either Dunning's correlation consistent cc-pVDZ or cc-pVTZ basis set.[213,214] An overview of the MP2 method and RI approximation is provided in Chapter 2 Section 2.4.

The Gupta dataset comprises two protein targets cyclin-dependent kinase 2 (CDK2) and interleukin-2-inducible T-cell kinase (ITK) paired with 13 and 14 ligands, respectively, and was selected for several reasons. First, these targets are of significance in drug discovery. For instance, CDK2 deregulation is linked to tumor growth, and growing evidence suggests that its inhibition has anticancer potential,[276] whereas inhibition of ITK is of interest in treating inflammatory disorders.[277] Second, the structures comprise less than 500 atoms, rendering full system Hartree-Fock calculations to be computationally feasible and enabling a direct comparison of the error incurred from fragmentation. Only full system computations at the HF level are performed as RI-MP2 is accompanied by high memory requirements. Third, experimental binding affinities are available for these structures and have been employed in other studies to measure the accuracy of end-point methods.[5,7,278]

Table 6.1 provides a summary of the calculations performed on the Gupta dataset. Briefly, the purpose of performing these calculations are to validate the fragmentation approach and the basis set used. It is known that large basis sets such as cc-pVTZ are important to accurately treat nonbonded interactions critical in binding.[7,69] However,

use of triple-ζ basis sets comes at a significantly higher cost and motivated us to examine whether comparable results could be achieved using smaller bases like double-ζ basis sets. In a similar vein, the performance between MBE2 and the more computationally costly MBE3 scheme are also compared.

	Full system	MBE2	MBE3
Theory level	HF	RI-MP2	RI-MP2
Basis set	cc-pVTZ, cc-pVDZ	cc-pVTZ, cc-pVDZ	cc-pVTZ, cc-pVDZ

Table 6.1: Summary of SPE calculations performed on the Gupta dataset.

Wang Dataset

Of particular interest is the Wang dataset which comprises eight target systems each comprising between 2,000 and 8,000 atoms, spanning 198 protein-ligand complexes as shown in Table 6.2. First compiled in 2015 by Wang *et al.* to demonstrate the accuracy of free energy perturbation methods which subsequently catapulted its popularity,[55] the associated dataset has become an established benchmark dataset to evaluate the performance of computational methods including scoring functions and alchemical free energy methods.[55,110,112,279–283] The Wang dataset is employed herein to measure the performance of the proposed QM/PBSA workflow.

Target	Number of Ligands
BACE	36
CDK2	16
JNK1	21
MCL1	42
P38	34
PTP1B	23
thrombin	[a]10
TYK2	16

[a]Number of ligands is one less than that in the Wang dataset due to convergence issues associated with an iodine-containing ligand.

Table 6.2: Description of the Wang dataset.

Structures of the Wang dataset were obtained from two different sources. The first involves the original dataset used by Wang *et al*[55] which was made available as part of a larger collection of datasets by Ross *et al* (Dataset I).[112] The second source includes structures originally provided by Zariquiey *et al.*[282] and subsequently optimised by Jalaie *et al* (Dataset II).[283] The two datasets differ across multiple areas including ligand placement, protein crystal structure and hydrogen position corrections. A full description of the differences can be found elsewhere.[283] An additional version of the Wang

dataset (Dataset III) is included which is obtained by performing an alternative protonation of the protein structures provided by Ross *et al.* This process was performed with the *pdb2gmx* module in GROMACS with the *-ignh* option which disregards the original hydrogen atoms in the protein system and re-inserts these according to standard geometry rules for distinct functional groups.[284]

Dataset I	Dataset II	Dataset III
Ross *et al.*[112]	Jalaie *et al.*[283]	Ross *et al.* with alternative protonation.

Table 6.3: Sources of the three Wang datasets used herein.

To summarise, three versions of the Wang dataset is explored as shown in Table 6.3. Multiple versions of the same dataset have been employed as it is known that the performance of computational methods are linked to the quality of initial structures. For example, Jalaie *et al.* demonstrated that the correlation of binding affinities predicted by semiempirical quantum-mechanical (SQM)-based scoring function was heavily dependent on the initial structures of the Wang dataset.[283] Thus, this motivated the study of Dataset I and II which exhibit significant structural differences, as well as Dataset III which differs to I only in the protonation states of the residues. These structures were used as the starting point for molecular dynamics simulations whose protocol is described in Section 6.2.3.

6.2.2 Performance Evaluation

To evaluate the performance of the QM/PBSA workflow, the Pearson correlation coefficient is used as the primary performance metric to measure the agreement between predicted binding affinities and experimental binding free energies. Experimental data for the Wang dataset comprises a mixture of ΔG_{bind}, IC_{50} and K_i values as listed in Table 6.4.

Target	Measurement type	Reference
BACE	K_i	[285]
CDK2	IC_{50}	[286]
JNK1	IC_{50}	[287]
MCL1	K_i	[288]
P38	IC_{50}	[289]
PTP1B	K_i	[290]
thrombin	ΔG_{bind}	[291]
TYK2	K_i	[292, 293]

Table 6.4: Experimental data for the Wang dataset.

Experimental binding free energies (ΔG_{exp}) were calculated as

$$\Delta G_{exp} \approx RT \ln(IC_{50}) \tag{6.1}$$

or

$$\Delta G_{exp} = RT \ln(K_i) \tag{6.2}$$

depending on the measurement type for each target system. R is the gas constant and $T = 300$ K.

6.2.3 Molecular Dynamics Simulations

In this Section, the molecular dynamics protocol conducted on the Wang dataset is provided. An overview of MD simulations is provided in Chapter 2 Section 2.2. GROMACS 2023.2 was used for all the molecular dynamics simulations. All simulations were performed on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center of the United States Department of Energy.

The MD setup, minimisation, heating and equilibrium routines followed the same protocol as that detailed in Chapter 5 Section 5.3.2. After the equilibrium simulations, production simulations were then performed in the *NpT* ensemble for 10 ns. Therein, Langevin dynamics was applied to regulate temperature and Parrinello–Rahman barostat[294] for the pressure. Snapshots were taken every 10 ps, generating 1,000 conformations for each of the protein-ligand systems.

Sampling Method

In this section, a description is provided of the sampling method employed to extract structures from the MD trajectories in which QM calculations are subsequently performed on. The MD simulation protocol described in the previous section generates 1,000 structures per protein-ligand complex. Due to the computational cost of quantum chemical calculations, a sampling procedure is adopted to reduce the number of frames QM calculations are performed on. The sampling approach employs the dihedral angle autocorrelation function (ACF) to provide a measure of the geometric similarity between conformations.

At time $j\Delta t$ where Δt is the time step (10 ps) the dihedral ACF is computed as the average of the correlation of all time points separated by $j\Delta t$ ps according to the following equation[295]:

$$C(j\Delta t) = \frac{1}{N_s - j} \sum_{i=0}^{N_s-1-j} \cos\left[\phi(i\Delta t) - \phi((i+j)\Delta t)\right] \tag{6.3}$$

where N_s is the total number of snapshots taken across the trajectory, i and j index

the time steps and ϕ denotes the dihedral angle. The *angle* module of GROMACS was employed to compute the dihedral ACFs.

A threshold value of 0.2 is used for the dihedral ACFs, specifically, an ACF value of 0.2 or less indicates that conformations (or frames) are sufficiently geometrically distinct to warrant sampling. The time at which this threshold is attained is recorded for each dihedral, and the overall sampling time for the specific protein-ligand complex is taken as the minimum of these times (*i.e.* where ACF < 0.2) across all the dihedral ACFs. The resulting number of conformations for each protein-ligand complex employed for the Wang dataset in the latter part of Section 6.3.2 is listed in Appendix D Table D.1.

Energetic quantities are computed for each of these conformers extracted according to the above sampling procedure, and the overall energetic quantity is taken as the average across all these conformers, thus, adjustments in weighting cancels out. To illustrate this, for example, consider for a particular system that a sampling frequency of 500 ps is obtained with the above procedure, meanwhile the frequency that snapshots are taken from the MD production run is 10 ps. Then conformers are extracted every 500 ps (*i.e.* every 50^{th} snapshot). This effectively means that every 50 consecutive conformers are represented by one conformer. Thus, the weighted average energy quantity would be represented as

$$\langle E \rangle = \frac{1}{N_{snapshots}} \sum (50 \cdot E_0 + 50 \cdot E_1 + \ldots + 50 \cdot E_n) = \frac{50}{N_{snapshots}} \sum_i E_i \tag{6.4}$$

where $\langle E \rangle$ is the weighted average energetic quantity, $N_{snapshots}$ is the total number of snapshots (extracted at a frequency of 10 ps), and E_i is the energetic quantity evaluated with conformer i, and i indexes every 50^{th} conformer.

$N_{snapshots}$ can be rewritten as the product of 50 and N^{50}_{frames} where the latter is the number of conformers/frames extracted every 500 ps. Then, the weighted average quantity simplifies to

$$\langle E \rangle = \frac{1}{N^{50}_{frames}} \sum_i E_i \tag{6.5}$$

which is just the average across all the sampled conformers. This can be extended to other sampling frequencies.

Since the area of greatest interest is the ligand and its binding pocket, the computation of the dihedral ACFs is limited to the ligand and the pocket residues. Furthermore, dihedrals containing hydrogen atoms are also excluded. The following definition of pocket residues is employed herein. A residue is classified as being part of the binding pocket if the following criterion is met[296]

$$R_{ij} < r_i^{vdW} + r_j^{vdW} + 2r_{probe} \tag{6.6}$$

where i and j denote heavy atoms on the protein and ligand, respectively, R_{ij} is the distance between the two atoms, r_i^{vdW} refers to the van der Waal radius for atom i, and r_{probe} is the probe radius. A value of 1.4 Å is used for r_{probe} corresponding to water and the Bondi van der Waal radii are used.[265] This definition of a pocket residue was selected over the traditional hard distance cutoff criterion as this incorporates information about the solvent and accounts for the identity of the atoms participating in bonds by way of the vdW radius.

Using the above definition, pocket residues are identified across each of the 1,000 frames sampled per protein-ligand system. The union of the residues that fulfill Eq. (6.6) identified throughout the MD trajectory is taken as the set of pocket residues for that specific protein-ligand complex. For each protein target, the final set of pocket residues used with the ACF sampling technique is defined as the union of pocket residues across all ligands associated with that target. In this way, a consensus binding pocket definition for each protein target is achieved. Henceforth, all mentions of pocket residues refer to this consensus set.

6.2.4 Single Point Energies

In this study, both fragmented and full system single point energy (SPE) calculations are performed. SPE computations were performed with the GPU-accelerated Extreme-scale Electronic Structure System (EXESS) quantum chemistry program.[22–29] All fragmented SPE calculations were performed on the Frontier supercomputer at the Oak Ridge Leadership Computing Facility, and full system SPE computations were performed on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center, both of which are part of the United States Department of Energy.

All SPE computations on the Wang dataset were performed with the resolution-of-identity (RI) approximation[275] of HF and MP2 with Dunning's correlation consistent cc-pVDZ basis set.[213,214] A description of the RI approximation is provided in Chapter 2 Section 2.4.3.

Herein, only the protein was fragmented amino-acid wise by breaking the bond connecting the alpha carbon and the carbon belonging to the carbonyl group. Fragmented SPE calculations are performed using the many body expansion method (see Section 2.4.5). The MBE method (Eq. (2.46)) is used to compute the gas phase interaction energies.

However, since only the interaction energy (ΔE_{MBE}) of the ligand with the protein is of interest and not the total energy of the full ligand-protein system, a modified version of Eq. (2.46) is used as follows

$$\Delta E_{MBE} = \sum_{I} \Delta E_{\ell I} + \sum_{I<J} \Delta E_{\ell IJ} + \dots \tag{6.7}$$

where ℓ indexes the ligand, I and J denote monomers of the protein target system, $\Delta E_{\ell I}$ and $\Delta E_{\ell I J}$ denote the two-body and three-body corrections to the interaction energy, respectively. This modified version of Eq. (2.46) reduces the computational cost considerably; MBE3 and MBE2 of Eq. (6.7) scales as MBE2 and MBE1 of Eq. (2.46), respectively. Herein, MBE calculations are performed at both the MBE2 and MBE3 levels according to Eq. (6.7).

In all MBE2 calculations unless stated otherwise, dimers were assembled using the full set of protein residues, none were excluded. Contrastingly, in all MBE3 calculations applied to the Wang dataset, only trimers comprising pocket residues were assembled. MBE3 computations on the Gupta dataset involved the full set of protein residues as these systems are significantly smaller (less than 500 atoms).

6.2.5 Solvation Energies

The use of two different types of continuum solvation models was explored, the Poisson-Boltzmann surface area model, and the semi-empirical quantum mechanical conductor-like screening model 2 (COSMO2) at the PM6-D3H4X level.[297] All PBSA and COSMO2 calculations were performed on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center of the United States Department of Energy.

Poisson-Boltzmann Surface Area Calculations

All Poisson Boltzmann-Boltzmann Surface Area calculations were performed with EXESS, specifically, using the GPU-accelerated PBE solver presented in Chapter 5. The same parameters (*i.e.* temperature, grid spacing, solvent probe, *etc.*) employed in Chapter 5 are also utilised herein with the exception of the solute dielectric constant. In this study, the use of three different solute dielectric constants (ε_{solute}) is explored, specifically, 1, 4 and 6. It is known that the accuracy of PBSA is sensitive to the choice of ε_{solute},[298] with the optimal ε_{solute} dependent on the characteristics of the protein-ligand system being studied.[262] Therefore, the use of $\varepsilon_{solute} = 4$ and 6 is examined in addition to the default value of $\varepsilon_{solute} = 1$ typically used in PBSA-based workflows.

COSMO2 calculations

In addition to PBSA, the COSMO2[297] model is also employed for the computation of solvation energies. COSMO2 is a re-parameterisation of the COSMO model[299] and includes a non-polar term that scales according to the solvent accessible surface area (SASA).[297] The reasons for the selection of COSMO2 solvation model were twofold. First, the performance of PBSA is highly dependent on the choice of a user provided input ε_{solute}, as is demonstrated later in Section 6.3.2. Thus, an alternative continuum

solvation model was sought that exhibited no dependence on ε_{solute} as an input parameter. Second, the SQM PM6-D3H4X/COSMO2 approach has been previously applied to a range of protein-ligand systems, demonstrating high correlations with experimental data.[283,300]

COSMO2 solvation energies were computed by coupling it to a single point energy calculation performed at the PM6-D3H4X level of theory.[301–303] Such calculations were performed using the Cuby4 interface[304,305] of MOPAC software package[306] with the MOZYME algorithm.[307] The most recent parameters for halogen-bond corrections for PM6 were utilised where relevant.[308] The solvation energy is computed according to the following equation

$$\Delta G_{solv} = \Delta G_{solv}^{COSMO2} - \Delta G_{solv}^{0}, \quad \text{where } \Delta G_{solv} = G_{solv}^{PL} - G_{solv}^{P} - G_{solv}^{L} \tag{6.8}$$

Here, PL, P and L denote protein-ligand complex, protein, and ligand structures, respectively. Two types of calculations are performed: one in the presence of solvent (G_{solv}^{COSMO2}) and one in vacuum (G_{solv}^{0}).

Due to the computational cost of this more sophisticated solvation model, the protein structures were truncated, containing only the binding pocket residues and also limited to the structures sampled according to the protocol given in Section 6.2.3.

6.2.6 QM/PBSA workflow

The proposed QM/PBSA workflow is summarised in Fig. 6.1. The workflow begins with a starting structure of the bound protein-ligand complex as an input. Following, an MD simulation is performed on the latter according to the protocol as described in Section 6.2.3. Next, all solvent and counter ions are removed from the trajectory, leaving only the trajectory of the complex from which unbound protein and unbound ligand structures are extracted from. Solvation energy calculations are performed using COSMO2 or PBSA for each of the unbound protein, unbound ligand, and bound protein-ligand structures to attain the corresponding solvation energy (see Section 6.2.5). Concurrently, single point energy calculations are performed on selected frames (sampling procedure described in Section 6.2.3) using a fragmented MBE-based approach detailed in Section 6.2.4 to attain $\langle \Delta E_{MBE} \rangle$. Finally, the gas phase interaction energy and solvation energies are combined to give the binding affinity $\Delta G_{bind}^{QM/PBSA}$.

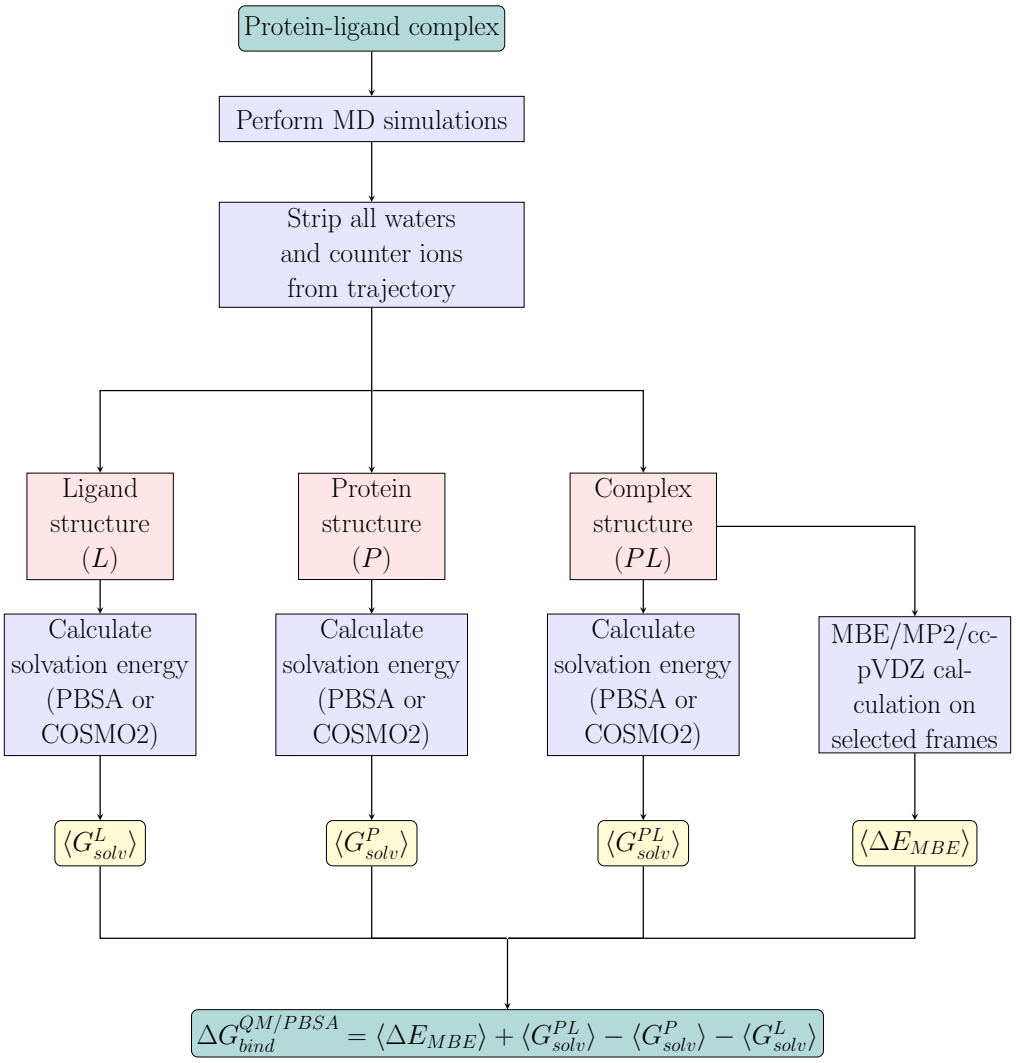


Figure 6.1: The proposed QM/PBSA workflow employed for each protein-ligand system herein.

6.3 Results and discussion

6.3.1 Quantum Chemical Treatment Validation

In this section, the fragmented quantum chemical treatment to compute the gas phase interaction energies is validated.

Table 6.5 lists the correlations attained using different fragmentation schemes and basis sets on the Gupta dataset. A few salient observations can be made here. First, all quantum chemical treatments yielded high correlations with experimental results; the minimum Pearson coefficient observed is 0.93 and 0.78 for the CDK2 and ITK systems, respectively. The lower correlation observed for the ITK systems can be attributed to the lack of available crystal structures used to generate the corresponding protein-ligand structures; crystal structures exist for all 13 complexes in the CDK2 series whereas the

14 complexes of the ITK series were generated from a single crystal structure.[274] Nevertheless, high correlations of ≥ 0.78 are attained which are also consistent with previous studies on the same CDK2 and ITK series,[5,7,274,278] highlighting the importance of the interaction energy contribution in predicting binding affinities.

Fragmentation	CDK2[a]		ITK[b]	
	cc-pVDZ	cc-pVTZ	cc-pVDZ	cc-pVTZ
HF(full)/MP2(MBE2)	0.94	0.96	0.82	0.80
HF(full)/MP2(MBE3)	0.94	0.96	0.82	0.80
HF(MBE2)/MP2(MBE2)	0.93	0.95	0.79	0.78
HF(MBE3)/MP2(MBE3)	0.95	0.96	0.83	0.81

[a]Experimental data taken from Ref. [309, 310]. [b]Experimental data taken from Ref. [278, 311].

Table 6.5: Pearson correlation of gas phase interaction energy with experimental binding free energies using various fragmentation approaches and basis sets on the Gupta dataset.

Additionally, use of fragmentation for the HF component led to no deterioration of R; the correlation always lies between the narrow range of 0.93 to 0.96 for CDK2 and 0.78 to 0.83 for ITK. Furthermore, higher fragmentation levels (MBE3) yields either no effect or increases the correlation slightly. Specifically, there is no change in R when comparing HF(full)/MP2(MBE2) and HF(full)/MP2(MBE3) for both targets; the correlation remains steady at 0.94(0.96) and 0.82(0.80) with the cc-pVDZ(TZ) basis for the CDK2 and ITK series, respectively. However, a marginal rise in the correlation is observed when comparing HF(MBE2)/MP2(MBE2) with HF(MBE3)/MP2(MBE3) for both targets, with increases varying between 0.01 and 0.04. Such results highlight that at least for these systems, whose structures have been truncated to include only nearby residues within 5 Å of each ligand,[7] both MBE2 and MBE3 yield similar correlations to each other. In the following Section 6.3.2, the application of MBE3 to a larger pocket "shell" on the Wang dataset is investigated, comprising protein targets significantly larger than those belonging to the Gupta dataset. Nevertheless, the outcomes here validate the fragmentation approach, with MBE2 and MBE3 both achieving similar correlations as that of the non-fragmented approach.

Besides fragmentation, the influence of basis sets is also considered. Similar correlations are attained across both double-ζ and triple-ζ basis sets. Use of a larger basis set resulted only in marginal increases in R for CDK2 (maximum increase of 0.02), whilst slight reductions in R by up to 0.02 were observed with ITK. Though large basis sets are known to be important for the description of protein-ligand binding, the results herein suggest that in the context of correlation, smaller double-ζ bases are sufficient in describing the electronic effects paired with MP2.

Thus, in this section, the use of the MBE/MP2/cc-pVDZ approach is validated, forgoing the more computationally expensive triple-ζ basis set. Having validated this

quantum chemical treatment, in the next section the MBE/MP2/cc-pVDZ approach is applied to the Wang dataset which comprises eight protein targets comprising between 2,000 and 9,000 atoms.

6.3.2 Binding Affinity Predictions

In this Section, the QM/PBSA workflow is applied to the Wang datasets. To begin, the gas phase interaction energies are computed and the resulting correlations are compared between the three variants of the Wang dataset. Following, solvation effects are incorporated to examine its influence on the prediction of binding affinities. Finally, the performance of the proposed QM/PBSA workflow is compared to other computational techniques.

Interaction Energy

This Section begins by comparing the MBE2 gas phase interaction energies across the three variants of the Wang dataset to assemble the "best" set of starting structures. All MBE2 computations utilise the full structure of the protein, namely, all possible dimers were assembled (no cut off was employed).

Target	Dataset I	Dataset II	Dataset III	Best
BACE	0.22	0.07	0.43	0.43
CDK2	0.52	0.75	0.71	0.75
JNK1	0.66	0.60	0.29	0.66
MCL1	0.35	0.48	0.40	0.48
P38	0.42	0.49	0.62	0.62
PTP1B	0.26	0.45	0.52	0.52
thrombin	0.24	0.25	0.55	0.55
TYK2	0.60	0.60	0.65	0.65
Average	0.41	0.46	0.52	0.58

Table 6.6: Pearson correlation coefficient of MBE2 gas phase interaction energies with experimental binding free energies on the three Wang datasets. Best correlation for each target are shaded in gray.

Table 6.6 lists the correlations attained with the MBE2 gas phase interaction energy across the three Wang datasets. Noticeably, the correlation varies significantly with the starting structure for the majority of the targets. For instance, CDK2 and PTP1B exhibit a 0.23 and 0.19 decrease in the correlation, respectively, when the initial structures are obtained from Dataset I compared to Dataset II. Between Datasets I and II, the latter consistently displays higher correlations; the average Pearson coefficients of 0.41 and 0.46 are attained for Datasets I and II, respectively.

To examine the importance of protonation, Dataset III was also included whose starting structures differ to that of I only by the protonation states of the residues. The improvements in the correlation simply from a change in the protonation procedure are striking. With the exception of one target (JNK1), all targets experience an improvement in their correlation with the alternative protonation procedure; the average correlations increased from 0.41 to 0.52 (see Table 6.6). This is most prominently illustrated through the thrombin and PTP1B targets where an improvement in the correlation by 0.31 and 0.26 were observed, respectively. Geometric structures containing the varying protonation states for all protein-ligand systems are provided in Appendix D.

Across all three datasets, Dataset III comprises the most target structures (five) with the highest correlation whereas Dataset I comprises the least (one). These results together with those comparing Datasets I and II underscore the importance of the initial structure towards the accuracy of predicting binding affinities with QM methods; though quantum chemical methods offer significantly higher accuracy compared to approaches like those employed in MM/PBSA, their effective application also depends on the availability of high quality initial structures. Given the importance of the starting structure, there is an unmistakable need for a robust protein preparation protocol to fully exploit the accuracy of QM methods. However, this lies beyond the scope of the current study, which seeks to evaluate the performance of QM and QM/PBSA on existing structures that have previously served as a benchmark dataset for other studies.

To provide a better assessment of the performance of the QM/PBSA workflow, a new dataset is assembled comprising structures for each target from Datasets I-III that displayed the highest correlation based on their MBE2 gas phase interaction energies (see last column of Table 6.6). All mentions of the Wang dataset henceforth refer to this "best" collection of structures.

Next, the influence of restricting the two-body calculations to the pocket residues is examined, as the results reported thus far include dimers assembled using the full set of protein residues/fragments. The corresponding Pearson correlation coefficients are listed in Table 6.7. Notably, truncating the protein system to the pocket residues virtually yields no change in R_{av}, which remains steady at 0.58, highlight the significant role of pocket residues at the MBE2 level in the correlations attained. However, it should be noted that the R values listed in Table 6.7 are to two decimal places only. The choice to report R values to two decimal places herein is to remain consistent with other studies and to enable easier comparison. When correlations are considered to three decimal places, a slight deterioration of the average correlations from 0.582 to 0.578 is observed when restricting the MBE2 calculations to pocket residues (see Appendix D Table D.2). Taken together with that fact that MBE2 calculations for the untruncated structures are relatively inexpensive (see Figure D.1), the MBE2 results obtained for the full, untruncated structure are used for the remainder of this study.

Target	Full	Pocket
BACE	0.43	0.41
CDK2	0.75	0.76
JNK1	0.66	0.65
MCL1	0.48	0.44
P38	0.62	0.61
PTP1B	0.52	0.57
thrombin	0.55	0.57
TYK2	0.65	0.62
Average	0.58	0.58

Table 6.7: Pearson correlation coefficients of MBE2 gas phase interaction energies computed using either the full protein or pocket residues for the two-body calculations with experimental binding free energies on the Wang dataset.

Following, the effect of increasing the fragmentation level to MBE3 is also explored. Although Section 6.3.1 demonstrated minimal differences between the two- and three-body fragmentation levels, the effect of MBE3 on the structures in the Wang dataset was of interest as they are significantly larger, at least an order of magnitude larger (approximately between 2,500 and 9,000 atoms) compared to the Gupta dataset where protein systems were truncated to less than 500 atoms by including only nearby residues within 5 Å of the ligands. Due to the number of trimers scaling quadratically with the number of fragments, the three-body calculations were limited to only the binding pocket residues. It should be noted that the definition of binding pocket used herein (described in Section 6.2.3) is more relaxed than the strict threshold of 5 Å used for the Gupta dataset and protein residues up to 17.8 Å away from the ligand are included (see Appendix D Table D.3) yielding systems of up to 919 atoms. Furthermore, since three-body effects decay faster with distance than two-body effects, and trimers were constructed using pocket residues previously demonstrated (see Table 6.7) to yield almost no difference in R_{av} for MBE2 compared to the full system, residues beyond the pocket shell are not anticipated to contribute significantly at the three-body level. Therefore, only the pocket residues are included when constructing trimers for MBE3 calculations.

Table 6.8 lists the correlations achieved at the different fragmentation levels. Consistent to the results attained in Section 6.3.1, both MBE2 and MBE3 yield similar results on the Wang dataset; the average correlation observed with MBE2 and MBE3 are 0.58 and 0.57, respectively. Half of the targets (JNK1, MCL1, P38 nad PTP1B) saw no difference in the correlation with a change in fragmentation level. Three targets CDK2, thrombin and TYK2 experienced slight improvements in the correlation with increases ranging between 0.01 to 0.03. Conversely, BACE was the only target where increasing the fragmentation level to MBE3 exhibited a noticeable deterioration from 0.43 to 0.32. Upon initial glance, such observation may appear contradictory since as demonstrated

Target	MBE2	MBE3	best
BACE	0.43	0.32	0.43
CDK2	0.75	0.77	0.77
JNK1	0.66	0.66	0.66
MCL1	0.48	0.48	0.48
P38	0.62	0.58	0.62
PTP1B	0.52	0.52	0.52
thrombin	0.55	0.58	0.58
TYK2	0.65	0.66	0.66
Average	0.58	0.57	0.59

Table 6.8: Pearson correlation coefficient of MBE2 and MBE3 gas phase interaction energies with experimental binding free energies on the Wang dataset. Best correlation for each target are shaded in grey.

in Chapter 3 Section 3.4.2 (*e.g.* see Fig. 3.16), MBE3 is an improved quantum chemical treatment over MBE2. However, as the results presented in Table 6.6 demonstrate, QM methods are highly sensitive to the structure employed and utilisation of a more accurate QM treatment does not yield necessarily yield better correlations—in fact, the opposite can occur as in the case of BACE. Such poor correlations of the BACE target are consistent with other studies employing alchemical free energy methods, where BACE typically exhibited the lowest correlation across the Wang dataset.[112,279,281,282] This may be in part due to the mismatch between the pH at which the crystallisation process occurred (7.5),[312] and that of the experimental binding affinity measurements (5.0).[285,312] This hypothesis is further supported by the correlation differences listed in Table 6.6 between Datasets I and III, which underscores the importance of protonation states on binding affinity predictions.

Thus far, only gas phase interaction energies have been considered in the evaluation of binding affinities. It is also important to account for solvation effects for a more accurate description of the latter. In the following section, solvation energies are incorporated with the gas phase energies employing the optimal fragmentation level—either two- or three-body—identified in this section.

Effect of Solvation

In this Section, the inclusion of solvation effects in the prediction of binding affinities is analysed. Table 6.9 lists the correlations attained from the inclusion of solvation using the PBSA model with different solute dielectric constants. Focusing only on the average correlation, minimal or no improvement in the correlation is observed with the inclusion of the solvation energy. Using solute dielectric constants of 1, 4 and 6 produced average correlations of 0.54, 0.59 and 0.60, respectively—comparable to the gas phase interaction energy correlation of 0.59. However, it should be noted that the performance

of PBSA models is sensitive to the choice of the solute dielectric constant. For instance, the correlation of BACE deteriorates drastically from 0.43 to 0.26 when $\varepsilon_{solute} = 1$ is used compared to 0.42 with $\varepsilon_{solute} = 4$. On the other hand, using $\varepsilon_{solute} = 1$ noticeably improves the correlation for both thrombin and TYK2 to 0.75 and 0.80, respectively, compared to those obtained with $\varepsilon_{solute} = 4$ (0.63 and 0.68) or $\varepsilon_{solute} = 6$ (0.61 and 0.67).

Target	MBE	MBE/PBSA(1)	MBE/PBSA(4)	MBE/PBSA(6)	MBE/PBSA (best ε_{solute})
BACE	0.43	0.26	0.42	0.43	0.43
CDK2	0.77	0.28	0.67	0.77	0.77
JNK1	0.66	0.48	0.65	0.65	0.65
MCL1	0.48	0.57	0.51	0.50	0.57
P38	0.62	0.63	0.63	0.63	0.63
PTP1B	0.52	0.59	0.54	0.55	0.59
thrombin	0.58	0.75	0.63	0.61	0.75
TYK2	0.66	0.80	0.68	0.67	0.80
Average	0.59	0.54	0.59	0.60	0.65

Table 6.9: Pearson correlation coefficients attained with the incorporation of solvation by the PBSA model. Numbers in $(\ldots)$ denote the solute dielectric constant employed. Shaded cells correspond to the best ε_{solute}.

Such inconsistent performance between differing dielectric constants comes as no surprise as previous studies have demonstrated similar outcomes.[262,298,313–315] Specifically, different dielectric constants can overestimate or underestimate the polarisation of the protein-ligand binding interface and the 'optiminal' ε_{solute} is linked to the surface area of the protein exposed to the ligand.[298] Clearly, the varied results with differing dielectric constants observed herein as well as those reported in previous studies not only underscore the importance of the choice of ε_{solute} but highlight a potential need for a protocol of obtaining 'optimal' dielectric constants for PBSA models.

Due to the variability in the performance of PBSA models with dielectric constants, only the results obtained employing the most favourable solute dielectric constant for each target are considered to assess whether the inclusion of solvation *via* the PBSA model generally improves correlation. The corresponding correlations are listed in the last column of Table 6.9. Comparing MBE/PBSA(best ε_{solute}) with MBE where no solvation effects are included, the average correlation improves from 0.59 to 0.65. Most noticeably, this rise in the average R value is attributed to the thrombin and TYK2 targets whose correlations improved by 0.15 and 0.14, respectively.

Given the system-dependent performance of PBSA models, this motivated us to explore alternative implicit solvation models that do not rely on ε_{solute} as a parameter. In

particular, the semi empirical PM6-D3H4X/COSMO2 model is employed to compute the solvation energy.

Target	MBE	MBE/PBSA (best ε_{solute})	MBE/COSMO2	PM6-D3H4X/COSMO2
BACE	0.43	0.43	0.42	0.26
CDK2	0.77	0.77	-0.19	-0.12
JNK1	0.66	0.65	0.32	-0.06
MCL1	0.48	0.57	0.47	0.54
P38	0.62	0.63	0.77	0.21
PTP1B	0.52	0.59	0.33	0.18
thrombin	0.58	0.75	0.57	0.74
TYK2	0.66	0.80	0.75	0.59
Average	0.59	0.65	0.43	0.29

Table 6.10: Pearson correlations attained with varied implicit solvation models PBSA and COSMO2 on the Wang dataset.

Table 6.10 lists the correlations attained with the two different solvation models. With the exception of P38, COSMO2 consistently led to a deterioration in the correlations. This is especially pronounced in the case of CDK2 where the inclusion of the COSMO2 energy with the MBE gas phase interaction energy decreased the correlation from 0.77 to -0.19. The stark decline of the correlation with COSMO2 is quite surprising since the latter is a re-parameterisation of COSMO, which is generally considered a more accurate solvation model than PBSA. Additionally, accurate binding affinities have been previously achieved with this solvation model.[283,300]

This counterintuitive observation where more accurate solvation models yield lower correlations that less accurate ones, has also been reported previously.[67–69] For instance, Yuan *et al.* reported that using a fragmented QM treatment (fragment molecular orbital) with the solvation model density (SMD) or the polarisable continuum model (PCM) resulted in lower correlations than when using COSMO,[67] a technique less accurate than either of PCM or SMD. Therein, it was hypothesised the declined performance was attributed to the poor treatment of the non-polar component, where the inclusion of the latter in PCM had been previously shown to yield lower correlations.[68,316] However, herein, as shown in Appendix D Table D.4, the exclusion of the non-polar term of COSMO2 yielded no improvement in the correlation; rather, it led to either marginal or noticeable declines in R. Specifically, R_{av} decreased from 0.43 to 0.33 when the non-polar term was omitted. Such outcome indicates the issue lies in the polar contribution rather than the non-polar term as was the case in the SMD or PCM solvation model.

We entertained two potential causes for the poor outcome of the COSMO2 results that do not stem from the limitations of the solvation model itself. As described in Section 6.2.5 the COSMO2 computations were performed on a truncated protein system,

potentially introducing errors, unlike that of PBSA calculations which was performed on the full system. However, this hypothesis is not supported by prior findings, as the PM6-D3H4X/COSMO2 approach has been previously shown to be largely insensitive to protein truncation in binding affinity predictions.[300]

Secondly, it should be noted that the COSMO2 solvation energies were obtained by coupling the latter to an SCF calculation performed at the PM6-D3H4X level of theory, compared to the MP2/cc-pVDZ method used for the MBE calculations. Therefore, it may be inappropriate to combine the COSMO2 solvation energies with the gas phase MBE interaction energies instead of those attained at the PM6-D3H4X level. However, binding affinities predicted with the PM6-D3H4X/COSMO2 model yielded considerably worse correlations (see Table 6.10) than either of MBE/COSMO2 or MBE/PBSA; an average correlation of 0.29 was observed. This is in stark contrast to the correlations attained by Jalaie *et al.* using the same PM6-D3H4X/COSMO2 method where an average R of 0.68 was observed on the Wang dataset.[283] However, given the different starting structures used here, such deviations in the correlations observed herein and that by Jalaie *et al.* may be a result of structural differences.

The outcomes presented in this Section underscore the importance of accounting for solvation effects, namely, its ability to improve the correlation as in the case of PBSA, though its performance is sensitive to the ε_{solute} parameter. Furthermore, the results here also echo a similar finding from the previous section (6.3.2) analysing quantum chemical treatments; the use of a more accurate solvation model, such as COSMO2, does not necessarily lead to improved correlations compared to less accurate ones (*e.g.* PBSA) and appear to be highly sensitive to starting structures.

Comparison of QM/PBSA to other methods

The aim of this work is to assess the performance of a proposed QM/PBSA workflow and to examine the influence of quantum chemical treatments and solvation models in binding affinity predictions—rather than to develop a novel, robust end state method for predicting binding affinities. Nevertheless, in this section a comparison of the performance of the proposed QM/PBSA workflow to other computational methods is also provided.

Table 6.11 contains the average correlations attained across four classes of computational methods for binding affinity predictions—alchemical free methods (AFE), conventional scoring functions (SF), machine learning-based scoring functions (ML SF), end point methods—compared to the QM/PBSA approach proposed herein. Noticeably, QM/PBSA consistently yields higher correlations than either of the conventional or ML-based SFs; the average R of the QM/PBSA method is at least 0.13 higher than the SF techniques. Such an outcome is unsurprising given the known low predictability of scoring function methods in general as explained under Chapter 1 Section 1.2. Also

unsurprising given its rigorous formulation is the higher average correlation achieved by AFE methods ($R_{av} = 0.72$) compared to the QM/PBSA approach ($R_{av} = 0.65$).

	[a]AFE	[b]SF	[c]ML SF	[d]End Point	QM/PBSA
BACE	0.57	0.25	0.35	0.27	0.43
CDK2	0.71	0.56	0.48	0.37	0.77
JNK1	0.79	0.30	0.47	0.60	0.65
MCL1	0.71	0.50	0.55	0.58	0.57
P38	0.69	0.51	0.56	0.52	0.63
PTP1B	0.78	0.69	0.61	0.43	0.59
thrombin	0.68	0.69	0.65	0.83	0.75
TYK2	0.84	0.37	0.48	0.71	0.71
Average	0.72	0.48	0.52	0.54	0.65

[a]Data taken from Refs [55, 110, 112, 279–282]. [b]Data taken from Ref [55, 283]. [c]Data taken from [283] .[d]Data taken from Ref [55, 283, 317].

Table 6.11: Pearson correlation coefficients, R, attained on the Wang dataset with various methods. Correlations listed in the AFE, SF, ML SF and End Point columns are taken as the average across multiple studies. References are listed in the footnote.

Also observed in Table 6.11 is the higher correlation of QM/PBSA compared to other end point methods with average correlations of 0.65 and 0.54, respectively. It should be noted that the corresponding correlations listed in Table 6.11 are computed as the average across three end state methods—the MM/GBSA,[55] VeraChem's mining minima (VM2)[317] and SQM2.20[300] approaches (see Appendix D Table D.5). The lower average correlation of 0.54 is primarily due to the negative correlations achieved on the BACE and CDK2 systems with MM/GBSA and also on the PTP1B system with VM2 (see Appendix D Table D.5 for the breakdown). If instead these correlations are set to zero, the average correlation of the other end point methods increases to 0.58, striking a level of accuracy closer to QM/PBSA.

The proposed QM/PBSA method achieves a similar level of accuracy as that of other end point methods which may appear to be computationally cheaper—both SQM2.20 and VM2 exhibits wall times of the order of ten minutes on single-core and multi-core CPUs, respectively[283,317]—compared to QM/PBSA which involves utilisation of GPUs for both the QM and PBSA routines (see Sections 6.2.4 and 6.2.5). However, the PBSA and QM software utilised in this study exhibit extremely high parallel efficiencies and can be scaled up significantly to reduce wall times. Specifically, herein, each MBE calculation was performed across 128 AMD MI250X GPUs and exhibited wall times of the order of minutes whilst a PBSA computation was performed on a single NVIDIA A100 GPU and exhibited wall times of the order of 10 seconds (see Appendix D Figures D.1 and D.2). To run these calculations across all structures in the Wang dataset and across all conformers sampled, the massive parallelism of the OLCF Frontier and NERSC Perlmutter supercomputers was utilised, each comprising thousands of GPUs, to perform

these calculation in parallel, yielding a full QM/PBSA workflow (minus the MD simulation step) across hundreds of structures on the order of minutes. In light of this, the results presented here demonstrate the increasing potential of QM/PBSA workflows at large scales. To reiterate, the aim of this study is not to present a new, robust protocol for accurate prediction of binding affinities but rather to examine the performance of a computationally feasible QM/PBSA workflow across a number of diverse targets and identify its limitations. Clearly, there are many factors highlighted herein (*e.g.* protonation states, quantum chemical treatments and solvation models) critical towards the accurate prediction of binding affinities by QM/PBSA and warrant further study in order to improve the predictive performance of the latter.

6.4 Conclusion

In this Chapter, a highly parallelisable QM/PBSA workflow for binding affinity prediction is presented and its performance on a set of 198 protein-ligand complexes (Wang dataset) spanning eight targets is analysed. The proposed QM/PBSA workflow incorporates a fragmentation-based many body expansion quantum chemical treatment at the MP2/cc-pVDZ level to compute gas phase interaction energies, with the PBSA solvation model for calculating solvation energies.

Firstly, the sensitivity of quantum chemical methods to starting structures in the prediction of accurate binding affinities is highlighted. Specifically, correlations varied significantly when applying the MBE2/MP2/cc-pVDZ method across three versions of the Wang dataset, each exhibiting distinct structural differences (*e.g.* protonation states, ligand placement and protein crystal structures). Furthermore, employing a more accurate quantum chemical treatment by increasing the fragmentation level (from MBE2 to MBE3) does not necessarily correspond to improvements in the correlation, rather, the opposite could be observed. From only gas phase interaction energies (no solvation effects), an average Pearson correlation coefficient of 0.59 could be attained by using the optimal fragmentation levels and initial structures.

Next, solvation effects were incorporated with the PBSA solvation model and the use of three different solute dielectric constants (1, 4 and 6) was explored. Consistent with other studies, the performance of the combined MBE/PBSA model varied with ε_{solute}; some systems exhibited improved correlations with the default $\varepsilon_{solute} = 1$ whereas other targets exhibited deteriorated correlations. When the best performing ε_{solute} for each target is considered, the addition of solvation energies with the PBSA model improved the average correlation coefficient to 0.65. Given the ε_{solute}-dependent performance of PBSA solvation models, the use of a semi-empirical quantum mechanical solvation model was explored, namely, COSMO2 evaluated at the PM6-D3H4X level. However, this led to the noticeable reduction in the correlation for the majority

of the targets, with an average correlation of 0.43 achieved.

Finally, the QM/PBSA method was compared to a range of other computational methods: alchemical free methods, conventional and machine learning-based scoring functions, as well as other end point methods (MM/GBSA, VM2 and SQM2.20). On average, QM/PBSA ($R_{av} = 0.65$) exhibits better performance than either of the conventional ($R_{av} = 0.48$) or machine learning-based scoring functions ($R_{av} = 0.52$). Expectantly, AFE methods achieve higher average correlations ($R_{av} = 0.72$) than QM/PBSA owing to their more rigorous formulations. Compared to three other end point methods ($R_{av} = 0.54$), QM/PBSA exhibits comparable performance. Additionally, the computational costs between these methods are also considered; QM/PBSA makes extensive use of GPUs and its performance can be further scaled by leveraging the massive parallelism of modern supercomputer architectures. The proposed QM/PBSA workflow uses software that exhibits extremely high parallel efficiencies, with the full QM/PBSA workflow (excluding MD simulation step) across hundreds of protein-ligand structures being achieved on the order of minutes when leveraging the massive parallelism of supercomputers.

Taken together, this Chapter advances the application of QM/PBSA methodologies, recognising that their accuracy depends on several factors such as structural preparation protocols, solvation models and quantum chemical methods. It marks an important step towards the broader adoption of QM/PBSA workflows for large scale binding affinity prediction and demonstrates their growing potential and computational feasibility for practical application protein-ligand systems.

Conclusion

This thesis presents computational methods and algorithms developed for the realisation of large scale, accurate, computationally feasible QM/PBSA workflows in predicting binding affinities, focusing on improving efficiency and accuracy. The automated molecular fragmentation, improved initial guesses for SCF, and GPU-accelerated PBE solver provide researchers techniques to study solvated molecular systems at large scales and accuracies.

Chapter 1 highlighted the computational challenges precluding the large scale adoption of existing accurate chemical modelling techniques for protein-ligand binding affinity predictions. Such techniques include quantum mechanical methods such as HF and MP for predicting gas phase interaction energies, as well as PBSA solvation models. The need for computationally efficient QM and PBSA computations was emphasised, motivating the exploration of techniques for their acceleration.

Chapter 2 provides a background description of the core computational and chemical concepts used within this thesis including high performance computing, existing methods used for modelling protein-ligand systems, and quantum mechanical methods.

Chapter 3 presents an automated molecular fragmentation scheme (QFRAGS) suitable for large molecular systems. The proposed scheme aims to locate the optimal set of bonds to break to fragment the system by an evolutionary optimisation of a scoring function that considers a range of factors concerning the fragment size and maintaining the integrity of the chemical environment. QFRAGS was applied to two types of biological systems, namely, proteins and lipoglycans/glycolipids. Combined with the many body expansion approach at the three-body level to estimate energies at the HF/6-31G* theory level, mean absolute energy errors of 24.3, 2.2 and 0.3 kJ mol^{-1} were achieved on the set of large protein, small protein and lipoglycan/glycolipid structures, respectively. Though the molecular fragmentation method was developed as a tool to enable computationally feasible QM gas phase energies for QM/PBSA on protein systems, its excellent performance on other types of structures (*i.e.* lipoglycans/glycolipids) highlight the general applicability of QFRAGS to other biological systems, and is generally applicable to organic systems.

Besides fragmentation, another approach of reducing the computational cost of quantum chemical calculations on large molecular systems involves accelerating convergence by improving the initial guess utilised in self-consistent field calculations to reduce iteration counts. Chapter 4 presents a performance analysis of two existing initial guess schemes—many body expansion and basis set projection—as well as a newly proposed scheme that combines the two against the traditional superposition of atomic densities. Across varying theory levels (HF, B3LYP and MN15) and system types (proteins, nucleic acids, carbohydrates), the results consistently demonstrate the lower iteration counts and wall times of non-SAD techniques compared to SAD, with higher speedups being observed for larger systems. Such observations highlight that more sophisticated non-SAD techniques are of greater value for larger systems. Though the focus of this thesis is predominantly protein systems, the merit of these initial guess techniques has also been demonstrated for several class of systems beyond proteins, underscoring the general applicability of these schemes.

In addition to computing gas phase energies, another key component of QM/PBSA workflows are solvation models. Chapter 5 presents a new GPU-accelerated linear PBE solver and its performance is evaluated on a set of 335 protein-ligand systems ranging between 2,500 and 10,000 atoms, with comparisons made against two popular PBE solvers: DelPhi (CPU-based) and AMBER's *pbsa.cuda* (GPU-based). The results demonstrate the significant speedups of the proposed PBE solver over both DelPhi and *pbsa.cuda*, with respective average speedups of 7.3× and $8.5\times$ observed. The availability of this faster PBE solver, which can be scaled by leveraging the parallelism of modern supercomputers, contributes towards the realisation of large scale QM/PBSA workflows.

With techniques developed to accelerate QM and PBSA computations, a QM/PBSA workflow is presented in Chapter 6 and its performance is analysed on a diverse set of 198 protein-ligand complexes spanning eight targets. The proposed workflow utilises a fragmentation-based many body expansion quantum chemical treatment of gas phase interaction energies at the MP2/cc-pVDZ level and a PBSA model for solvation. GPU-accelerated software were employed for both types of calculations and can be scaled up to exploit the parallel architecture of supercomputers, yielding a GPU-accelerated QM/PBSA workflow. Applied to the 198 protein-ligand dataset, an average correlation of $R_{av} = 0.65$ was attained. In addition, it was observed that the performance of the latter was highly sensitive to starting structures, fragmentation level and solvation models. Such outcomes underscore the importance for further study in these areas in order to improve performance of computational methods in predicting binding affinities. Additionally, the performance of the QM/PBSA method was also compared to a range of other computational approaches: scoring functions ($R_{av} = 0.50$), alchemical free energies ($R_{av} = 0.72$), and other end point methods ($R_{av} = 0.54$).

Altogether, this thesis presents several new techniques—including molecular frag-

mentation, improved initial guesses for SCF and faster linear PBE solvers—towards the realisation of computationally feasible QM/PBSA workflows at large scales, and provides a comprehensive benchmark analysis of a QM/PBSA workflow using these methods. The availability of these tools used to assemble a QM/PBSA workflow represents a significant step forward in accelerating drug discovery, capitalising on the parallel architecture of modern HPC machines.

Though this thesis largely focuses on protein systems, the computational methods and algorithms presented herein, can largely be extended to molecular systems in general, broadening the range of systems and scientific fields in which such approaches can be effectively employed. With the continued advancement of high performance computing, the availability and development of such techniques become increasingly important, requiring the adaption of methodologies and algorithms to changing supercomputing architecture.

7.1 Future Work and Outlook

The QM/PBSA workflow presented herein demonstrates the growing potential and computational feasibility of its application at large scales. As highlighted in Chapter 6, the performance of this technique is highly sensitive to starting structures, quantum chemical models and solvation models. Thus, in order to exploit the accuracy of QM/PBSA, the relationship between these factors needs to be well understood and optimised. The importance of these factors is not unique to QM/PBSA only. For example, the performance of other end point techniques in addition to QM/PBSA (*e.g.* VM2[317] and SQM2.20[283]) have been shown to be highly dependent on starting structures. The structure of proteins can vary significantly even for the same protein target depending on the ligand it is bound to, as well as its protonation states and more. Therefore, there is a general need for a robust protein preparation protocol that accounts for various factors including protonation states at a given pH, ligand placement, elimination of steric clashes *etc*. The availability of this tool would not only be of enormous benefit to the drug discovery research community, but also in the fields of computational biology and chemistry, considerably reducing the amount of time spent preparing protein structures.

Other avenues worth pursuing are concerned with solvation models. Specifically, the optimisation of solute dielectric constants for the PBSA solvation model. As demonstrated in Chapter 6, the performance of QM/PBSA is highly dependent on the dielectric constant selected for the solute. Whilst there has been some attempt to characterise the relationship between protein-ligand structures and the 'optimal' ε_{solute},[298] there is an overall lack of the explicit pursuit of optimising ε_{solute} to exploit the benefit of the PBSA model. In addition to ε_{solute}, the performance of PBSA is also dependent on other factors

including protein partial charges which are typically derived from force fields. Furthermore, there may also be issues associated with the separate treatment the two different energetic terms (gas phase interaction energy and solvation energy) in QM/PBSA. Therefore, it would also be worthwhile in shifting towards QM continuum solvation models in which the latter is coupled to a quantum chemical calculation. This approach not only enables simultaneous evaluation of gas phase and solvation energies, but should also provide more accurate energies: the gas phase energy is computed in the presence of a continuum field, and the evaluation of the solvation component employs partial charges derived from the QM calculation. Additionally, the application of molecular fragmentation (Chapter 3) has only previously been applied to QM calculations *in vacuo* but could easily be extended to such QM computations coupled to a continuum solvation model (*e.g.* PCM, SMD, COSMO2), reducing their high computational costs.

Moreover, though the GPU-accelerated linear PBE solver presented in Chapter 5 has only been applied to water-solvated systems in this thesis, it can easily be extended to other non-water solvents (*e.g.* ethanol, hexane, ethyl acetate or dimethyl sulfoxide) to study systems beyond the context of drug discovery at large scales. This is particularly important as modern computing systems shift toward greater reliance on parallel architectures (*i.e.* GPUs).

Furthermore, there are also limitations associated with the use of force fields for sampling of the bound protein-ligand conformations for QM/PBSA. The shortcomings of a molecular mechanical treatment of molecular systems has been well discussed in this thesis, particularly, its inability to accurately describe electronic effects important in binding. Therefore, the conformations generated from this sampling procedure may not be reflective of the protein-ligand structures in nature. In addition, there are also concerns associated with the different energy functions utilised to generate the conformations (AMBER ff03 force field) and compute gas phase energies (MP2/cc-pVDZ) in the current QM/PBSA workflow presented herein. Instead, another worthwhile direction to pursue involves the realisation of computationally feasible QM-based molecular dynamics (*i.e. ab initio* molecular dynamics - AIMD) simulations to perform the sampling routine. For example, the molecular fragmentation and initial guess approaches presented in Chapters 3 and 4, respectively, can be extended to AIMD applications to reduce its steep computational overhead.Given the sensitivity of end state methods such as QM/PBSA to the geometry of molecular systems, the availability of computationally feasible AIMD workflows would theoretically generate binding poses more reflective of nature and enable researchers to more accurately assess the performance of computational methods.

Thus far, future developments or suggestions have been limited to the area of computational methods. However, there is also a need for a high quality dataset for perfor-

mance evaluation of computational methods in predicting binding affinities. Problems exist in established datasets commonly used to assess computational approaches. For instance, the Wang dataset employed in Chapter 6 to assess the performance of the proposed QM/PBSA workflow on includes at least two problematic targets. Specifically, the crystallographic structure used for BACE is obtained at a different (7.5) than that at which experimental binding affinities were measured (pH of 5.0).[285] Additionally, the JNK1 crystallographic structure was obtained at a relatively low resolution (3.5 Å)[287] compared to the higher resolutions of the remaining targets (all less than 2.5 Å). The Wang dataset is arguably the most studied benchmark dataset for evaluating the performance of computational methods in predicting binding affinities. Therefore, access to a high quality dataset with accompanying experimental data is of critical importance in order to accurately assess various computational approaches in a standardised manner, allowing researchers to make more informed decisions about which techniques to use.

Overall, with advancements in high performance computing and computational chemistry algorithms, including those presented in this thesis, the practical application of QM/PBSA methods for large scale virtual screening is becoming increasingly feasible, with the potential to substantially accelerate drug development and discovery. However, several concerns remain that hinders its widespread adoption, as discussed previously. Significant research is still required to fully exploit the benefits of these computational methods.

Appendix A

Supporting Material for Chapter 3

The geometries of the structures in the datasets, geometries of the fragments, full system energies and MBE2/3 energies are available in the GitHub repository `https://github.com/fionacyu/fragPaper`.

Reference points for initial guess

The approach for computing reference points involves dividing the space occupied by the molecular system into three-dimensional rectangular intervals, and the midpoint of each interval is taken as a reference point. The intervals are formed by partitioning in the direction of the three principal axes of inertia. This first involves computing the inertia tensor and diagonalising it. The eigenvectors v_i of the inertia tensor can be arranged in matrix form

$$V = \begin{bmatrix} \vdots & \vdots & \vdots \\ v_1 & v_2 & v_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{A.1}$$

The coordinates of the atoms are projected from the Euclidean space to the Eigenspace by taking the product of V and the matrix containing the Cartesian coordinates of all atoms as follows

$$V \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ v_1 & v_2 & v_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & \dots & x'_n \\ y'_1 & y'_2 & \dots & y'_n \\ z'_1 & z'_2 & \dots & z'_n \end{bmatrix} \tag{A.2}$$

where (x_i, y_i, z_i) and (x'_i, y'_i, z'_i) denote the coordinates of atom i in the Euclidean and Eigenspace, respectively.

In the transformed coordinate system, the range (Δt) along the direction of the three eigenvectors are computed. Δt is given by

$$\Delta t = t_{max} - t_{min} \tag{A.3}$$

where t_{max} and t_{min} denote the maximum and minimum coordinate values in the Eigenspace along the t-direction. As we have three eigenvectors which form the basis of the Eigenspace, we also have three range values. Of these, we take the minimum range ($\Delta t'_{min}$) to determine the number of intervals (n_{min}) along the direction of the eigenvector that corresponds to $\Delta t'_{min}$. n_{min} is calculated as

$$n_{min} = \left\lceil \frac{\Delta t'_{min}}{\Delta t^*} \right\rceil \tag{A.4}$$

where Δt^* is a hyperparameter and is given the default length of 15 Å. For the other two directions, the number of intervals (n_{int}) is dependent on the value of n_{min}.

$$n_{int} = n_{min} \left\lceil \frac{\Delta t}{\Delta t'_{min}} \right\rceil \tag{A.5}$$

Furthermore, the length of each interval along the t-direction (l_t) can be computed according to:

$$l_t = \frac{\Delta t}{n_{int}} \tag{A.6}$$

For convenience let us denote the set of axes corresponding to the basis vectors (eigenvectors of the inertia tensor) as x', y' and z' axes. The number of intervals along the direction of each of the three basis vectors is $n_{x'}$, $n_{y'}$, $n_{z'}$, and the corresponding interval lengths are $l_{x'}$, $l_{y'}$ and $l_{z'}$, respectively. The total number of intervals (n_{tot}) is given by the following product

$$n_{tot} = n_{x'} n_{y'} n_{z'} \tag{A.7}$$

The corresponding reference points (in the Eigenspace) is calculated according to the following formula

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x'_{min} \\ y'_{min} \\ z'_{min} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} l_{x'} \\ l_{y'} \\ l_{z'} \end{bmatrix} + \begin{bmatrix} il_{x'} \\ jl_{y'} \\ kl_{z'} \end{bmatrix} \tag{A.8}$$

where i, j and k are integer counters along the x', y' and z' directions, respectively, and these take on values between 0 and their corresponding n_{int}.

The reference points calculated with Eq. (A.8) are located in the Eigenspace. To transform these back to the Euclidean space, we multiply the inverse of V with each of the reference points calculated with Eq. (A.8). Since V is orthonormal, its inverse is

equivalent to its transform. Thus, instead of computing the inverse which is computationally expensive, the reference points in the Euclidean space can be accessed by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = V^T \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \dots & v_1 & \dots \\ \dots & v_2 & \dots \\ \dots & v_3 & \dots \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{A.9}$$

Appendix B

Supporting Material for Chapter 4

Geometries of the structures in the data set, geometries of fragments with varying fragment sizes employed for MBE are available in the GitHub repository `https://github.com/fionacyu/initial_guess_paper`.

Methodology

SAD guess

Table B.1: Element and number of unpaired electrons used to generate SAD guesses. Only elements included in dataset are specified.

Element	Unpaired Electrons
H	1
B	1
C	2
N	3
O	2
F	1
P	3
S	2

Fragmentation Schemes

The fragmentation algorithm for covalently bonded systems is detailed in Algorithm B.1. The core concept of this algorithm is that fragmentation is primarily driven by a target fragment size and only severs single bonds.

Algorithm B.1 starts by sorting atoms in ascending order based on their distance from the center of nuclear charge of the molecular system. This ensures that the frag-

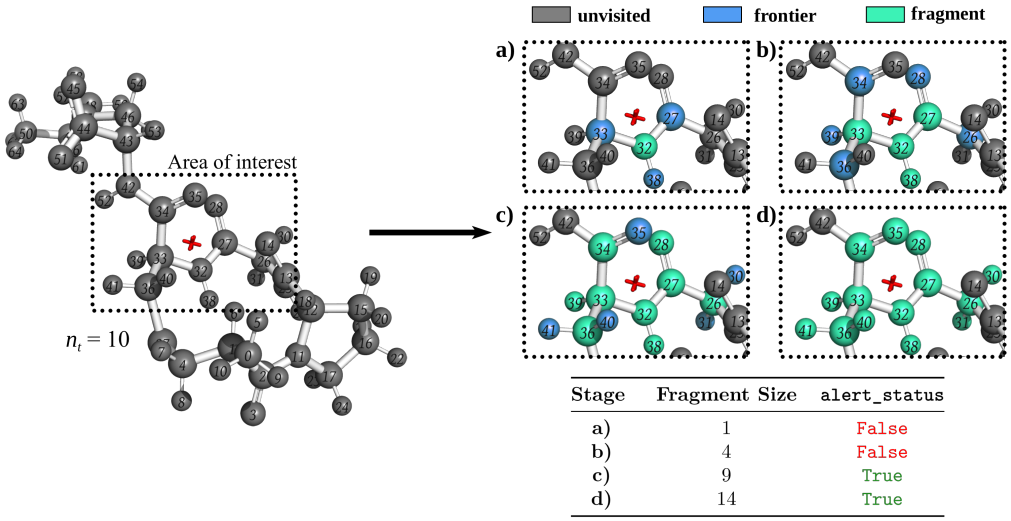
Algorithm B.1 Fragmentation scheme for covalently bonded systems.

Require: Molecular System, n_t
1: Sort atoms by distance to centre of charge in ascending order
2: Initialise zero vector `atom_frag_labels`
3: frag_index $\Leftarrow 0$
4: **for** each atom i **do**
5: **if** atom $i \neq$ visited **then**
6: $N_a \Leftarrow 1$
7: `frag_index` $\Leftarrow$ `frag_index` + 1
8: `atom_frag_labels`$[i] \Leftarrow$ `frag_index`
9: Initialise empty queue container Q
10: Enqueue atom i to Q
11: **while** $Q \neq$ empty **do**
12: atom $j \Leftarrow$ dequeue Q
13: **for** each unvisited neighbor atom k of atom j **do**
14: `alert_status` $\Leftarrow N_a \geq 0.85 \cdot n_t$
15: **if** `alert_status` **then**
16: $N_a \Leftarrow N_a + 1$
17: **if** bond order between j and $k > 1$ **then**
18: Enqueue atom k to Q
19: `atom_frag_labels`$[k] \Leftarrow$ `frag_index`
20: **else if** degree of atom $j == 1$ **then**
21: Enqueue atom k to Q
22: `atom_frag_labels`$[k] \Leftarrow$ `frag_index`
23: **end if**
24: **else**
25: Enqueue atom k to Q
26: $N_a \Leftarrow N_a + 1$
27: **end if**
28: **end for**
29: **end while**
30: **end if**
31: **end for**

mentation scheme is invariant with respect to rotations, translations, and reflections in the geometry. A zero vector `atom_frag_labels` is initialised on line 2, containing the fragment index to which each atom is assigned. Each unvisited atom is iterated over, and a breadth-first search (BFS) is initiated to build a fragment. The BFS process, described from lines 11 to 29 in Algorithm B.1, involves sorting neighbors in ascending order by their distance to atom j.

A boolean variable `alert_status` on line 14 and signals the algorithm to begin severing bonds when set to `True`. This occurs when the number of atoms in the growing fragment (N_a) reaches or exceeds 85% of the target fragment size (n_t). At this point, only atoms connected by non-single bonds or atoms with a degree of one are added to the queue container Q, ensuring only single bonds are severed. Degree refers to the number of covalent bonds an atom is part of. The BFS procedure continues until Q is empty.

To illustrate the BFS step, consider Figure B.1 which showcases how a fragment is constructed using Algorithm B.1 with a target fragment size of 10 atoms. In a), the unvisited atom closest to the centre of nuclear charge (atom 32) is added to the growing fragment, and its neighbor atoms 27, 33 and 38 (depicted in blue) are on the frontier of the search. These are added to the queue container Q in Algorithm B.1. Next in b), the atoms previously on the frontier (27, 33 and 38) are added to the growing fragment and their respective neighbors (shown in blue) are enqueued to Q. In c), `alert_status`



Stage	Fragment Size	alert_status
a)	1	False
b)	4	False
c)	9	True
d)	14	True

Figure B.1: The construction of a fragment using Algorithm B.1 on a protein system (PDB ID: 6QM1). Target fragment size (n_t) is 10 atoms. Red cross indicates center of nuclear charge. a) to d) show the evolution of the growing fragment.

is set to True since the size of the growing fragment (9 atoms) exceeds 85% of 10. As atoms 30, 31, 40 and 41 have degree of one and a double bond connects atoms 34 and 35, these five atoms (displayed in blue in c)) are first enqueued to Q before being added to the growing fragment, as displayed in d).

The next fragment is generated beginning from the next unvisited atom closest to the the centre of nuclear charge. Algorithm B.1 continues until all atoms have been visited.

For bonded systems where fragments are produced by severing single bonds, valence is restored by appending hydrogen caps.

Algorithms B.2 and B.3 together describe the fragmentation procedure for non-covalent cluster systems, targeting varying fragment sizes n_t. Algorithm B.2 begins by identifying the unit systems within the molecular system, which are the smallest repeating units representative of the cluster. For instance, a single water molecule in a water cluster system or a single ion pair in an ionic liquid cluster.

Algorithm B.3 determines these unit systems and requires the number of components N_C in a unit system as an input. This number, representing the connected components, is evaluated using a BFS procedure: $N_C = 1$ for water or benzene, and $N_C = 2$ for ionic liquid clusters. The first step in Algorithm B.3 is a BFS procedure to identify disjoint components. If $N_C = 1$, the count of disjoint components equals the unit system count. For $N_C = 2$, the disjoint components are separated into two containers, A and B, based on size (line 5). After separation, container A and B will contain disjoint components of different sizes, with each container having components of the same size.

Next, disjoint components from A and B are paired together. The outer for loop

(line 8) iterates first over the container with components closest to the center of nuclear charge, sorted in ascending order of distance. In Algorithm B.3, this is container A. For each iteration in the outer loop, disjoint components in container B are also sorted by distance to the center of charge of system a. On line 10, component b is paired with a to form a unit system, which is then appended to container S. S is returned after all disjoint components are paired.

Algorithm B.2 Fragmentation scheme for non-covalent cluster systems.

Require: Molecular System, n_t
1: $S \Leftarrow$ Identify unit systems in molecular system (Algorithm B.3)
2: Initialize N_A as number of atoms per unit system
3: $N_{US} \Leftarrow n_t / N_A$
4: Initialize zero vector `unit_system_frag_label`
5: `frag_index` $\Leftarrow 0$
6: **for** each unvisited unit system i in S **do**
7: `frag_index` $\Leftarrow$ `frag_index` $+ 1$
8: $N_{US}^t \Leftarrow 1$
9: `unit_system_frag_label`$[i] \Leftarrow$ `frag_index`
10: **for** each unvisited unit system $j \neq i$ in S **do**
11: **if** $N_{US}^t == N_{US}$ **then**
12: **break**
13: **end if**
14: `unit_system_frag_label`$[j] \Leftarrow$ `frag_index`
15: $N_{US}^t \Leftarrow N_{US}^t + 1$
16: **end for**
17: **end for**

With the unit systems stored in S, Algorithm B.2 determines the number of unit systems per fragment N_{US} (line 3). A zero vector (`unit_system_frag_label`) is initialised on line 4 to store the fragment index for each unit system. The algorithm iterates over each unvisited unit system i, setting N_{US}^t to 1, representing the temporary count of unit systems in the growing fragment. Unit system i is labeled with the current fragment index (line 9). If N_{US}^t reaches the target N_{US}, the next unvisited unit system in S is processed (line 6). Otherwise, the unit systems in S are sorted by distance to the center of charge of unit system i (line 10). Unvisited unit systems j encountered are appended to the growing fragment, and labeled with the current fragment index (line 14). Once N_{US}^t reaches N_{US}, the next unvisited unit system is processed (line 6). Finally, with all unit systems labeled, corresponding fragments are extracted for MBE.

Algorithm B.3 Identifying unit systems for fragmentation of non-covalent cluster systems.

Require: N_C
Perform breadth first search to determine all disjoint components
if $N_C == 1$ **then**
    return disjoint components
else if $N_C == 2$ **then**
    Initialise empty disjointA system containers A and B
    Separate disjoint components to containers A and B
    Initialise empty unit system container S
    for disjoint system a in A **do**
        for unvisited disjoint system b in B **do**
            pair a and b forming a unit system, append to S
        end for
    end for
    return S
end if

Results

Basis Set Projection

Bootstrapping Basis Set

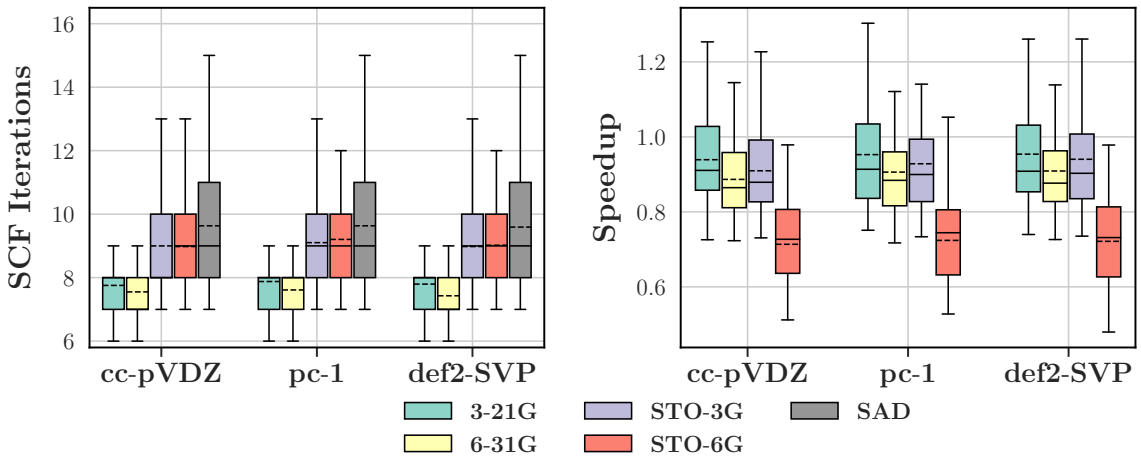


Figure B.2: Distribution of SCF iterations and wall-time speedups of double-ζ basis set BSP calculations with various bootstrapping basis sets for HF. Average is indicated by broken horizontal line.

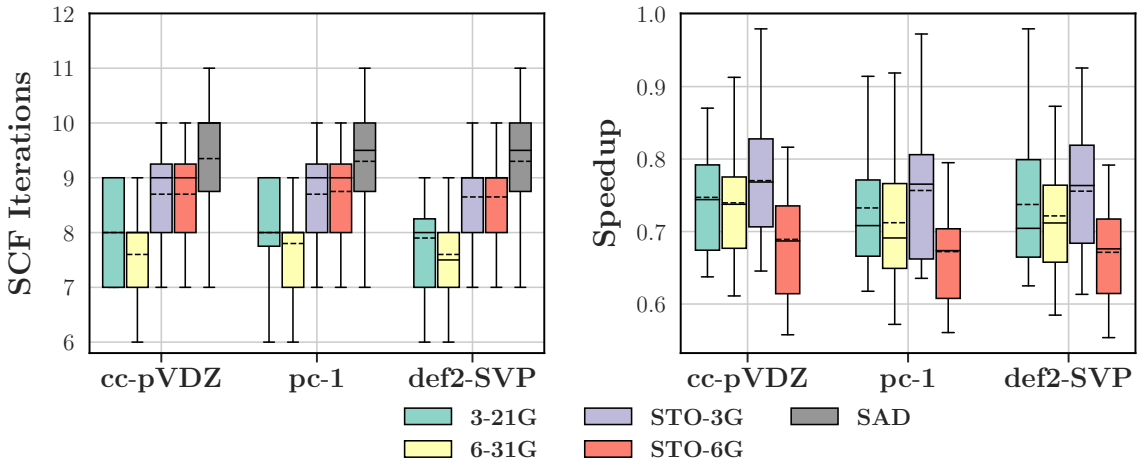


Figure B.3: Distribution of SCF iterations and wall-time speedups of double-ζ basis set BSP calculations with various bootstrapping basis sets for B3LYP. Average is indicated by broken horizontal line.

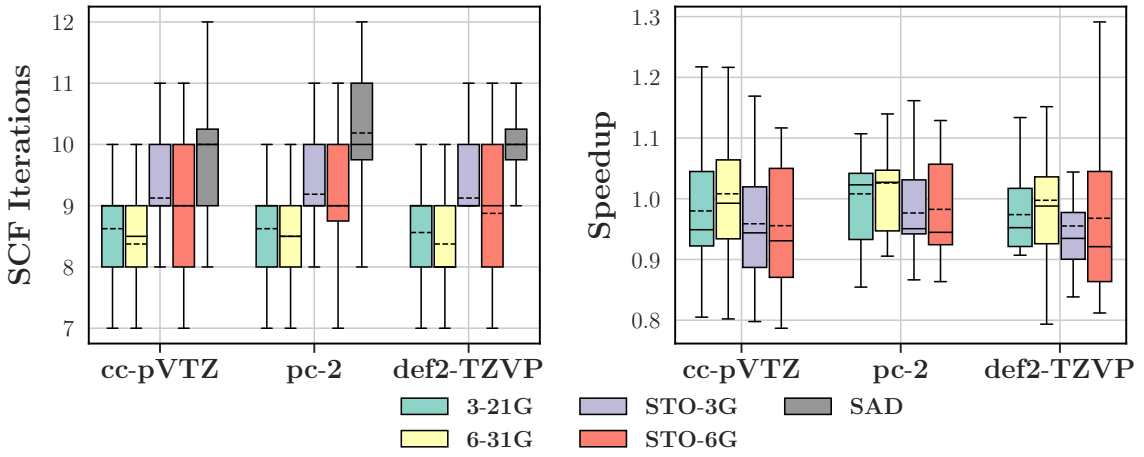


Figure B.4: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set BSP calculations with various bootstrapping basis sets for B3LYP. Average is indicated by broken horizontal line.

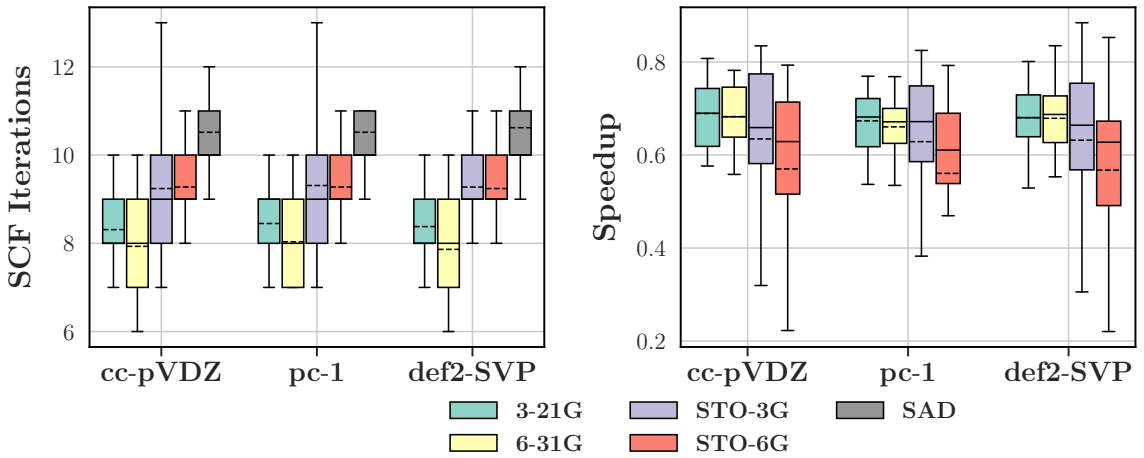


Figure B.5: Distribution of SCF iterations and wall-time speedups of double-ζ basis set BSP calculations with various bootstrapping basis sets for MN15. Average is indicated by broken horizontal line.

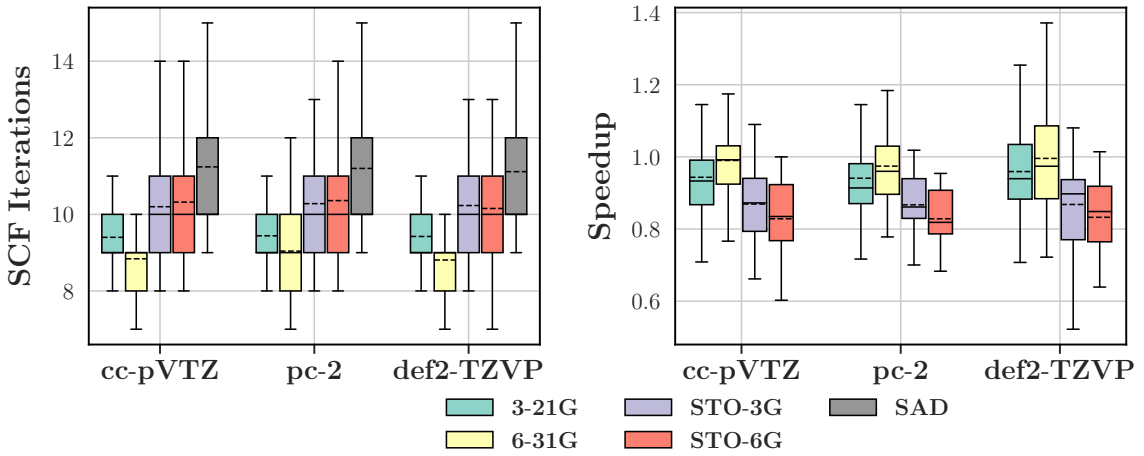


Figure B.6: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set BSP calculations with various bootstrapping basis sets for MN15. Average is indicated by broken horizontal line.

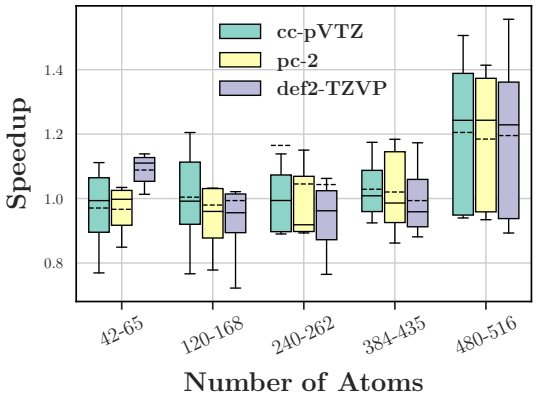


Figure B.7: Distribution of wall-time speedups of triple-ζ BSP calculations for increasingly large systems employing the 6-31G bootstrapping basis set at the MN15 level of theory. Average is indicated by horizontal dashed line.

Convergence Threshold

Employing the optimal thresholds listed in Table B.2, Figure B.8 displays the corresponding average wall-time speedups of BSP calculations. Noticeably, the mean speedups for the triple-ζ basis sets are higher than that of double-ζ; the speedups of the triple-ζ BSP computations for HF, B3LYP and MN15 are at least $1.13\times, 1.27\times$ and $1.26\times$ higher than their double-ζ counterparts. More significantly, in the DFT-based computations, the double-ζ BSP calculations on average exhibit worse wall-times than SAD. For instance, utilizing the pc-1 basis set, BSP yielded wall-times that were on average 14.7% and 16.9% longer than that of SAD with the B3LYP and MN15 methods, respectively.

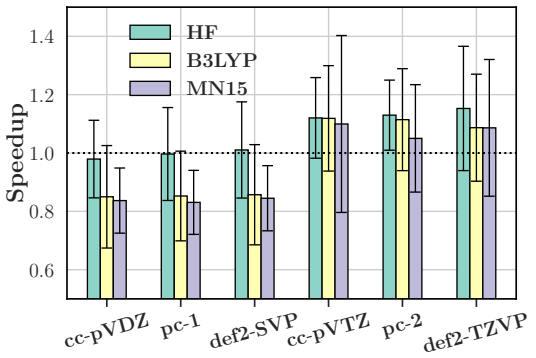


Figure B.8: Average wall-time speedups of BSP calculations for varying basis sets and levels of theory. Convergence thresholds employed are those listed in Table B.2. Error bars correspond to one standard deviation.

Table B.2 also outlines the additional speedup relative to SAD afforded using the optimal convergence thresholds compared to that of 10^{-6} a.u. With the exception of the DFT-based double-ζ BSP calculations which were previously discussed, only marginal

improvements in speedups were observed when utilizing larger convergence thresholds. In particular, the highest improvement in average speedup over 10^{-6} a.u. observed in double- and triple-ζ computations were 5.7% and 4.0%, respectively.

Table B.2: Improvement in speedup relative to SAD of BSP calculations using optimal thresholds compared to 10^{-6} a.u. across various basis sets and levels of theory. The exponents, λ, of convergence thresholds, 10^{λ}, are listed in parentheses.

	HF	**B3LYP**	**MN15**
cc-pVDZ	4.0%(-2)	12.9%(1)	11.4%(-2)
pc-1	4.4%(-2)	14.3%(0)	12.4%(-2)
def2-SVP	5.7%(-1)	15.8%(0)	12.2%(-2)
cc-pVTZ	1.6%(-2)	3.2%(-4)	3.1%(-4)
pc-2	1.2%(-2)	1.8%(-4)	1.9%(-4)
def2-TZVP	4.0%(-2)	2.9%(-4)	2.9%(-4)

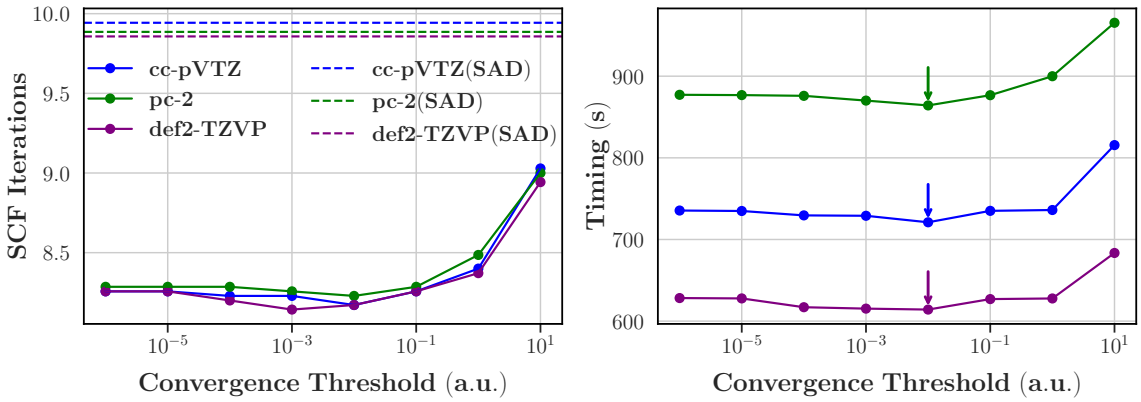


Figure B.9: Average number of SCF iterations and wall-times of triple-ζ basis set BSP calculations with varying fragment convergence thresholds at the HF level.

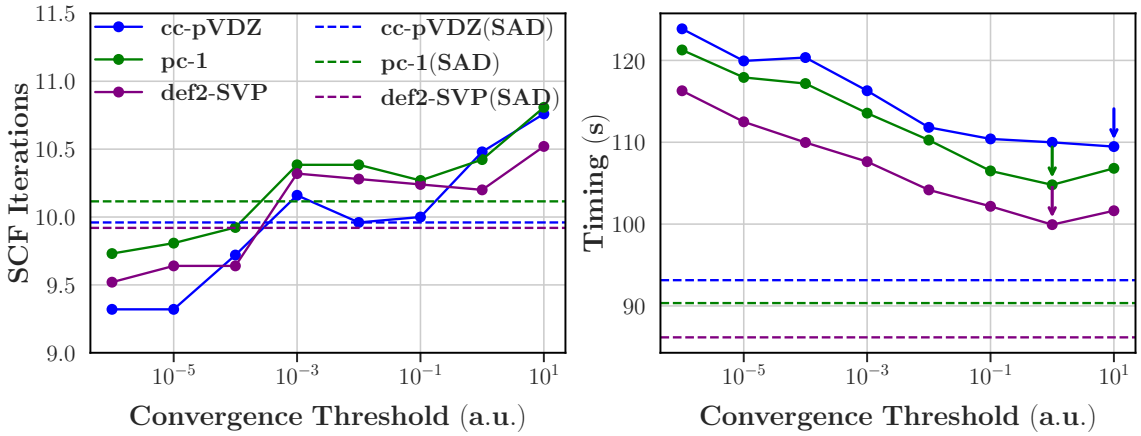


Figure B.10: Average number of SCF iterations and wall-times of double-ζ basis set BSP calculations with varying fragment convergence thresholds at the B3LYP level. Arrows indicate the convergence threshold with the lowest average wall-time.

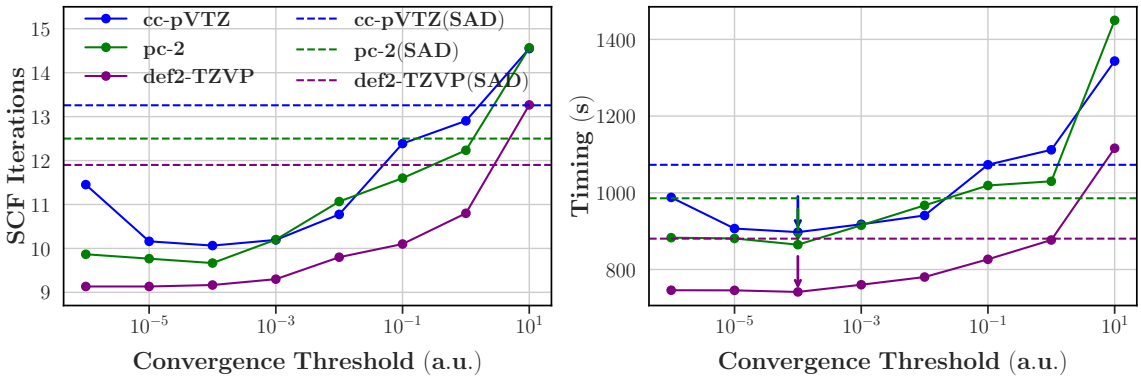


Figure B.11: Average number of SCF iterations and wall-times of triple-ζ basis set BSP calculations with varying fragment convergence thresholds at the B3LYP level. Arrows indicate the convergence threshold with the lowest average wall-time.

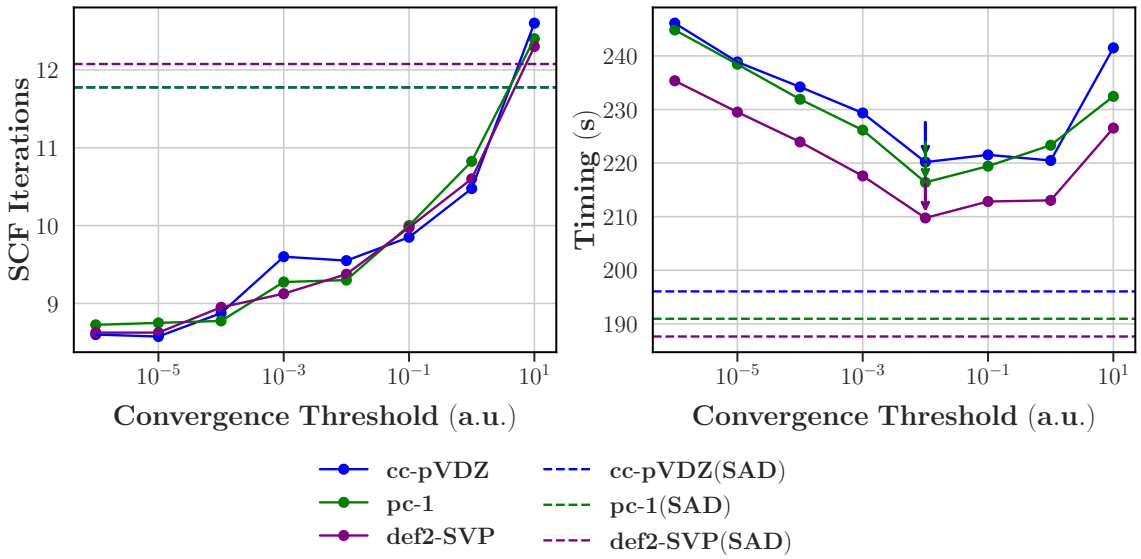


Figure B.12: Average number of SCF iterations and wall-times of double-ζ basis set BSP calculations with varying fragment convergence thresholds at the MN15 level. Arrows indicate the convergence threshold with the lowest average wall-time.

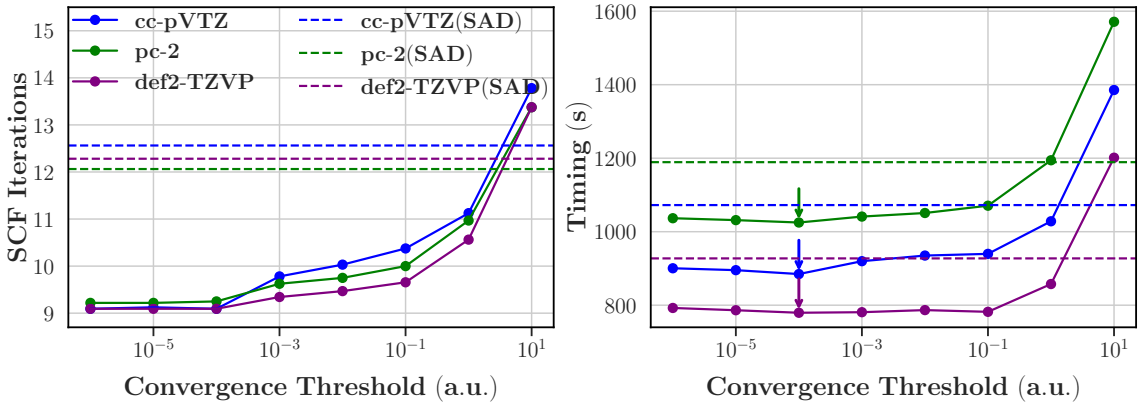


Figure B.13: Average number of SCF iterations and wall-times of triple-ζ basis set BSP calculations with varying fragment convergence thresholds at the MN15 level. Arrows indicate the convergence threshold with the lowest average wall-time.

Many Body Expansion

Fragment Size

	HF	B3LYP	MN15
cc-pVDZ	50	40	60
pc-1	60	60	60
def2-SVP	60	60	40
cc-pVTZ	40	60	10
pc-2	60	50	10
def2-TZVP	30	50	10

Table B.3: Optimal target fragment sizes across varying basis sets and theory levels.

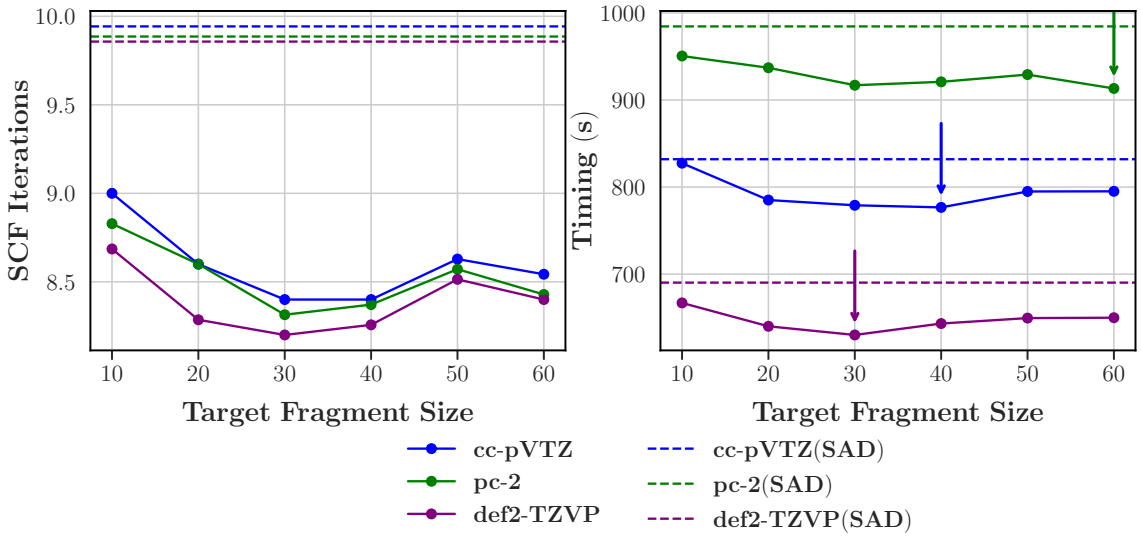


Figure B.14: Average number of SCF iterations and wall-times of triple-ζ basis set MBE calculations with varying target fragment sizes (no. of atoms) at the HF level. Arrows indicate the fragment size with the lowest average wall-time. Average SAD wall-times of cc-pVTZ, pc-2 and def2-TZVP are 831.96, 984.47 and 690.27 s, respectively.

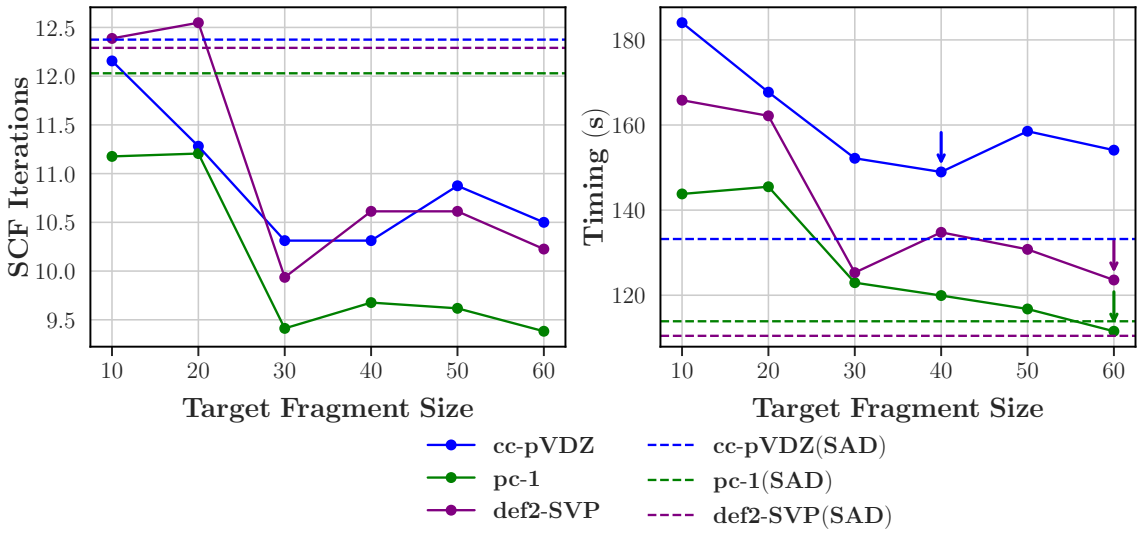


Figure B.15: Average number of SCF iterations and wall-times of double-ζ basis set MBE calculations with varying target fragment sizes (no. of atoms) at the B3LYP level. Arrows indicate the fragment size with the lowest average wall-time.

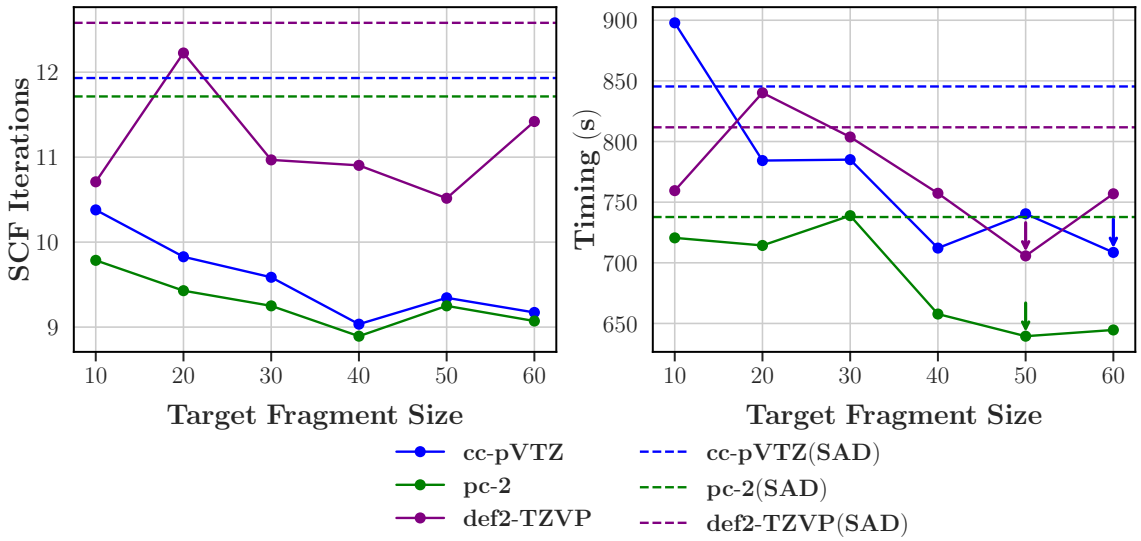


Figure B.16: Average number of SCF iterations and wall-times of triple-ζ basis set MBE calculations with varying target fragment sizes (no. of atoms) at the B3LYP level. Arrows indicate the fragment size with the lowest average wall-time.

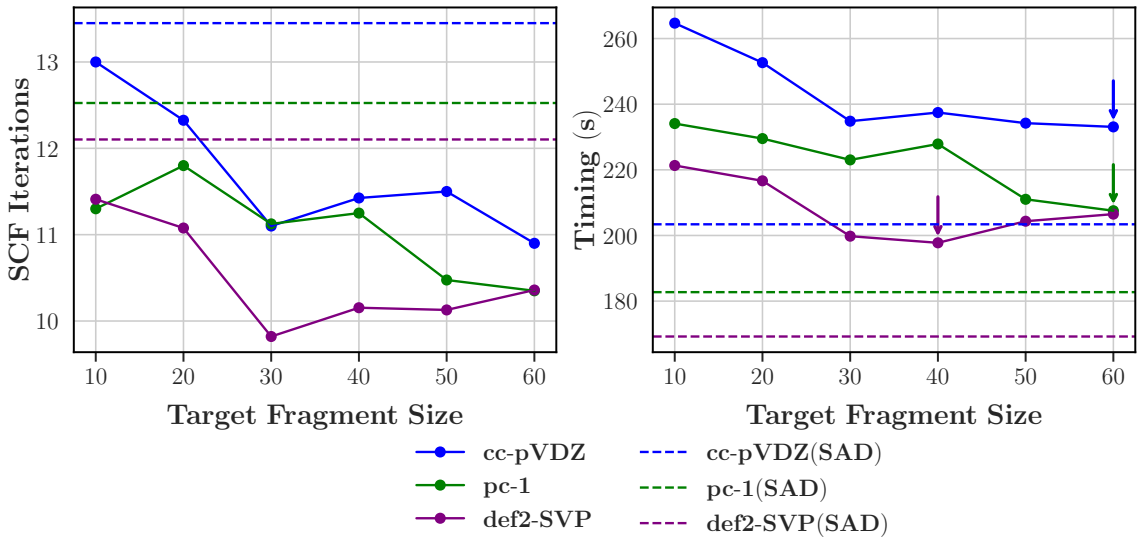


Figure B.17: Average number of SCF iterations and wall-times of double-ζ basis set MBE calculations with varying target fragment sizes (no. of atoms) at the MN15 level. Arrows indicate the fragment size with the lowest average wall-time.

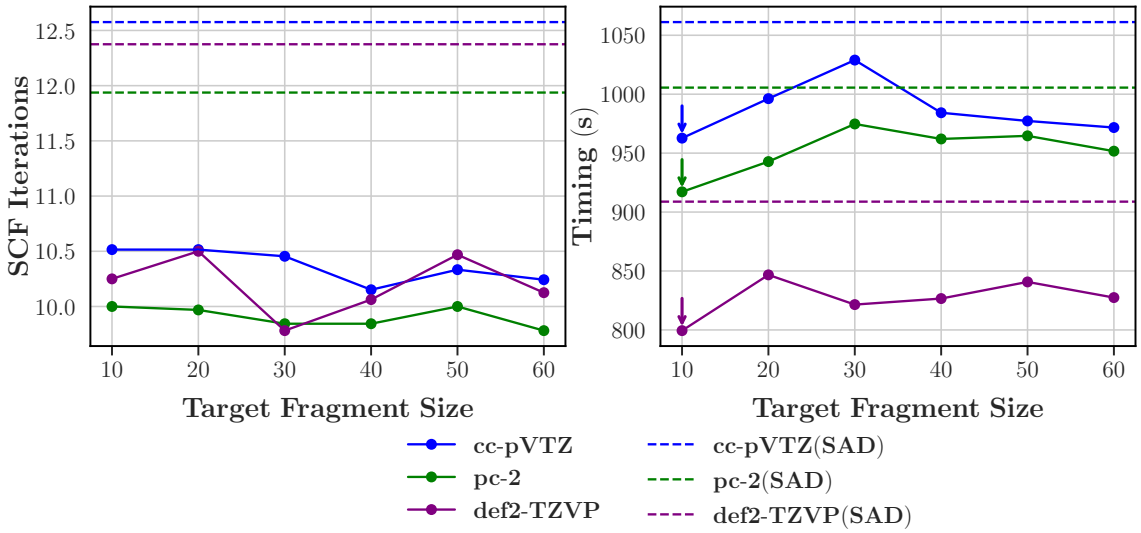


Figure B.18: Average number of SCF iterations and wall-times of triple-ζ basis set MBE calculations with varying target fragment sizes (no. of atoms) at the MN15 level. Arrows indicate the fragment size with the lowest average wall-time.

Convergence Threshold

Table B.4: Improvement in speedup relative to SAD of MBE calculations using optimal thresholds compared to 10^{-6} a.u. across various basis sets and levels of theory. The exponents, λ, of thresholds, 10^{λ}, are listed in parentheses.

	HF	**B3LYP**	**MN15**
cc-pVDZ	2.4%(-3)	0.0%(-6)	9.5%(-1)
pc-1	3.8%(-2)	6.8%(-1)	10.8%(-1)
def2-SVP	2.5%(-4)	4.8%(-3)	11.1%(-1)
cc-pVTZ	4.2%(-2)	0.4%(-1)	0.6%(-2)
pc-2	5.5%(-1)	3.3%(-3)	4.4%(-3)
def2-TZVP	4.7%(-1)	3.1%(-3)	2.7%(-3)

Table B.4 summarizes the improvement in speedups relative to SAD attained with optimal thresholds compared to 10^{-6} a.u. With double-ζ basis sets, improvements in the speedup by up to 11.1% could be observed with MN15 by using higher thresholds. However, as shown in Figure B.19, the resulting average wall-times remain higher than SAD due to the costly step of constructing the initial guess.

On the other hand for triple-ζ bases, marginal improvements in the speedups are observed ranging between 0.4% and 5.5%. With these improvements, these translate to average overall wall-time speedups relative to SAD by up to 9.1%, 11.7% and 6.8% for HF, B3LYP and MN15, respectively.

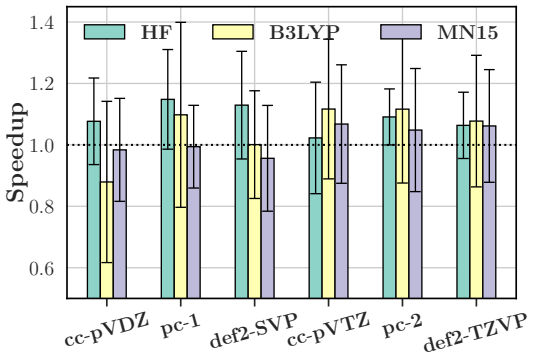


Figure B.19: Average wall-time speedups of MBE calculations for varying basis sets and theory levels. Optimal convergence thresholds are employed (listed in Table B.4). Error bars correspond to one standard deviation.

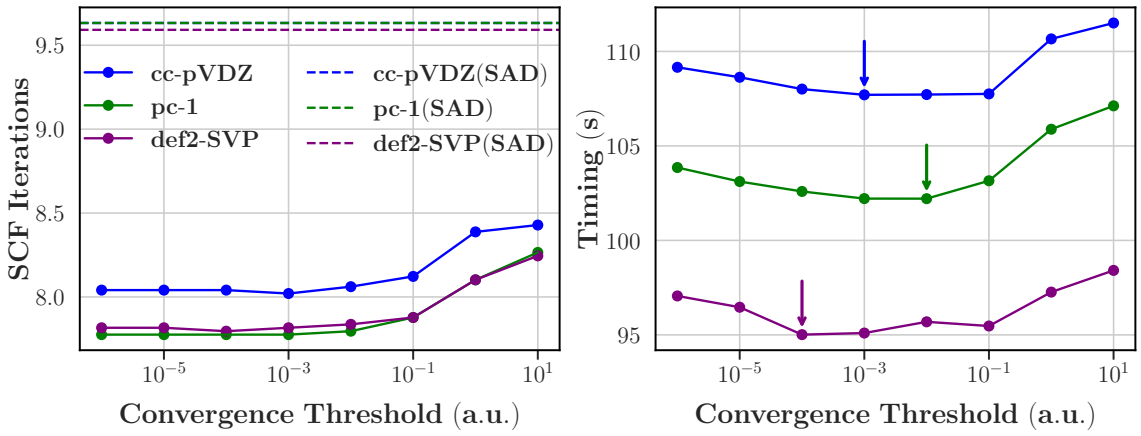


Figure B.20: Average number of SCF iterations and wall-times of double-ζ basis set MBE calculations with varying fragment convergence thresholds at the HF level. Arrows indicate the convergence threshold with the lowest average wall-time.

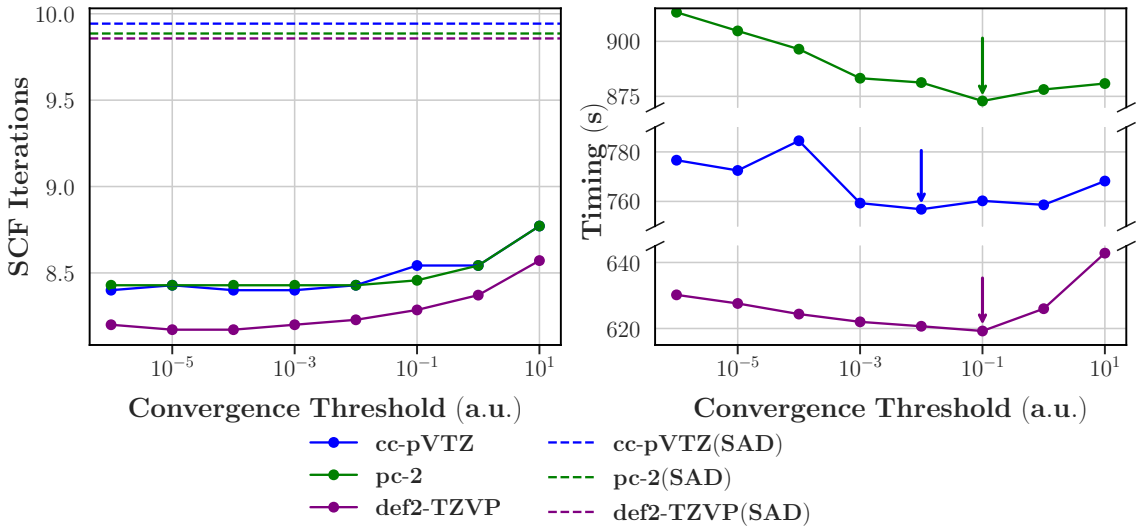


Figure B.21: Average number of SCF iterations and wall-times of triple-ζ basis set MBE calculations with varying fragment convergence thresholds at the HF level. Arrows indicate the convergence threshold with the lowest average wall-time.

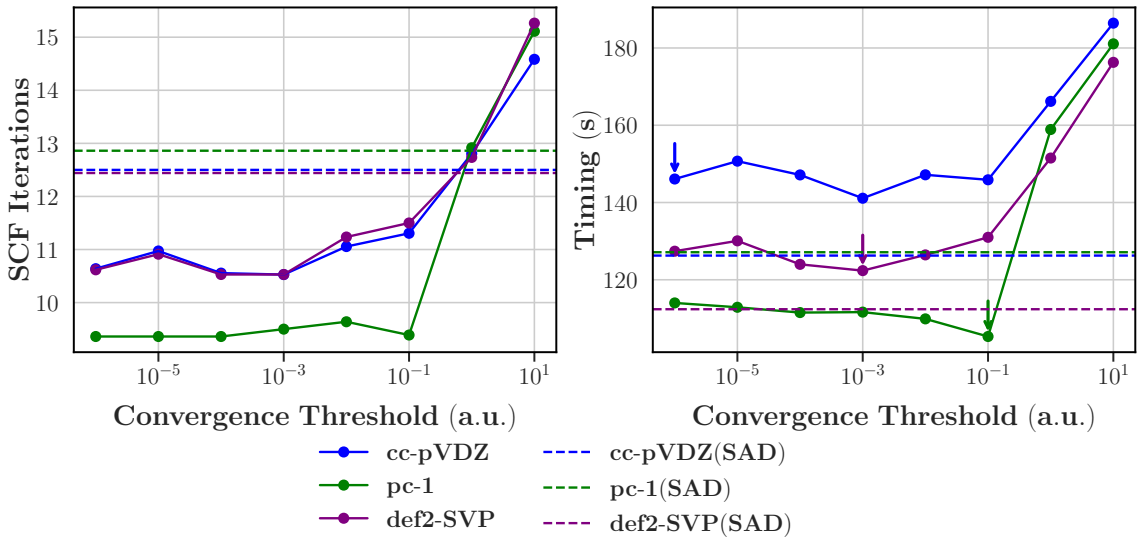


Figure B.22: Average number of SCF iterations and wall-times of double-ζ basis set MBE calculations with varying fragment convergence thresholds at the B3LYP level. Arrows indicate the convergence threshold with the lowest average wall-time.

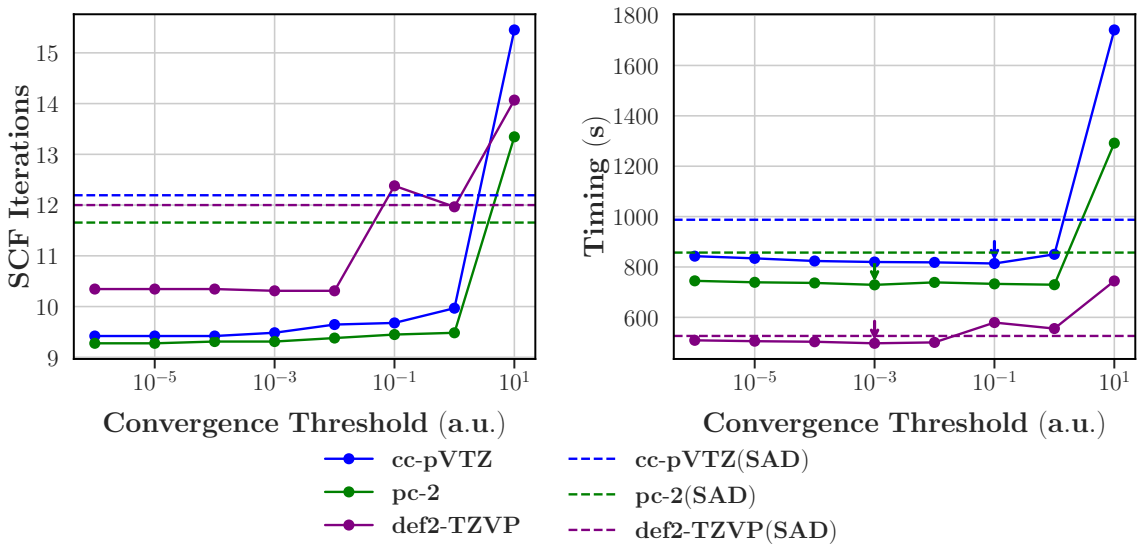


Figure B.23: Average number of SCF iterations and wall-times of triple-ζ basis set MBE calculations with varying fragment convergence thresholds at the B3LYP level. Arrows indicate the convergence threshold with the lowest average wall-time.

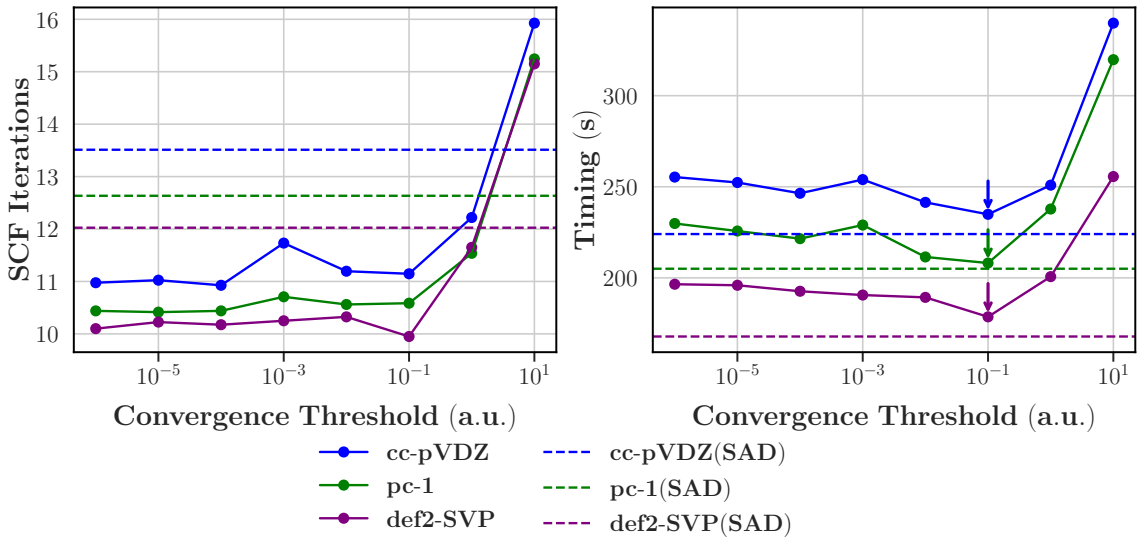


Figure B.24: Average number of SCF iterations and wall-times of double-ζ basis set MBE calculations with varying fragment convergence thresholds at the MN15 level. Arrows indicate the convergence threshold with the lowest average wall-time.

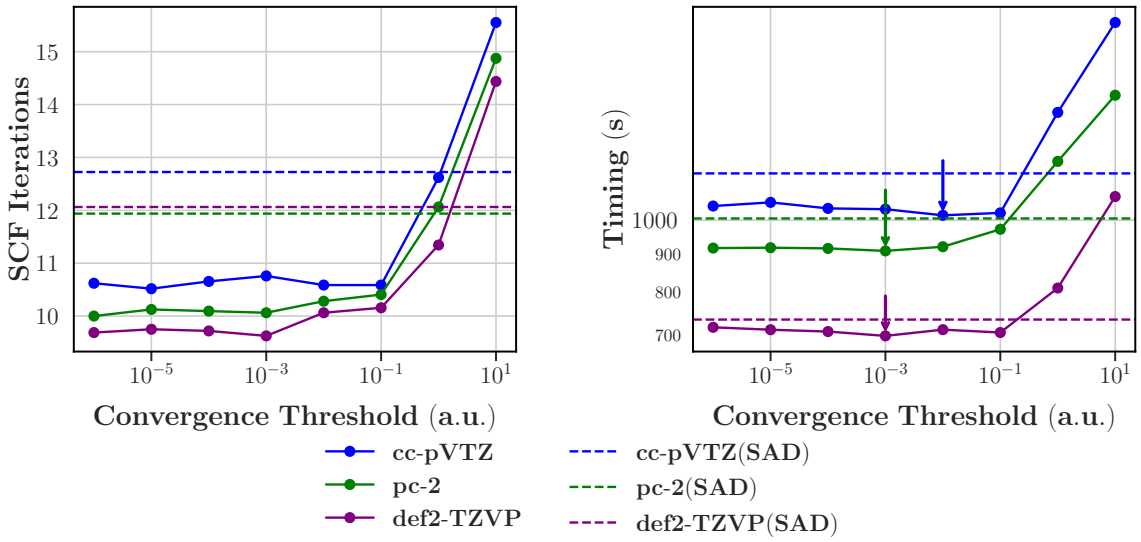


Figure B.25: Average number of SCF iterations and wall-times of triple-ζ basis set MBE calculations with varying fragment convergence thresholds at the MN15 level. Arrows indicate the convergence threshold with the lowest average wall-time.

Resolution-of-the-identity Approximation

RI calculations are constrained to Dunning's correlation consistent and Karlsruhe basis sets as auxiliary bases are available for these. Corresponding optimal fragment sizes and convergence thresholds as discussed in the main text were employed.

Since auxiliary bases are only available for Dunning's correlation consistent and

Karlsruhe basis sets, the discussion of the RI-HF approximation is constrained to these bases. Corresponding optimal fragment sizes and convergence thresholds as discussed earlier were employed.

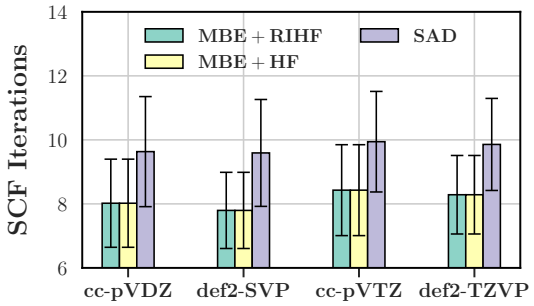


Figure B.26: Average SCF iteration count of MBE calculations with and without RI-HF approximation. SAD results are provided for reference. Error bars correspond to one standard deviation.

Figure B.26 illustrates the average SCF iteration count with and without the RI-HF approximation for the MBE initial guess across all basis sets. There is virtually no difference in the SCF iteration counts between calculations with and without RI-HF, indicating that RI-HF produces initial guesses of similar accuracy to HF.

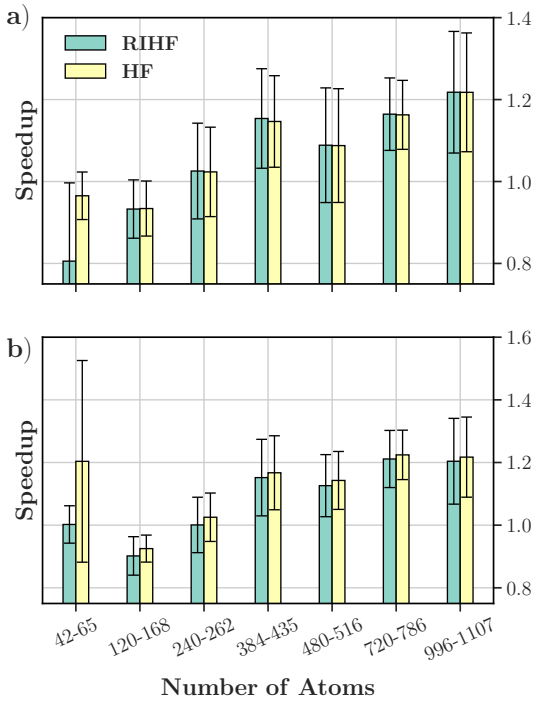


Figure B.27: Average wall-time speedups of MBE calculations with and without RI-HF for increasingly large systems employing the a) cc-pVDZ; and b) def2-SVP double-ζ basis sets. Number of atoms correspond to the system size.

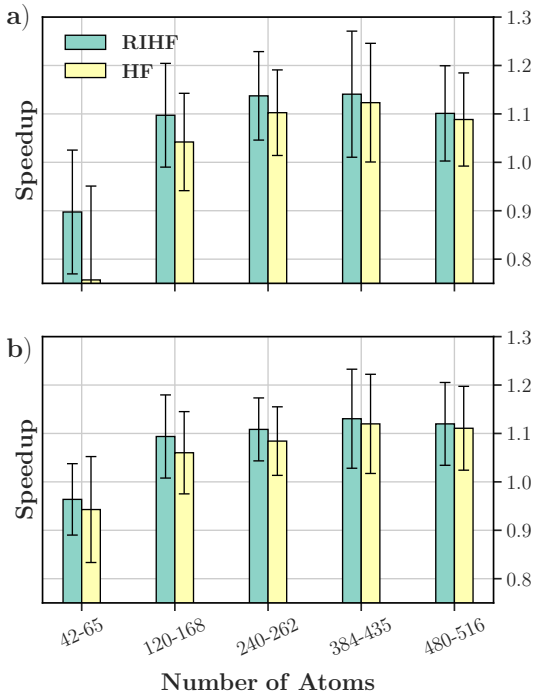


Figure B.28: Average wall-time speedups of MBE calculations with and without RI-HF for increasingly large systems employing the a) cc-pVTZ; and b) def2-TZVP triple-ζ basis sets. Number of atoms correspond to the system size.

Figures B.27 and B.28 showcase the variation of the wall-time speedups with system size with and without RI for the double- and triple-ζ basis sets, respectively. In particular for the double-ζ bases, as system size grows larger the speedups of RI-HF is only marginally higher than HF for cc-pVDZ. On the other hand for def2-SVP, RI-HF consistently performs worse than HF across all system sizes. The 'lower' performance of RI-HF against HF for smaller systems (42 and 65 atoms) is a result of the target fragment sizes employed (50 and 60 atoms for cc-pVDZ and def2-SVP, respectively) being comparable to the system size. In these cases, no fragmentation occurs, and computations labelled as HF in Figure B.28 correspond to a single SCF calculation (using SAD). Conversely, those labelled as RI-HF involves an additional SCF routine (a RI-HF calculation to construct the initial guess), yielding higher wall-times.

As displayed in Figure B.28, RI-HF consistently exhibits higher wall-time speedups than HF across all system sizes in the triple-ζ basis set calculations. The main source of these improvements is the lower computational overhead associated with the construction of the initial guess; the proportion of time spent assembling the initial guess reduces by 6.1% and 8.0% with RI-HF for cc-pVTZ and def2-TZVP, respectively, on average. With RI-HF, the decrease average wall-times are marginal; 2.0% and 1.0% for cc-pVTZ and def2-TZVP, respectively.

These differences in the performance of RI-HF between double- and triple-ζ basis

sets are primarily attributed to the optimal target fragment sizes used. As stated earlier, the optimal target fragment sizes for the triple-ζ bases are smaller (40 and 30 atoms for cc-pVTZ and def2-TZVP, respectively) than their double-ζ counterparts (50 and 60 atoms, respectively).

To emphasise this sentiment, we compare the existing results of the RI-HF calculations which employ the optimal target fragment sizes with those obtained with a target fragment size of 10 atoms. On average, with a target fragment size of 10 atoms, the time spent in the initial guess construction step was 1.45×, 1.69×, 2.87× and 1.53× faster for cc-pVDZ, def2-SVP, cc-pVTZ and def2-TZVP, respectively, than with the optimal target fragment size. However, this improvement is outweighed by an increase in the SCF iteration count due to less accurate initial guesses; wall-times increased on average by 8.7%, 8.6%, 3.7% and 3.3% for cc-pVDZ, def2-SVP, cc-pVTZ and def2-TZVP, respectively.

Such results highlight the slight advantage of employing approximate HF methods such as RI-HF in reducing wall-times of MBE calculations. However, these improvements are marginal and are generally limited to triple-ζ basis sets.

Fragmentation Level

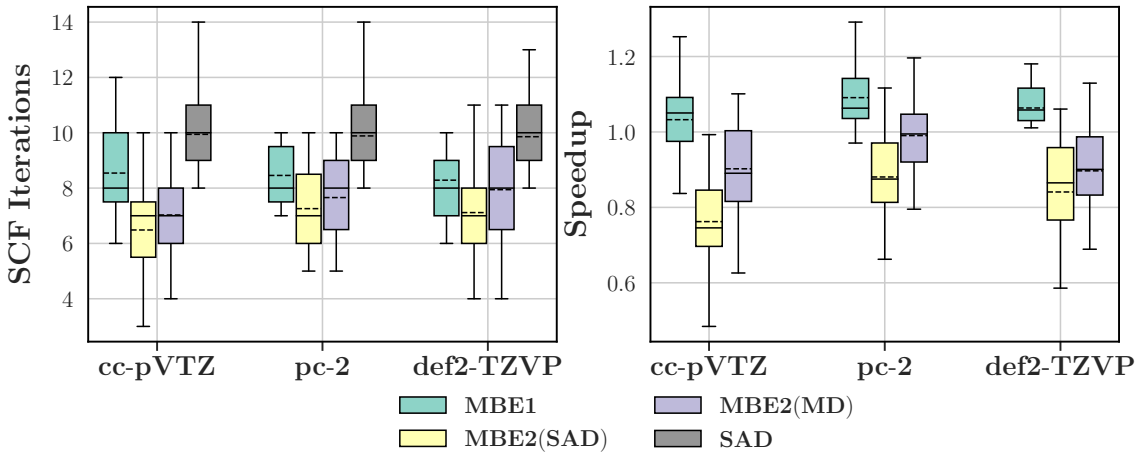


Figure B.29: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set MBE calculations at various fragmentation levels. No cutoffs were used for MBE2.

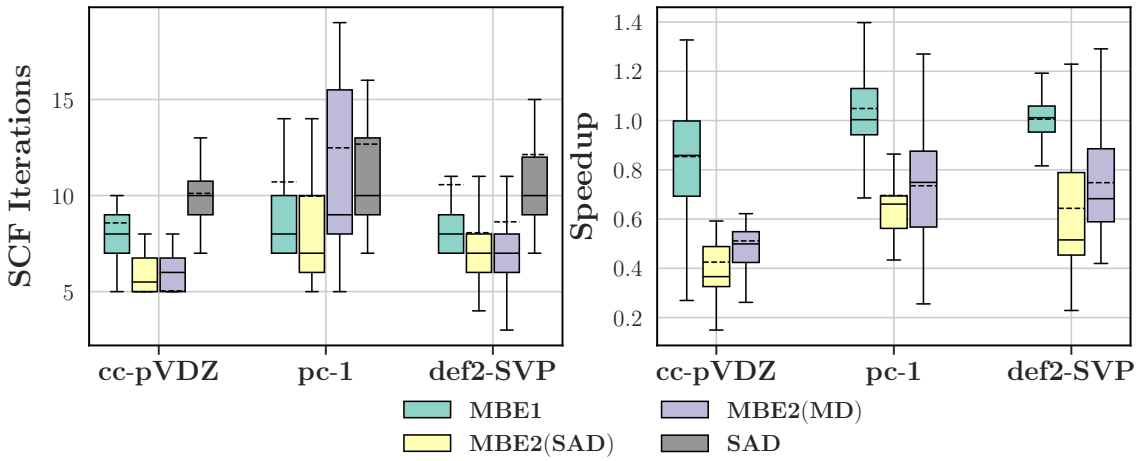


Figure B.30: Distribution of SCF iterations and wall-time speedups of double-ζ basis set MBE calculations at various fragmentation levels at the B3LYP level. No cutoffs were used for MBE2.

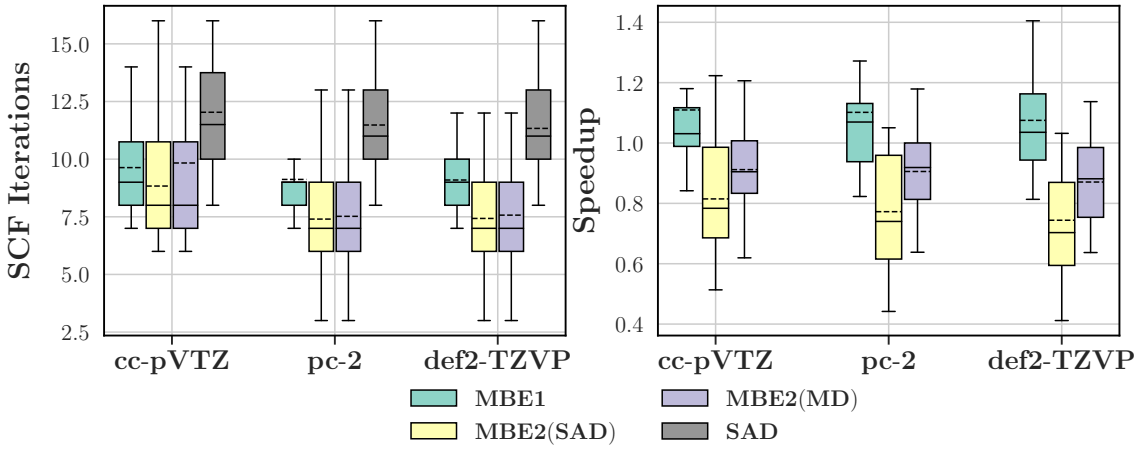


Figure B.31: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set MBE calculations at various fragmentation levels at the B3LYP level. No cutoffs were used for MBE2.

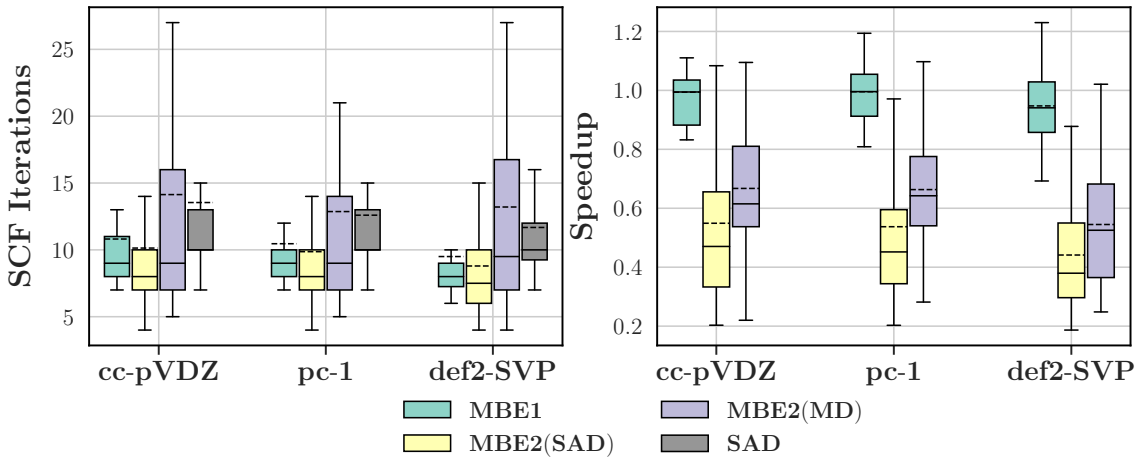


Figure B.32: Distribution of SCF iterations and wall-time speedups of double-ζ basis set MBE calculations at various fragmentation levels at the MN15 level. No cutoffs were used for MBE2.

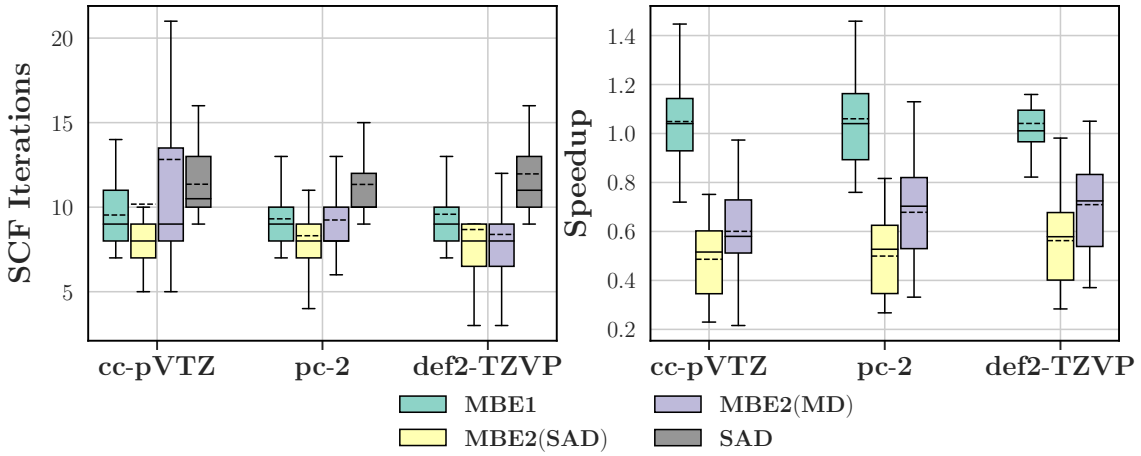


Figure B.33: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set MBE calculations at various fragmentation levels at the B3LYP level. No cutoffs were used for MN15.

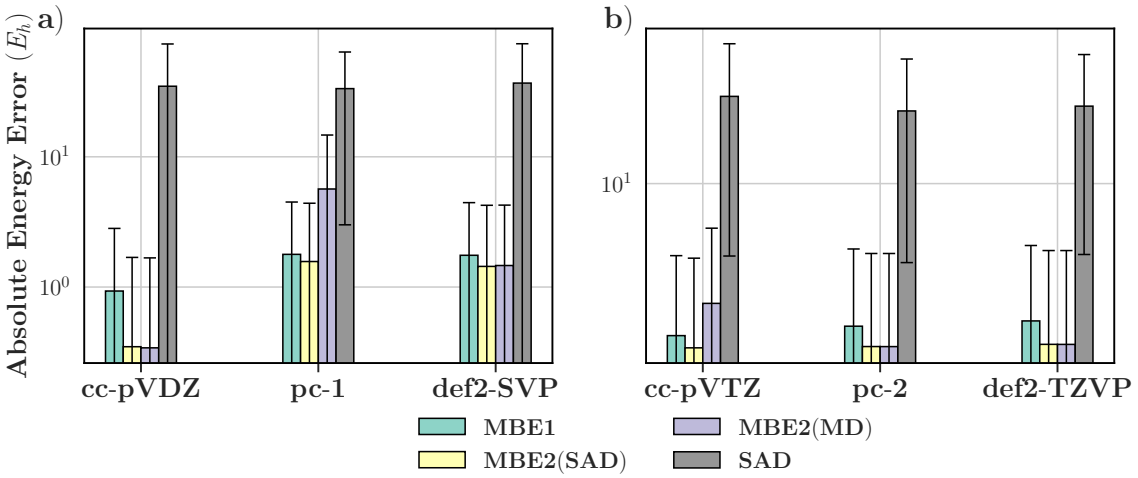


Figure B.34: Average absolute errors of initial guess energies relative to the converged energies a) double-ζ; and b) triple-ζ MBE computations at various fragmentation levels at the B3LYP level. No cutoffs were used for MBE2. Error bars corresponds to one standard deviation.

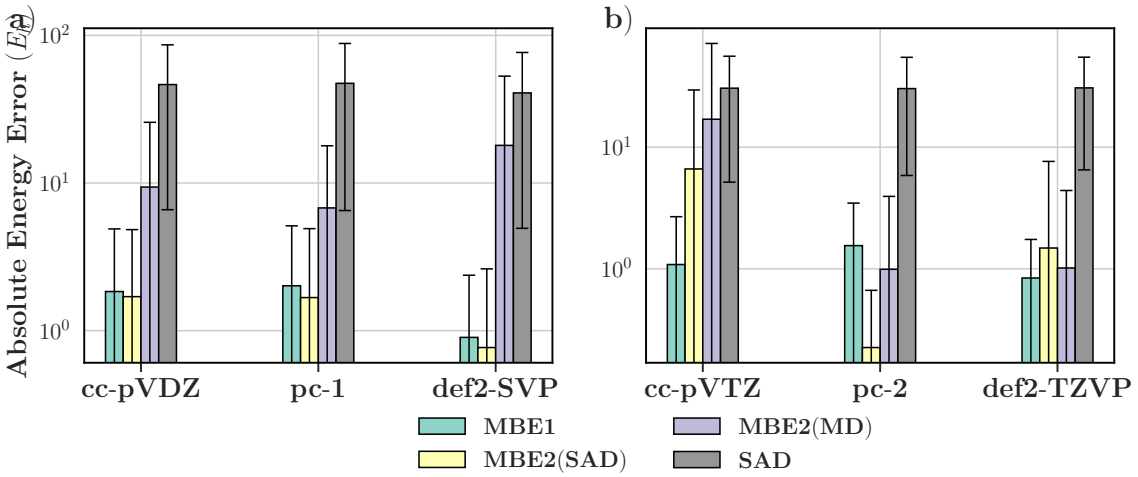


Figure B.35: Average absolute errors of initial guess energies relative to the converged energies a) double-ζ; and b) triple-ζ MBE computations at various fragmentation levels at the MN15 level. No cutoffs were used for MBE2. Error bars corresponds to one standard deviation.

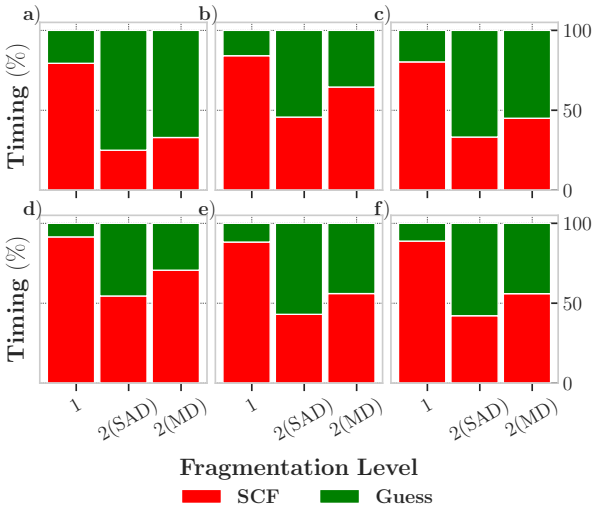


Figure B.36: Average timing breakdown of MBE calculations at various fragmentation levels for a) cc-pVDZ; b) pc-1; c) def2-SVP; d) cc-pVTZ; e) pc-2; and f) def2-TZVP basis sets at the B3LYP level. No dimer cutoffs were used for MBE2.

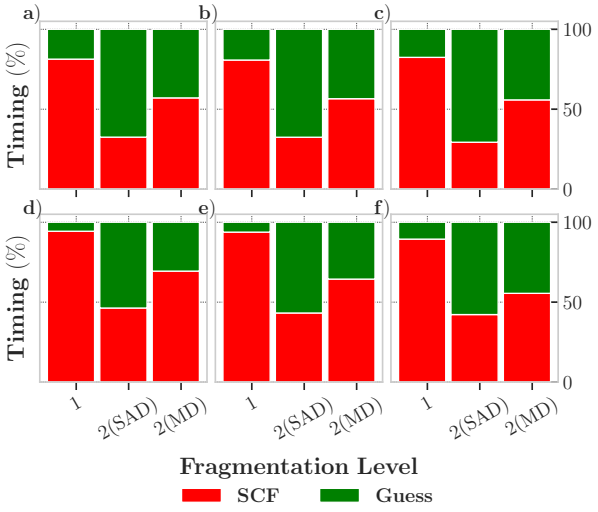


Figure B.37: Average timing breakdown of MBE calculations at various fragmentation levels for a) cc-pVDZ; b) pc-1; c) def2-SVP; d) cc-pVTZ; e) pc-2; and f) def2-TZVP basis sets at the MN15 level. No dimer cutoffs were used for MBE2.

Thus far, all MBE2 results reported employ no dimer cutoffs. Due to the substantial variation in guess timing between MBE1 and MBE2 computations being attributed to the presence of dimers, we test two cutoff distances (5 and 10 Å) to restrict the number of dimers assembled. Since MBE2(MD) consistently outperforms MBE2(SAD) across HF, B3LYP and MN15, we proceed with MBE2(MD) for the testing of cutoffs.

Figures B.40-B.44 illustrate the average SCF iteration count and wall-time speedups of MBE2 computations with varying cutoffs over the different theory levels and basis

sets. In HF, for both double- and triple-ζ calculations, the number of iterations generally reduces with larger cutoffs; larger cutoffs lead to the inclusion of more dimer interactions and therefore more accurate initial guesses.

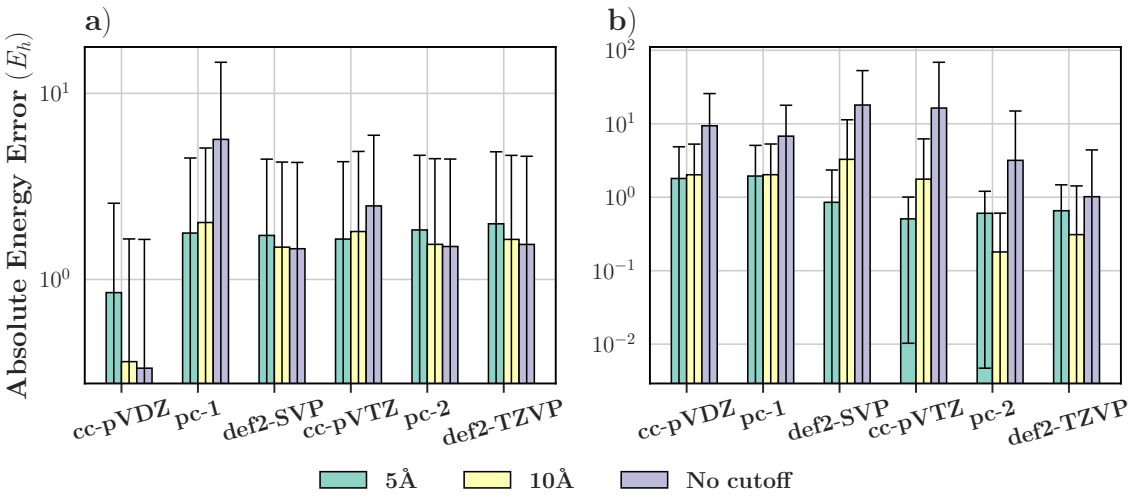


Figure B.38: Average absolute errors of initial guess energies relative to the converged energies of a) B3LYP and b) MN15 MBE2 computations with varying cutoffs. Error bars correspond to one standard deviation.

On the other hand for B3LYP and MN15, MBE2 without any cutoffs frequently yields higher SCF iteration counts. This is consistent with with the corresponding energy differences between the converged value and that of the initial guess presented in Figure B.38, which shows that 'no cutoff' in the MBE2 initial guess leads to either the same or an order of magnitude greater error than those with cutoffs imposed. This is likely linked to the observation made earlier of the non-convergent nature of hybrid DFT MBE calculations.[240] Capturing a greater degree of interactions by increasing fragmentation level from the one- to two-body level and by increasing the range of the dimer cutoff does not guarantee a better quality initial guess in hybrid DFT calculations.

As shown in Figures B.40-B.44, across all theory levels and basis sets, a cutoff of 5 Å consistently exhibits the highest average wall-time peedup. However, MBE1 still outperforms MBE2(5 Å) with respect to the wall-time speedups. The highest average speedups achieved with MBE2(5 Å) with HF and B3LYP were respectively 9.9% and 8.2% while its respective MBE1 calculation exhibited a speedup of 14.8% and 10.9%. Furthermore, MN15 MBE2(5 Å) led to mean wall-times higher than SAD whereas average speedups up to 6.0% could be observed with MBE1. Thus, while the imposition of dimer cutoffs yields an increase in the wall-time speedup of MBE2, the resulting acceleration remains less than that of MBE1.

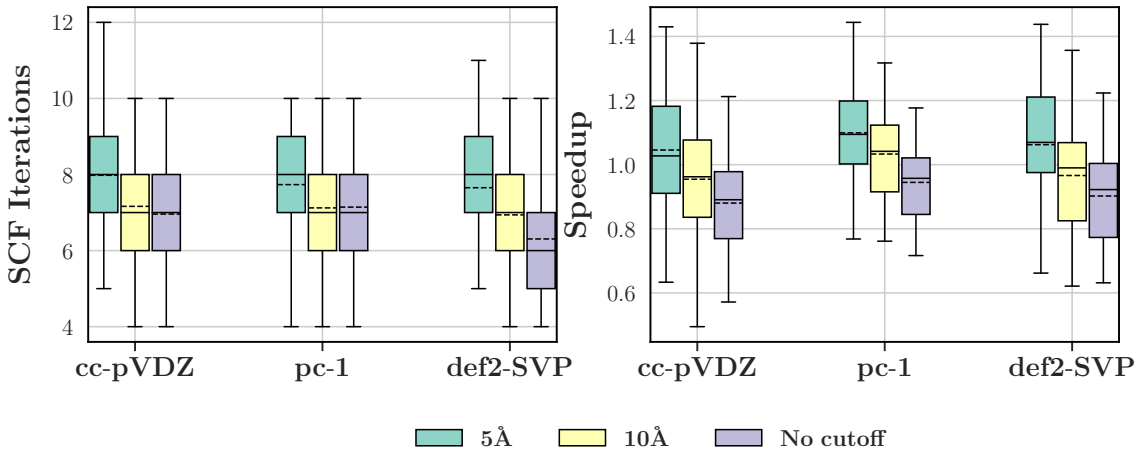


Figure B.39: Distribution of SCF iterations and wall-time speedups of double-ζ basis set MBE2 calculations with various cutoffs at the HF level.

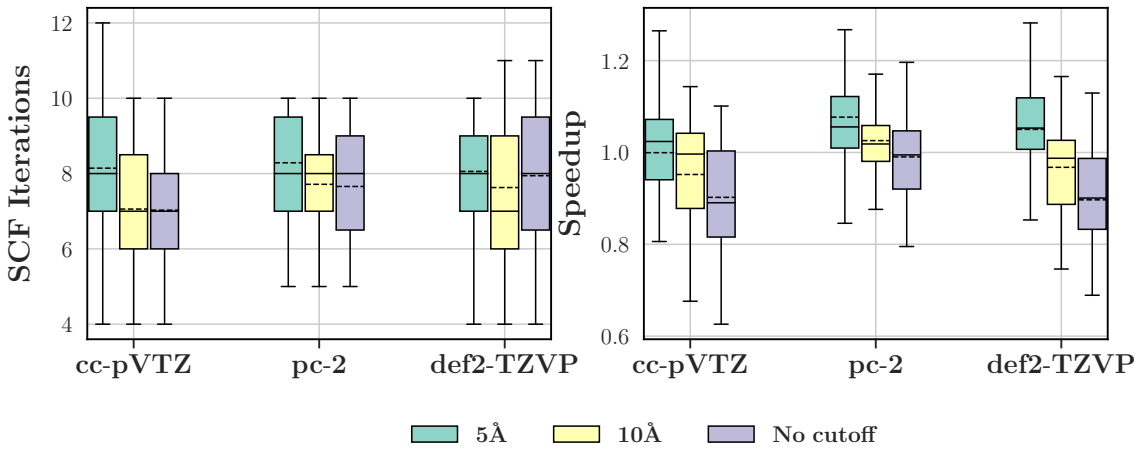


Figure B.40: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set MBE2 calculations with various cutoffs at the HF level.

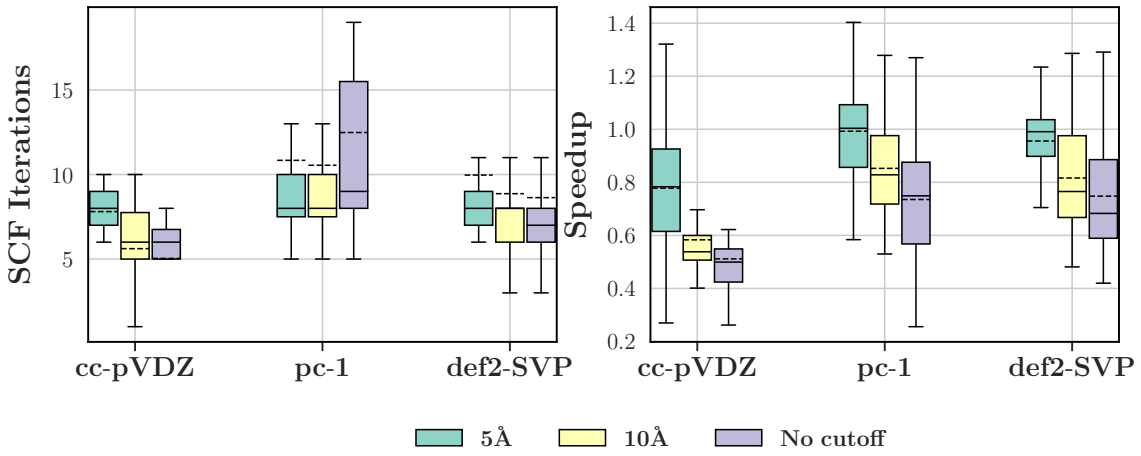


Figure B.41: Distribution of SCF iterations and wall-time speedups of double-ζ basis set MBE2 calculations with various cutoffs at the B3LYP level.

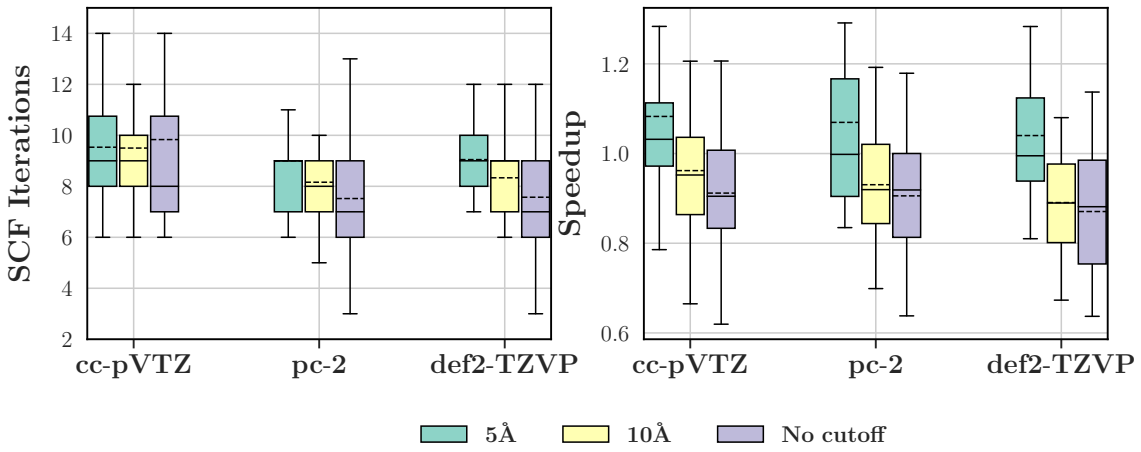


Figure B.42: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set MBE2 calculations with various cutoffs at the B3LYP level.

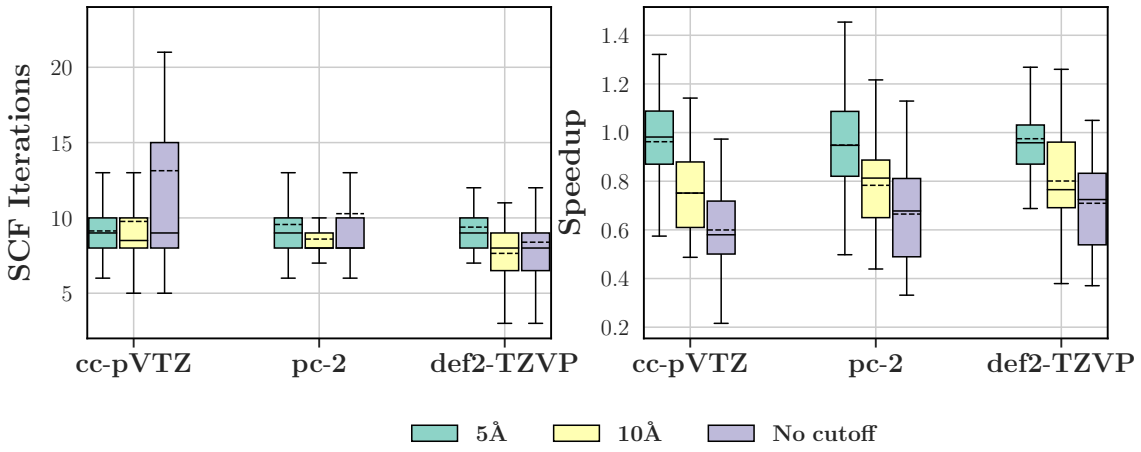


Figure B.44: Distribution of SCF iterations and wall-time speedups of triple-ζ basis set MBE2 calculations with various cutoffs at the MN15 level.

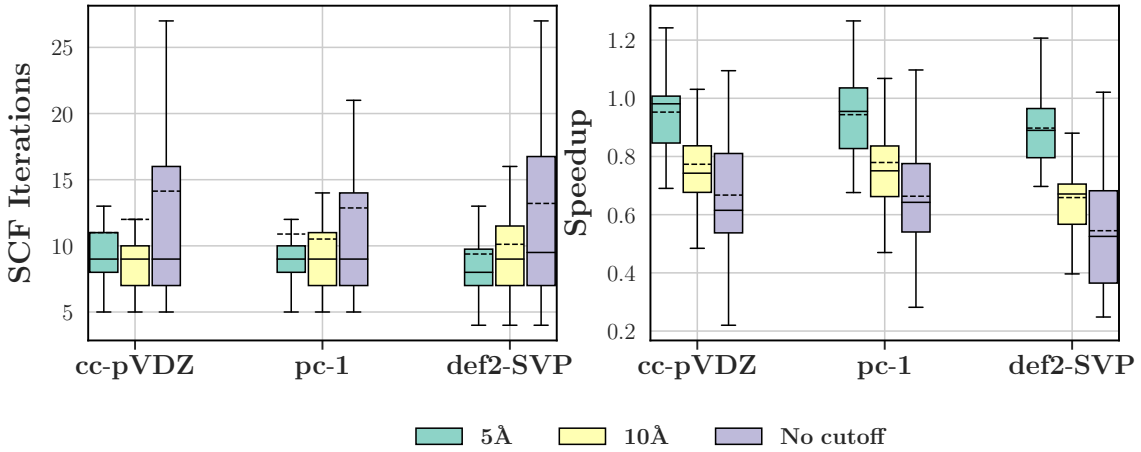


Figure B.43: Distribution of SCF iterations and wall-time speedups of double-ζ basis set MBE2 calculations with various cutoffs at the MN15 level.

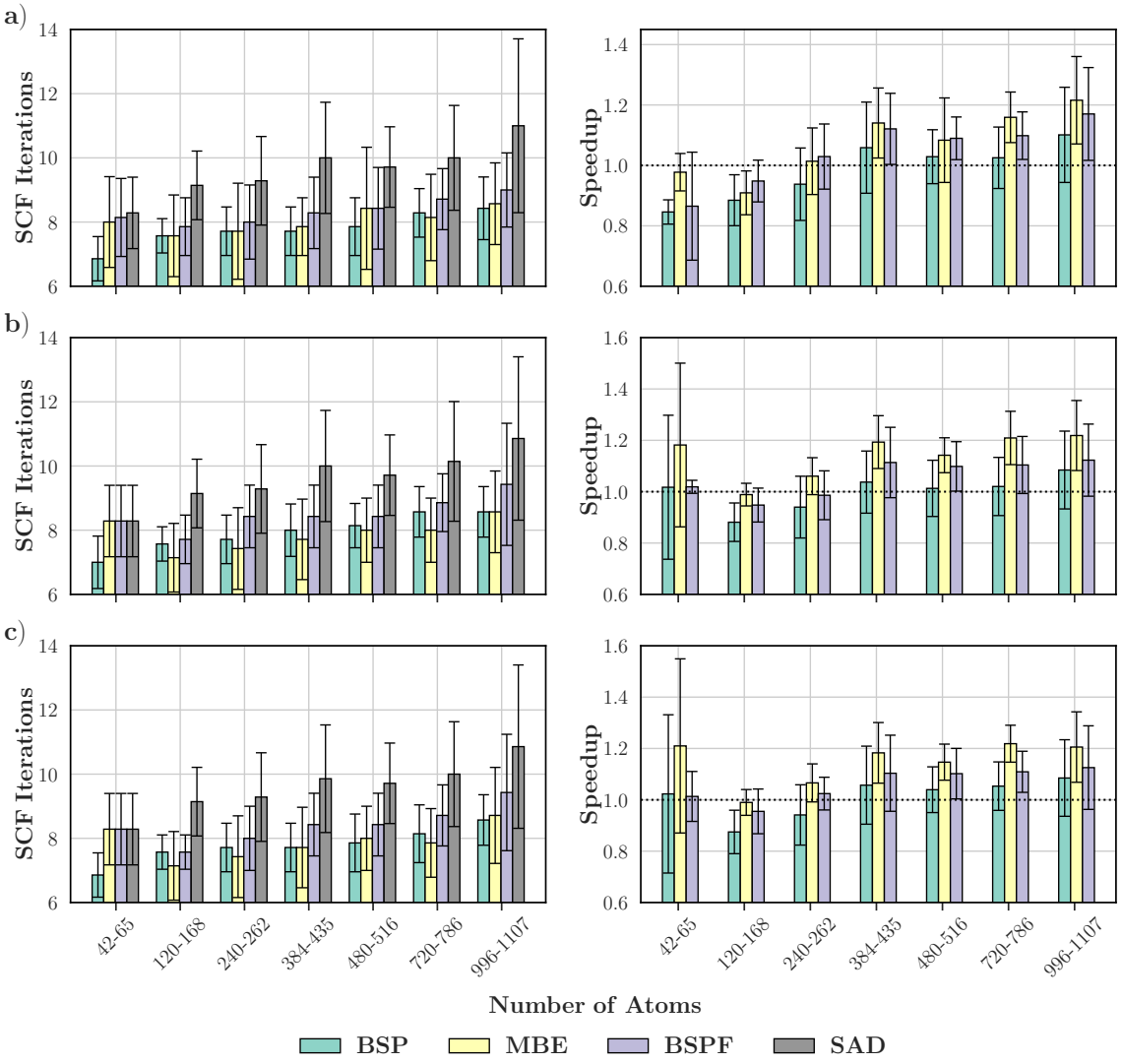


Figure B.45: Average number of SCF iterations and wall-time speedups of various initial guess approaches across for increasingly large systems employing the a) cc-pVDZ; b) pc-1; and c) def2-SVP double-ζ basis sets at the HF level. Error bars correspond to one standard deviation.

Comparison Between Initial Guess Methods

The following figures showcase the results for BSP, MBE, BSPF and SAD.

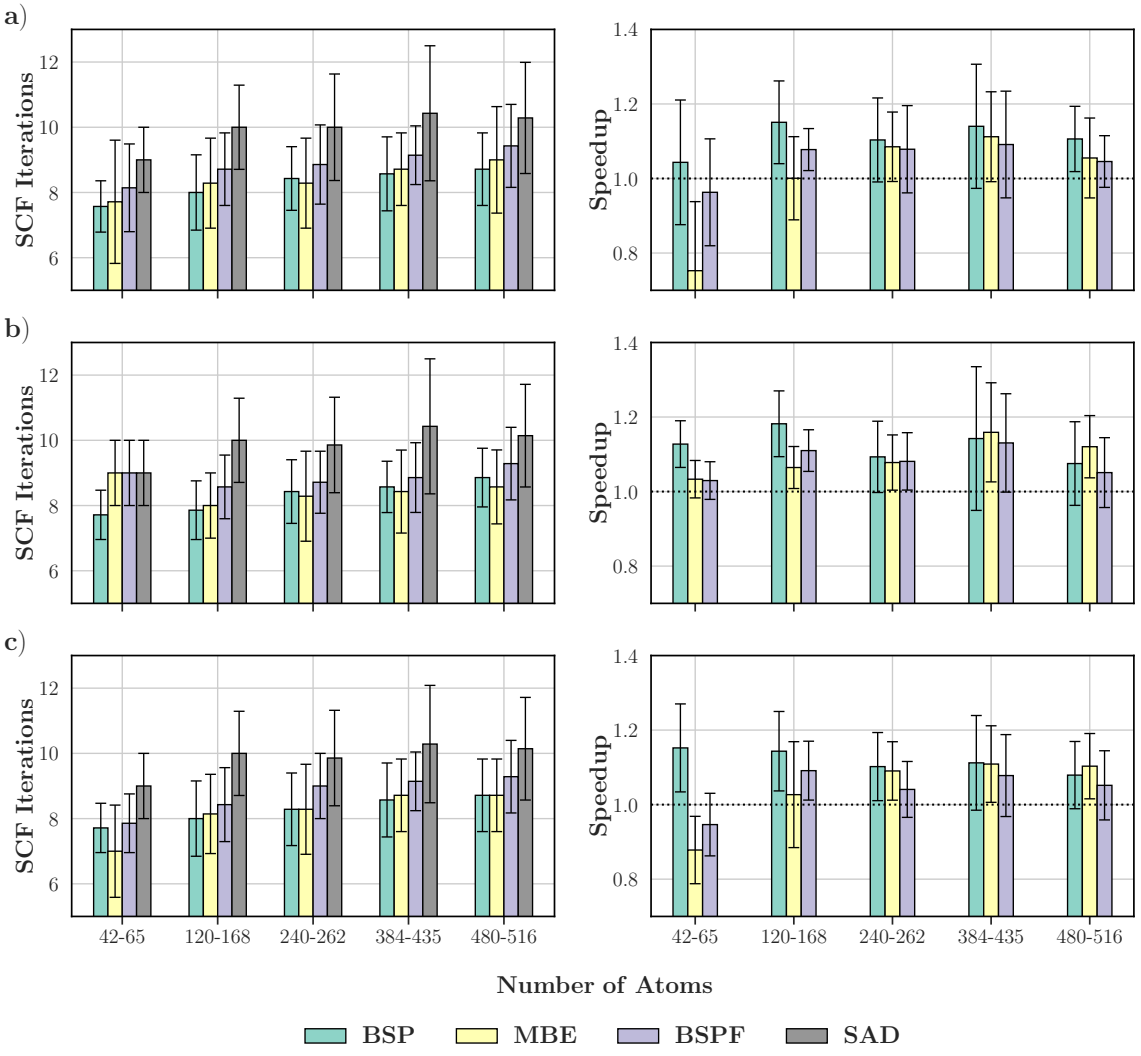


Figure B.46: Average number of SCF iterations and wall-time speedups of various initial guess approaches across for increasingly large systems employing the a) cc-pVTZ; b) pc-2; and c) def2-TZVP double-ζ basis sets at the HF level. Error bars correspond to one standard deviation.

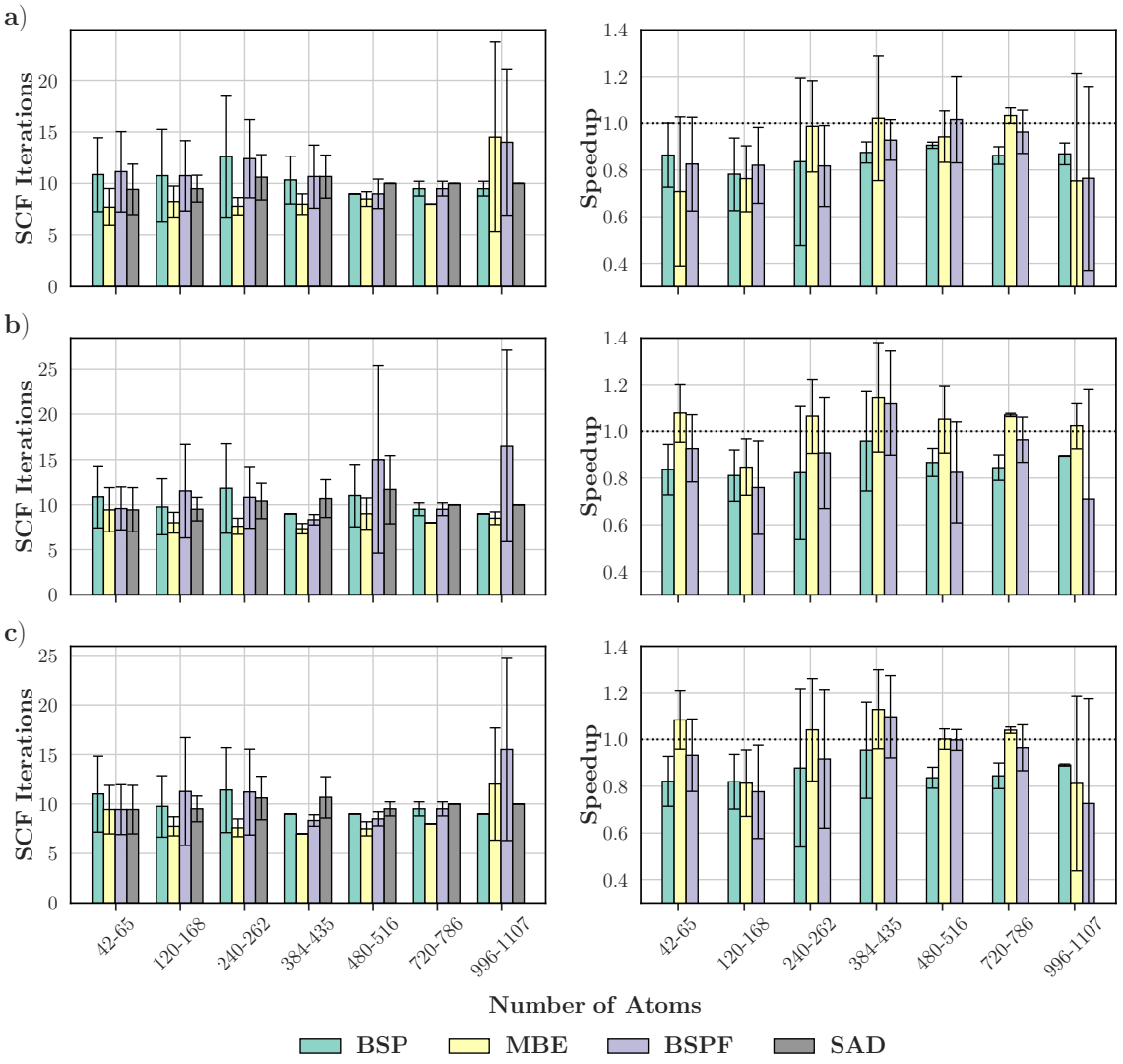


Figure B.47: Average number of SCF iterations and wall-time speedups of various initial guess approaches across for increasingly large systems employing the a) cc-pVDZ; b) pc-1; and c) def2-SVP double-ζ basis sets at the B3LYP level. Error bars correspond to one standard deviation.

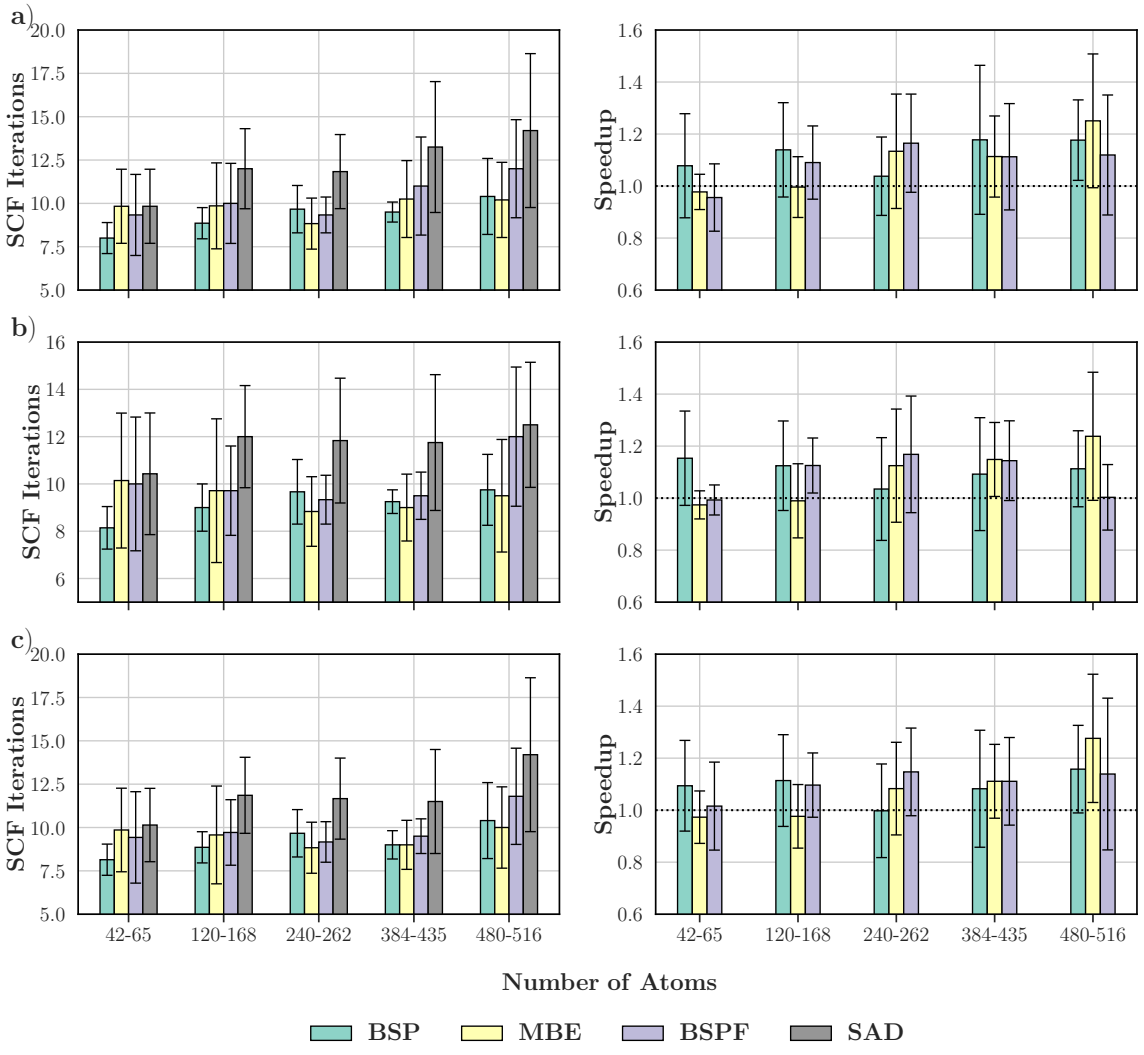


Figure B.48: Average number of SCF iterations and wall-time speedups of various initial guess approaches across for increasingly large systems employing the a) cc-pVTZ; b) pc-2; and c) def2-TZVP double-ζ basis sets at the B3LYP level. Error bars correspond to one standard deviation.

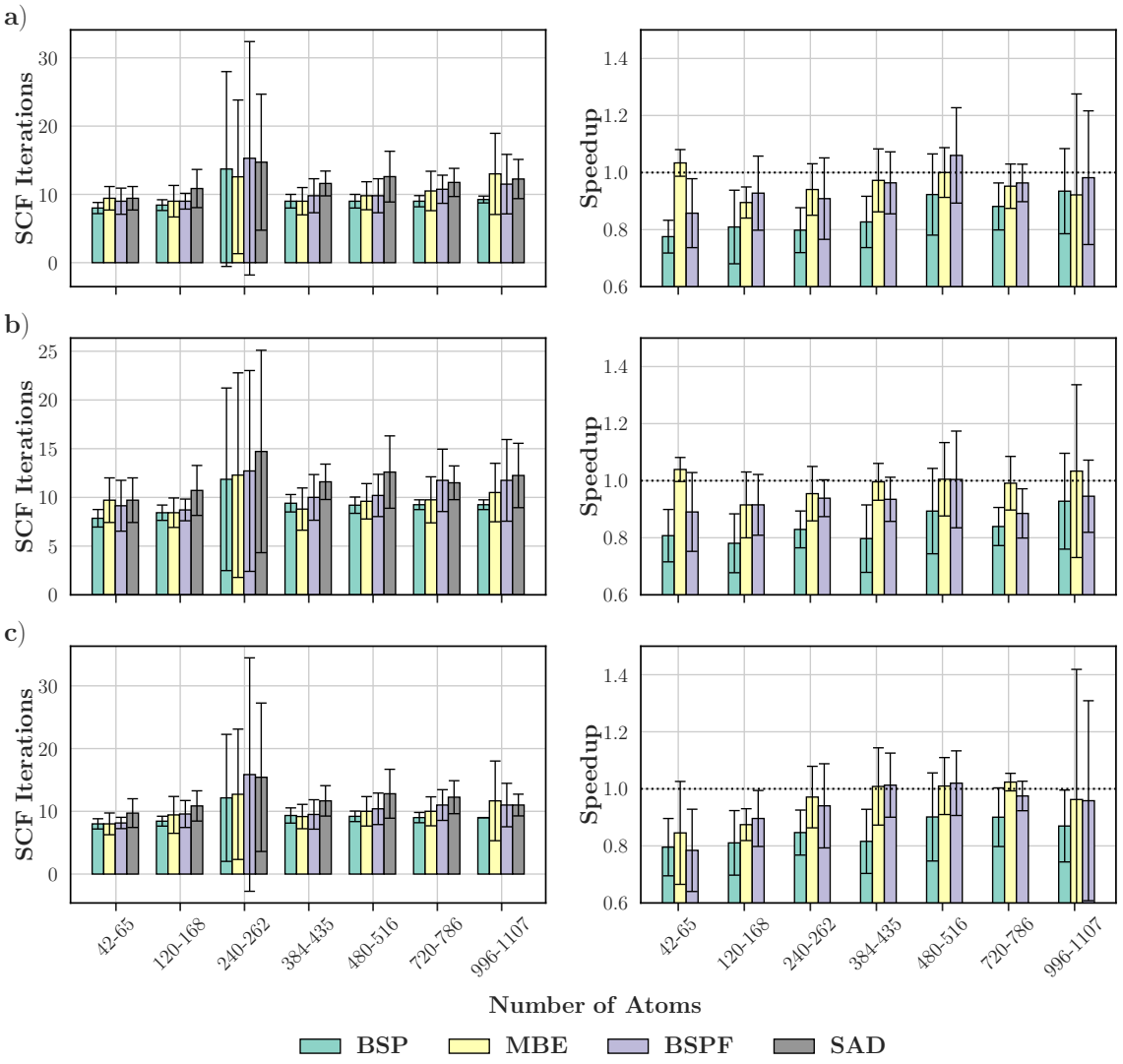


Figure B.49: Average number of SCF iterations and wall-time speedups of various initial guess approaches across for increasingly large systems employing the a) cc-pVDZ; b) pc-1; and c) def2-SVP double-ζ basis sets at the MN15 level. Error bars correspond to one standard deviation.

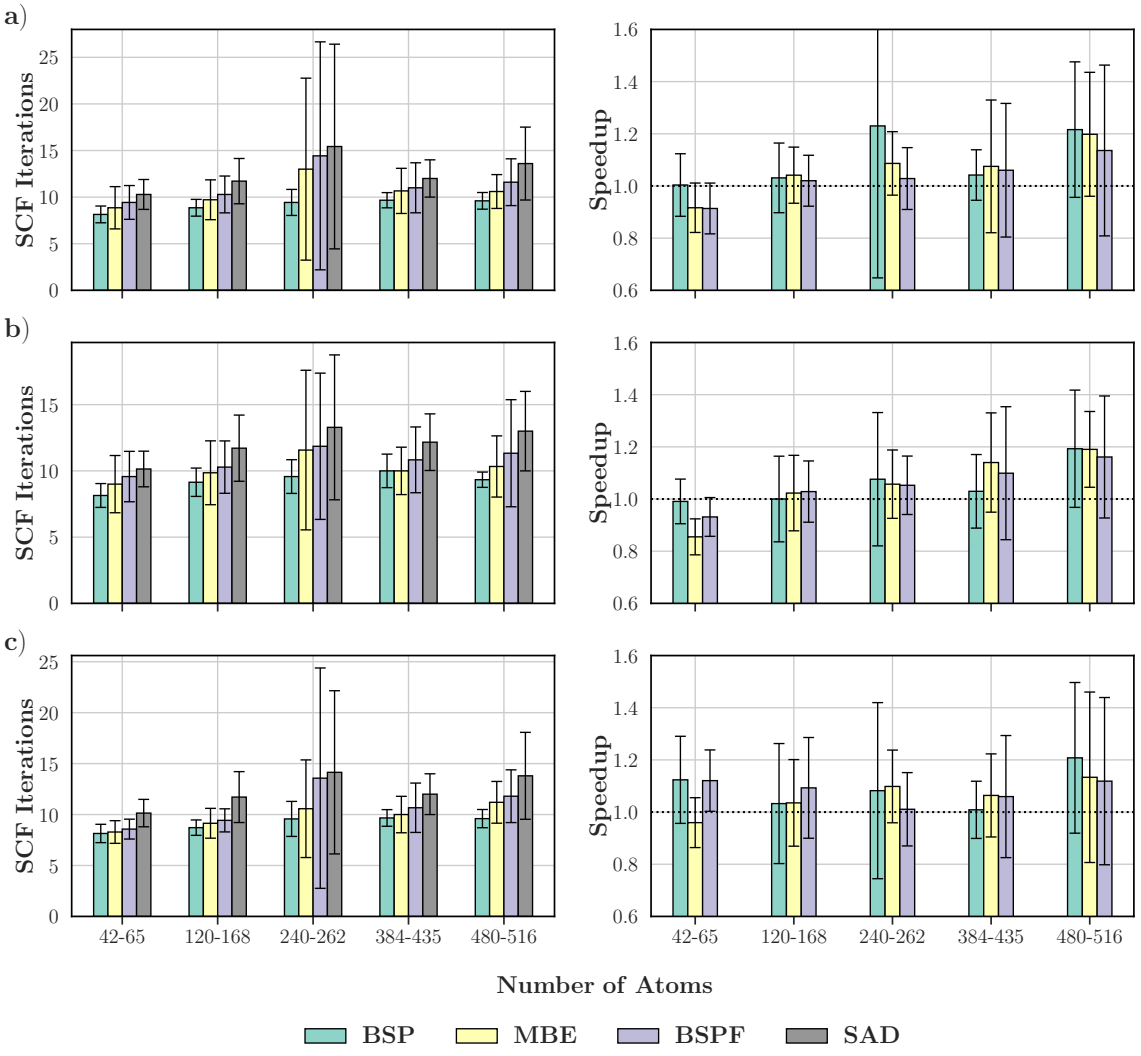


Figure B.50: Average number of SCF iterations and wall-time speedups of various initial guess approaches across for increasingly large systems employing the a) cc-pVTZ; b) pc-2; and c) def2-TZVP double-ζ basis sets at the MN15 level. Error bars correspond to one standard deviation.

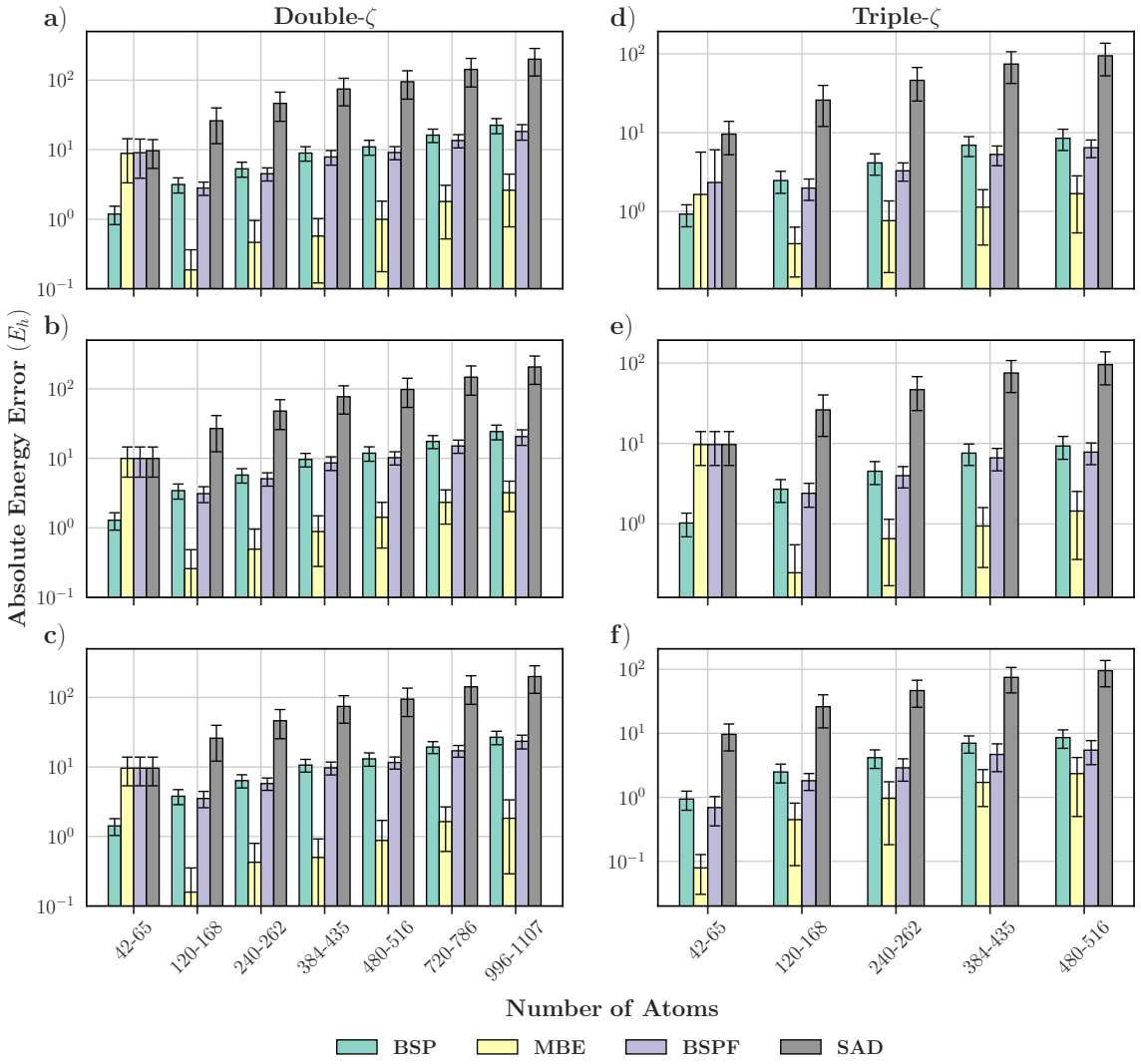


Figure B.51: Average absolute errors of initial guess energies relative to the converged energies of various initial guess methods for increasingly large systems employing double-ζ basis sets: a) cc-pVDZ; b) pc-1; c) def2-SVP and triple-ζ basis sets: d) cc-pVTZ; e) pc-2; f) def2-TZVP at the HF level. Error bars correspond to one standard deviation.

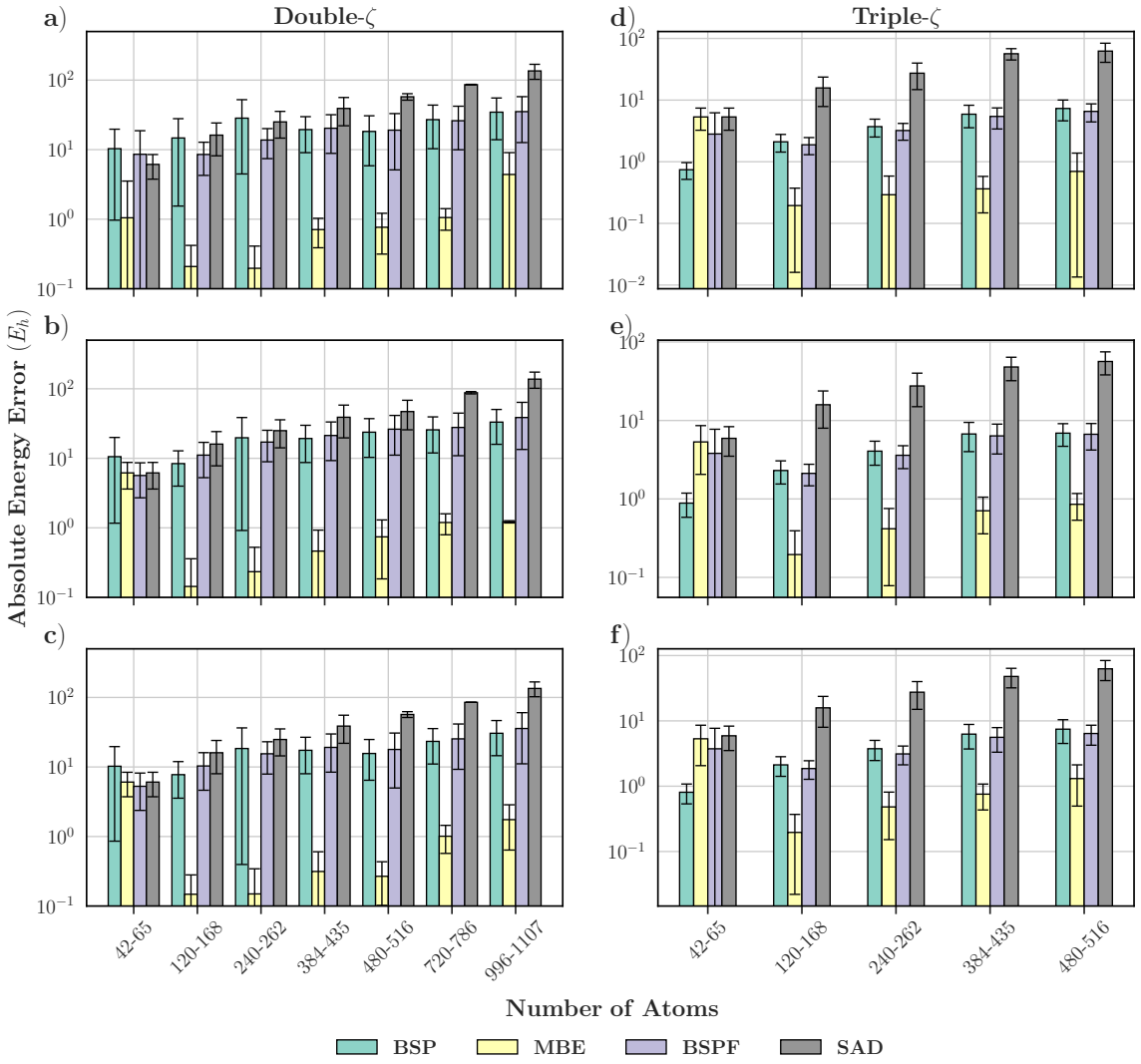


Figure B.52: Average absolute errors of initial guess energies relative to the converged energies of various initial guess methods for increasingly large systems employing double-ζ basis sets: a) cc-pVDZ; b) pc-1; c) def2-SVP and triple-ζ basis sets: d) cc-pVTZ; e) pc-2; f) def2-TZVP at the B3LYP level. Error bars correspond to one standard deviation.

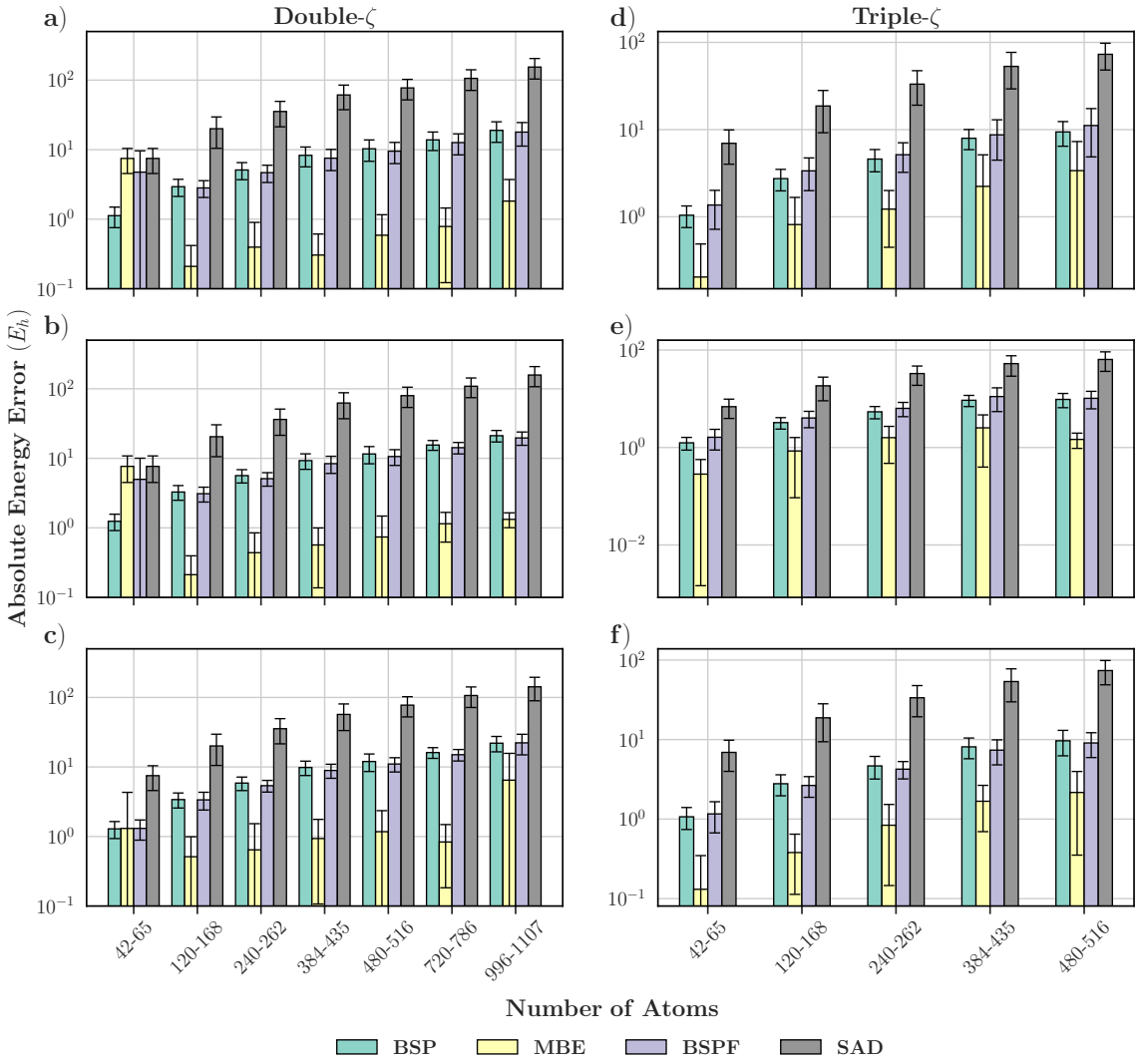


Figure B.53: Average absolute errors of initial guess energies relative to the converged energies of various initial guess methods for increasingly large systems employing double-ζ basis sets: a) cc-pVDZ; b) pc-1; c) def2-SVP and triple-ζ basis sets: d) cc-pVTZ; e) pc-2; f) def2-TZVP at the MN15 level. Error bars correspond to one standard deviation.

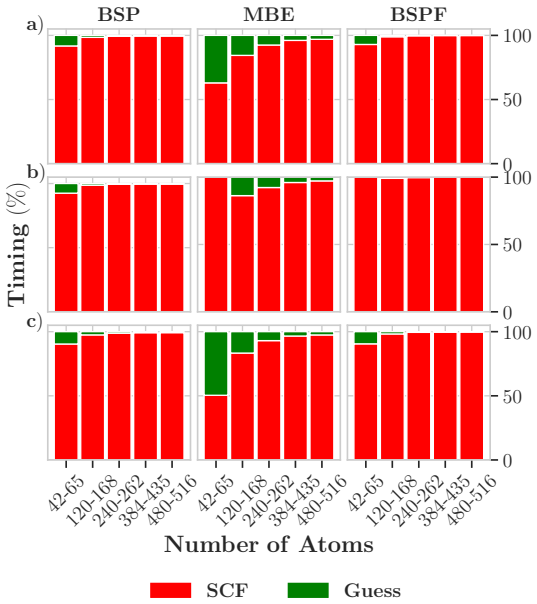


Figure B.54: Average timing breakdown of the BSP, MBE, BSPF initial guess approaches with varying system sizes employing the a) cc-pVTZ; b) pc-2; and c) def2-TZVP triple-ζ basis sets at the HF level.

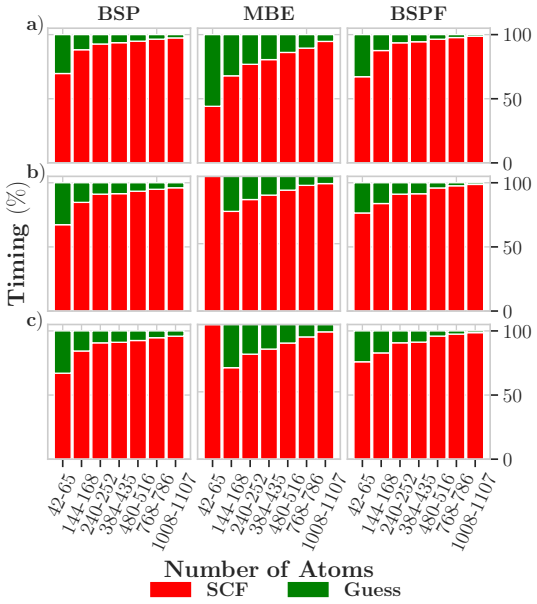


Figure B.55: Average timing breakdown of the BSP, MBE, BSPF initial guess approaches with varying system sizes employing the a) cc-pVDZ; b) pc-1; and c) def2-SVP double-ζ basis sets at the B3LYP level.

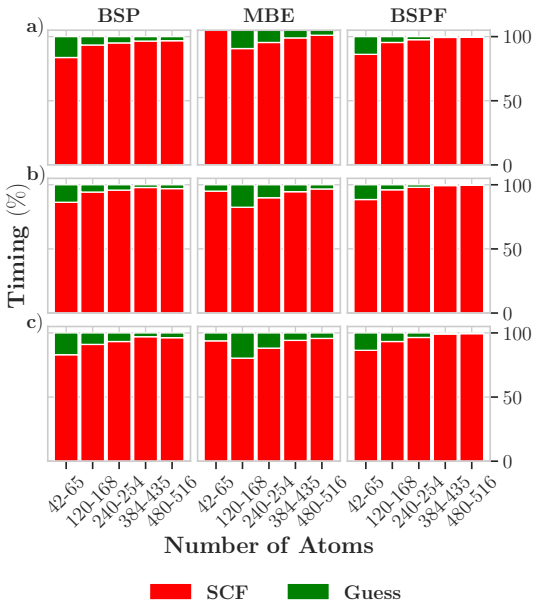


Figure B.56: Average timing breakdown of the BSP, MBE, BSPF initial guess approaches with varying system sizes employing the a) cc-pVTZ; b) pc-2; and c) def2-TZVP triple-ζ basis sets at the B3LYP level.

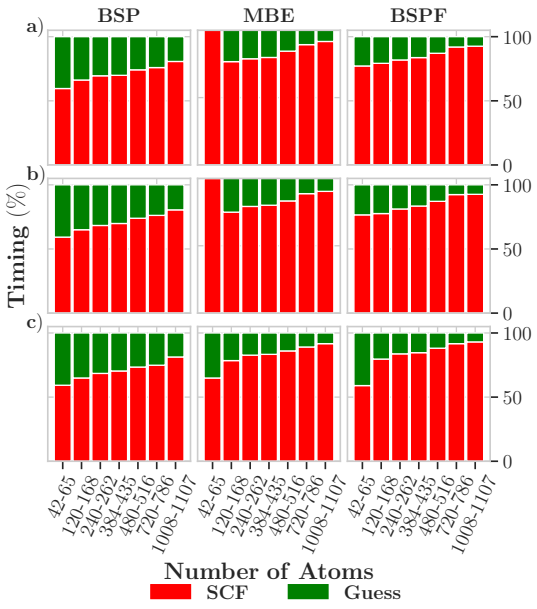


Figure B.57: Average timing breakdown of the BSP, MBE, BSPF initial guess approaches with varying system sizes employing the a) cc-pVDZ; b) pc-1; and c) def2-SVP double-ζ basis sets at the MN15 level.

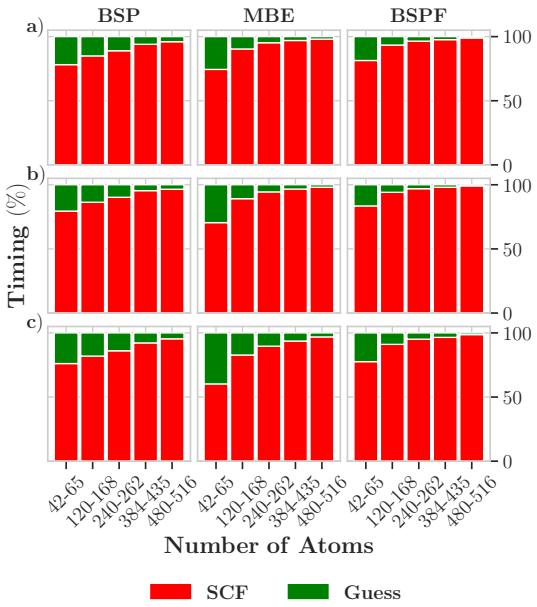


Figure B.58: Average timing breakdown of the BSP, MBE, BSPF initial guess approaches with varying system sizes employing the a) cc-pVTZ; b) pc-2; and c) def2-TZVP triple-ζ basis sets at the MN15 level.

Appendix C

Supporting Material for Chapter 5

Geometries of protein-ligand structures, representative input files for DelPhi and AMBER, calculated solvation energies and timing data are provided in the GitHub repository `https://github.com/fionacyu/pbe_solver`.

Comparing dielectric mapping schemes in AMBER

In AMBER's *pbsa.cuda* the ipb parameter controls the dielectric mapping scheme used. We compare only $ipb = 1$ and $ipb = 2$ and exclude the other options due to them being non-compatible with the molecular surface treatment chosen (solvent excluded surface) or the boundary condition (default option which uses all grid charges to compute boundary grid potentials). $ipb = 1$ corresponds to the same geometric approach DelPhi uses. $ipb = 2$ is the level set function-based dielectric mapping approach. We perform the comparison on two target protein systems: the smallest (MCL1 with 2448 atoms) and largest (CDK8 with 10456 atoms).

Tables C.1 and C.3 display the ΔG_{solv} values calculated using the different dielectric mapping schemes on the MCL1 and CDK8 systems, respectively. As observed from these tables, the difference in their solvation energies are negligible; MAEs of 0.16 and 0.12 kJ mol^{-1} are attained between $ipb = 1$ and $ipb = 2$ for the CDK8 and MCL1 systems, repectively.

Tables C.2 and C.4 showcase the wall times using the different electric mapping schemes on the MCL1 and CDK8 systems, respectively. We observe that employing $ipb = 2$ reduces the wall time of both MCL1 and CDK8, though minimally for MCL1. On average, using $ipb = 2$ reduced wall times by 1.32 s and 19.84 s for MCL1 and CDK8 systems, respectively.

Therefore, in light of the negligible difference in the ΔG_{solv} values calculated and $ipb = 2$ reducing the wall times of associated calculations, we selected $ipb = 2$ as the dielectric mapping scheme used for all AMBER *pbsa.cuda* calculations in this study.

Ligand	$ipb = 1$	$ipb = 2$
lig_27	594.94	594.78
lig_28	503.14	503.20
lig_30	564.15	564.09
lig_31	501.73	501.64
lig_32	506.91	506.60
lig_33	480.91	480.82
lig_34	472.72	472.63
lig_35	510.93	510.80
lig_36	547.32	547.47
lig_37	569.61	569.59
lig_43	560.09	560.07
lig_46	513.93	513.98
lig_47	555.19	555.03
lig_48	592.84	592.84
lig_49	554.82	554.58
lig_50	550.51	550.50
lig_52	539.22	539.09
lig_53	487.22	487.03
lig_56	604.94	604.98
lig_58	524.90	524.85
lig_60	529.01	528.82
lig_61	499.80	499.41
lig_63	492.82	492.71
lig_65	496.37	496.25
lig_67	548.72	548.72

Table C.1: Calculated ΔG_{solv} (kJ mol^{-1}) values using $ipb = 1$ and $ipb = 2$ on the MCL1 systems.

Ligand	$ipb = 1$	$ipb = 2$
lig_27	64.84	63.84
lig_28	63.53	62.21
lig_30	62.11	61.61
lig_31	61.14	60.27
lig_32	65.25	64.48
lig_33	65.24	64.39
lig_34	69.56	68.93
lig_35	64.46	63.24
lig_36	64.26	63.36
lig_37	69.28	68.69
lig_43	67.09	66.09
lig_46	64.72	63.87
lig_47	69.70	68.38
lig_48	61.44	60.55
lig_49	63.33	63.07
lig_50	69.43	68.28
lig_52	69.64	68.38
lig_53	68.79	67.98
lig_56	67.08	66.56
lig_58	64.15	63.50
lig_60	64.28	63.58
lig_61	64.61	64.34
lig_63	65.34	64.38
lig_65	69.37	68.75
lig_67	66.67	65.97

Table C.2: Wall times (s) of AMBER *pbsa.cuda* calculations using $ipb = 1$ and $ipb = 2$ on the MCL1 systems.

Ligand	$ipb = 1$	$ipb = 2$
lig_13	185.22	184.97
lig_14	107.78	107.87
lig_15	145.85	145.75
lig_16	112.19	112.12
lig_17	111.36	111.14
lig_18	154.50	154.58
lig_19	140.39	140.27
lig_20	170.03	169.70
lig_21	157.17	156.94
lig_22	141.59	141.54
lig_24	137.91	138.04
lig_25	118.73	118.88
lig_26	165.21	165.04
lig_27	193.10	192.75
lig_28	127.46	127.32
lig_29	198.75	199.06
lig_30	224.57	224.39
lig_32	145.31	145.32
lig_33	127.19	127.00
lig_34	118.73	118.64
lig_35	178.89	178.47
lig_36	210.61	210.44
lig_37	185.04	184.61
lig_38	200.82	200.87
lig_39	154.00	153.99
lig_40	127.83	127.84
lig_41	201.41	201.46
lig_42	165.22	165.05
lig_43	166.57	166.36
lig_44	148.46	148.24
lig_45	186.91	186.81

Table C.3: Calculated ΔG_{solv} (kJ mol^{-1}) values using $ipb = 1$ and $ipb = 2$ on the CDK8 systems.

Ligand	$ipb = 1$	$ipb = 2$
lig_13	998.18	980.67
lig_14	947.46	931.53
lig_15	976.82	960.45
lig_16	1001.65	987.35
lig_17	997.03	977.19
lig_18	999.42	982.51
lig_19	1024.57	1008.13
lig_20	977.10	961.47
lig_21	1004.63	986.41
lig_22	945.33	927.58
lig_24	976.25	958.14
lig_25	930.43	912.69
lig_26	988.37	970.38
lig_27	967.45	951.69
lig_28	983.92	968.81
lig_29	1002.11	984.94
lig_30	1006.02	988.96
lig_32	980.81	961.69
lig_33	1027.73	1009.32
lig_34	945.60	929.78
lig_35	950.72	932.62
lig_36	967.78	953.49
lig_37	962.79	946.64
lig_38	958.75	941.26
lig_39	1012.02	997.39
lig_40	993.65	977.09
lig_41	973.95	968.67
lig_42	1012.02	1004.47
lig_43	955.80	952.03
lig_44	963.27	943.74
lig_45	986.85	968.89

Table C.4: Wall times (s) of AMBER *pbsa.cuda* calculations using $ipb = 1$ and $ipb = 2$ on the CDK8 systems.

Results

Solvation Energies

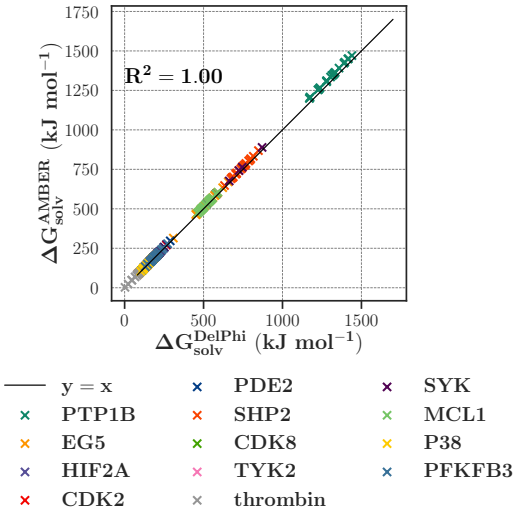


Figure C.1: Correlation between solvation free energies of protein-ligand systems attained with DelPhi and AMBER.

Profiling of AMBER's pbsa.cuda

We profiled AMBER's *pbsa.cuda* software using Linaro Forge's MAP on the MCL1 protein system on a GPU node of the Perlmutter supercomputer which comprises NVIDIA A100 40 GB GPUs and a AMD EPYC 7763 CPU.

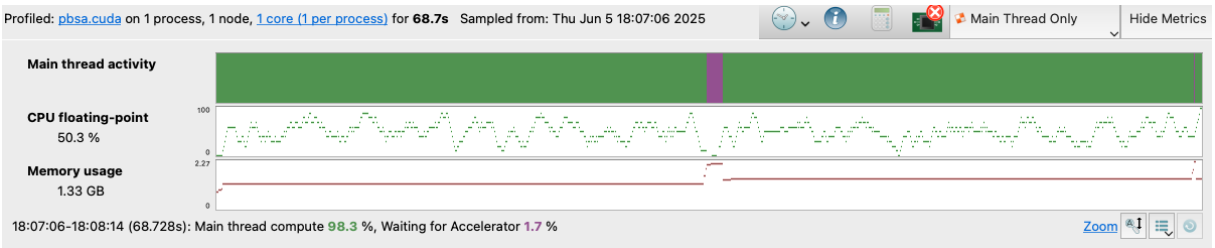


Figure C.2: Timeline of main thread activity, CPU floating point executions and memory usage of AMBER's *pbsa.cuda*. Purple and green sections in Main Thread Activity correspond to the GPU and CPU portions, respectively.

Figures C.2 and C.3 together reveal that the GPU portion constitutes a very small portion of the runtime. Specifically, only 1.7% (1.16 s) of the time is spent on the GPU and this timing of 1.16 s is much more consistent with the timings reported by Qi *et al.*[12]

Furthermore, Fig. C.3 not only highlights that the main bottleneck originates from routines run on the CPU, but that 31.2% of the time is spent performing expf64 op-

erations. These expf64 operations arise from calculating the boundary values on the lattice, which are performed on a single core on the CPU.

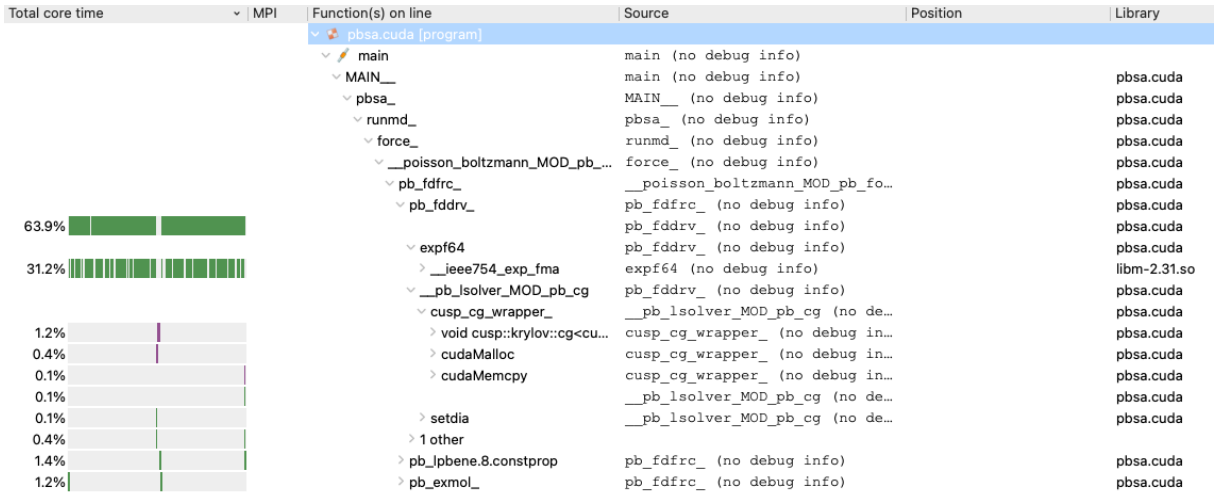


Figure C.3: Main thread stack of AMBER's *pbsa.cuda*.

Appendix D

Supporting Material for Chapter 6

Geometries of protein-ligand structures used for both PBSA and SPE calculations, partial charges of protein-ligand structures used for PBSA calculations, output gas phase and solvation energies are provided in the GitHub repository `https://github.com/fionacyu/qmpbsa_benchmark`.

Table D.1: Sampling times obtained with dihedral ACFs for each protein-ligand complex and the corresponding number of frames extracted.

Target	Ligand	Sampling time (ps)	Number of frames
cdk2	lig_17	260	39
cdk2	lig_1h1q	330	31
cdk2	lig_1h1r	230	44
cdk2	lig_1h1s	250	41
cdk2	lig_1oi9	280	36
cdk2	lig_1oiu	240	42
cdk2	lig_1oiy	390	26
cdk2	lig_20	180	56
cdk2	lig_21	130	77
cdk2	lig_22	150	67
cdk2	lig_26	310	33
cdk2	lig_28	200	51
cdk2	lig_29	240	42
cdk2	lig_30	350	29
cdk2	lig_31	330	31
cdk2	lig_32	450	23
mcl1	lig_23	220	46
mcl1	lig_26	260	39
mcl1	lig_27	190	53

Target	Ligand	Sampling time (ps)	Number of frames
mcl1	lig_28	250	41
mcl1	lig_29	210	48
mcl1	lig_30	280	36
mcl1	lig_31	220	46
mcl1	lig_32	280	36
mcl1	lig_33	290	35
mcl1	lig_34	360	28
mcl1	lig_35	250	41
mcl1	lig_36	210	48
mcl1	lig_37	210	48
mcl1	lig_38	230	44
mcl1	lig_39	160	63
mcl1	lig_40	290	35
mcl1	lig_41	350	29
mcl1	lig_42	210	48
mcl1	lig_43	200	51
mcl1	lig_44	490	21
mcl1	lig_45	180	56
mcl1	lig_46	370	28
mcl1	lig_47	280	36
mcl1	lig_48	130	77
mcl1	lig_49	350	29
mcl1	lig_50	290	35
mcl1	lig_51	260	39
mcl1	lig_52	280	36
mcl1	lig_53	160	63
mcl1	lig_54	190	53
mcl1	lig_56	250	41
mcl1	lig_57	150	67
mcl1	lig_58	260	39
mcl1	lig_60	180	56
mcl1	lig_61	170	59
mcl1	lig_62	170	59
mcl1	lig_63	350	29
mcl1	lig_64	240	42
mcl1	lig_65	200	51
mcl1	lig_66	170	59
mcl1	lig_67	340	30
mcl1	lig_68	280	36
jnk1	lig_17124	280	36

Target	Ligand	Sampling time (ps)	Number of frames
jnk1	lig_18624	340	30
jnk1	lig_18625	140	72
jnk1	lig_18626	200	51
jnk1	lig_18627	290	35
jnk1	lig_18628	150	67
jnk1	lig_18629	180	56
jnk1	lig_18630	480	21
jnk1	lig_18631	240	42
jnk1	lig_18632	310	33
jnk1	lig_18633	360	28
jnk1	lig_18634	180	56
jnk1	lig_18635	340	30
jnk1	lig_18636	140	72
jnk1	lig_18637	460	22
jnk1	lig_18638	430	24
jnk1	lig_18639	240	42
jnk1	lig_18652	150	67
jnk1	lig_18658	170	59
jnk1	lig_18659	300	34
jnk1	lig_18660	210	48
bace	lig_13a	190	53
bace	lig_13b	240	42
bace	lig_13c	340	30
bace	lig_13d	180	56
bace	lig_13e	100	101
bace	lig_13f	170	59
bace	lig_13g	210	48
bace	lig_13h	310	33
bace	lig_13i	180	56
bace	lig_13j	200	51
bace	lig_13k	640	16
bace	lig_13m	510	20
bace	lig_13n	200	51
bace	lig_13o	230	44
bace	lig_17a	1350	8
bace	lig_17b	60	167
bace	lig_17c	140	72
bace	lig_17d	180	56
bace	lig_17e	170	59
bace	lig_17f	200	51

Target	Ligand	Sampling time (ps)	Number of frames
bace	lig_17g	260	39
bace	lig_17h	690	15
bace	lig_17i	430	24
bace	lig_24	500	21
bace	lig_4a	90	112
bace	lig_4b	80	126
bace	lig_4c	180	56
bace	lig_4d	480	21
bace	lig_4i	170	59
bace	lig_4j	180	56
bace	lig_4k	170	59
bace	lig_4l	180	56
bace	lig_4m	210	48
bace	lig_4n	420	24
bace	lig_4o	430	24
bace	lig_4p	130	77
p38	lig_2aa	210	48
p38	lig_2bb	190	53
p38	lig_2c	290	35
p38	lig_2e	250	41
p38	lig_2ee	320	32
p38	lig_2f	230	44
p38	lig_2ff	220	46
p38	lig_2g	270	38
p38	lig_2gg	210	48
p38	lig_2h	220	46
p38	lig_2i	310	33
p38	lig_2j	230	44
p38	lig_2k	280	36
p38	lig_2l	200	51
p38	lig_2m	190	53
p38	lig_2n	160	63
p38	lig_2o	290	35
p38	lig_2p	250	41
p38	lig_2q	220	46
p38	lig_2r	170	59
p38	lig_2s	280	36
p38	lig_2t	250	41
p38	lig_2u	280	36
p38	lig_2v	180	56

Target	Ligand	Sampling time (ps)	Number of frames
p38	lig_2x	190	53
p38	lig_2y	130	77
p38	lig_2z	230	44
p38	lig_3fln	290	35
p38	lig_3flq	210	48
p38	lig_3flw	300	34
p38	lig_3fly	200	51
p38	lig_3flz	250	41
p38	lig_3fmh	240	42
p38	lig_3fmk	200	51
ptp1b	lig_20667	160	63
ptp1b	lig_20669	200	51
ptp1b	lig_20670	260	39
ptp1b	lig_23330	260	39
ptp1b	lig_23466	320	32
ptp1b	lig_23467	160	63
ptp1b	lig_23468	220	46
ptp1b	lig_23469	150	67
ptp1b	lig_23470	200	51
ptp1b	lig_23471	310	33
ptp1b	lig_23472	120	84
ptp1b	lig_23473	330	31
ptp1b	lig_23474	120	84
ptp1b	lig_23475	310	33
ptp1b	lig_23476	170	59
ptp1b	lig_23477	190	53
ptp1b	lig_23479	100	101
ptp1b	lig_23480	190	53
ptp1b	lig_23482	220	46
ptp1b	lig_23483	150	67
ptp1b	lig_23484	160	63
ptp1b	lig_23485	120	84
ptp1b	lig_23486	210	48
thrombin	lig_1a	310	33
thrombin	lig_1b	520	20
thrombin	lig_1c	370	28
thrombin	lig_3a	270	38
thrombin	lig_3b	490	21
thrombin	lig_5	150	67
thrombin	lig_6a	460	22

Target	Ligand	Sampling time (ps)	Number of frames
thrombin	lig_6b	440	23
thrombin	lig_6e	290	35
thrombin	lig_7a	200	51
tyk2	lig_ejm_31	430	24
tyk2	lig_ejm_42	280	36
tyk2	lig_ejm_43	550	19
tyk2	lig_ejm_44	510	20
tyk2	lig_ejm_45	260	39
tyk2	lig_ejm_46	310	33
tyk2	lig_ejm_47	300	34
tyk2	lig_ejm_48	310	33
tyk2	lig_ejm_49	650	16
tyk2	lig_ejm_50	280	36
tyk2	lig_ejm_54	400	26
tyk2	lig_ejm_55	260	39
tyk2	lig_jmc_23	530	19
tyk2	lig_jmc_27	470	22
tyk2	lig_jmc_28	820	13
tyk2	lig_jmc_30	600	17

	MBE2		MBE3	
Target	full	pocket	full	pocket
BACE	0.432	0.406	0.324	0.281
CDK2	0.752	0.762	0.769	0.773
JNK1	0.662	0.649	0.657	0.644
MCL1	0.480	0.444	0.480	0.423
P38	0.617	0.610	0.581	0.567
PTP1B	0.515	0.566	0.518	0.581
thrombin	0.551	0.568	0.575	0.562
TYK2	0.648	0.615	0.655	0.620
Average	0.582	0.578	0.570	0.556

Table D.2: Pearson correlation coefficients of MBE2 and MBE3 gas phase interaction energies computed using either the full protein or pocket residues for the two-body calculations with experimental binding free energies on the Wang dataset. Note all three-body calculations involve the pocket residues only.

Target	Distance (Å)
BACE	14.0
CDK2	15.2
JNK1	13.3
MCL1	15.5
P38	14.5
PTP1B	17.8
thrombin	12.4
TYK2	13.2

Table D.3: Maximum distance between pocket protein residues and ligands observed across all targets. Distance is calculated as the minimum atomic pair distance between a pocket protein residue and a ligand.

Target	including nonpolar	excluding nonpolar
BACE	0.42	0.44
CDK2	-0.19	-0.22
JNK1	0.32	0.00
MCL1	0.47	0.41
P38	0.77	0.71
PTP1B	0.33	0.23
thrombin	0.57	0.31
TYK2	0.75	0.75
Average	0.43	0.33

Table D.4: Pearson correlation coefficients obtained with MBE/COSMO2 including and excluding the nonpolar solvation component.

Target	MM/GBSA	VM2	SQM
BACE	-0.4	0.67	0.54
CDK2	-0.53	0.88	0.77
JNK1	0.65	0.67	0.49
MCL1	0.42	0.57	0.76
P38	0.66	0.37	0.54
PTP1B	0.67	-0.11	0.73
thrombin	0.93	0.74	0.81
TYK22	0.79	0.6	0.74

Table D.5: Pearson correlation coefficients obtained using different end point methods (MM/GBSA, VM2 and SQM). Data listed here are taken directly from Refs. [55] (MM/GBSA), [317] (VM2), [283] (SQM).

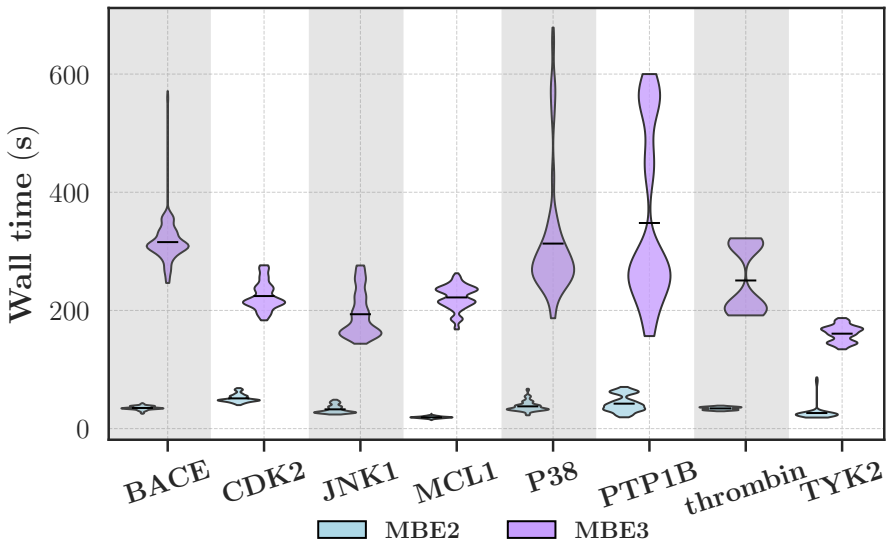


Figure D.1: Distribution of wall times of MBE calculations. Wall times are attained across 128 AMD MI250X GPUs.

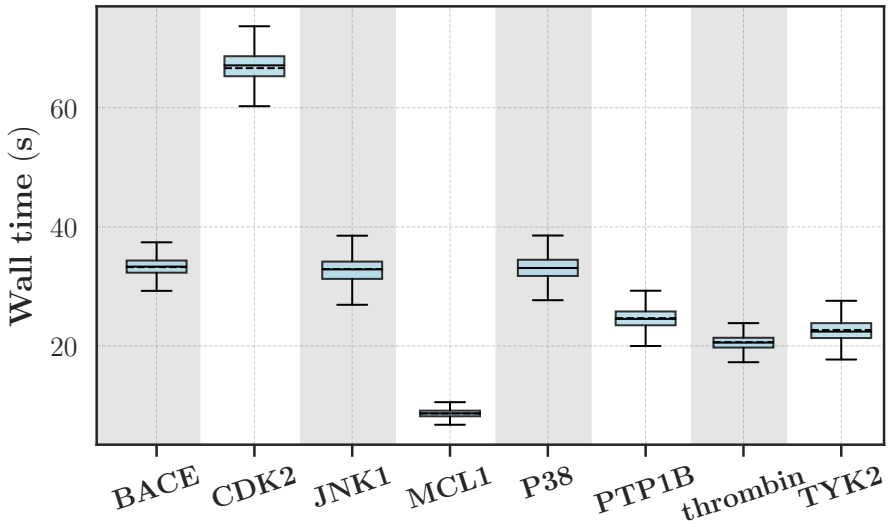


Figure D.2: Distribution of wall times of PBSA calculations. Wall times are attained on a single NVIDIA A100 80 GB GPU.

Bibliography

[1] R. Dennard, F. Gaensslen, H.-N. Yu, V. Rideout, E. Bassous, A. LeBlanc, *IEEE Journal of Solid-State Circuits* **1974**, *9*, 256–268.

[2] G. E. Moore, Cramming More Components onto Integrated Circuits, Electronics Magazine, 38, 114–117, Magazine article, **1965**.

[3] H. Sutter et al., *Dr. Dobb's journal* **2005**, *30*, 202–210.

[4] A. de Ruiter, D. Petrov, C. Oostenbrink, *Journal of Chemical Theory and Computation* **2020**, *17*, 56–65.

[5] S. Maier, B. Thapa, J. Erickson, K. Raghavachari, *Physical Chemistry Chemical Physics* **2022**, *24*, 14525–14537.

[6] Y. Meng, D. Sabri Dashti, A. E. Roitberg, *Journal of chemical theory and computation* **2011**, *7*, 2721–2727.

[7] B. Thapa, D. Beckett, J. Erickson, K. Raghavachari, *Journal of chemical theory and computation* **2018**, *14*, 5143–5155.

[8] A. Szabo, N. S. Ostlund, *Modern quantum chemistry: introduction to advanced electronic structure theory*, Dover Publications, Mineola, NY, **1996**.

[9] W. Rocchia, S. Sridharan, A. Nicholls, E. Alexov, A. Chiabrera, B. Honig, *Journal of Computational Chemistry* **2002**, *23*, 128–137.

[10] J. Wang, Q. Cai, Y. Xiang, R. Luo, *Journal of Chemical Theory and Computation* **2012**, *8*, 2741–2751.

[11] A. Li, *Journal of Theoretical and Computational Chemistry* **2014**, *13*, 1450040.

[12] R. Qi, W. M. Botello-Smith, R. Luo, *Journal of Chemical Theory and Computation* **2017**, *13*, 3378–3387.

[13] R. Qi, R. Luo, *Journal of Chemical Information and Modeling* **2019**, *59*, 409–420.

[14] J. L. Gustafson, *Communications of the ACM* **1988**, *31*, 532–533.

[15] A. Rizzi, D. Mandelli, *Expert Opinion on Drug Discovery* **2025**, *20*, 391–400.

[16] T.-S. Lee, Y. Hu, B. Sherborne, Z. Guo, D. M. York, *Journal of chemical theory and computation* **2017**, *13*, 3077–3084.

[17] T.-S. Lee, D. S. Cerutti, D. Mermelstein, C. Lin, S. LeGrand, T. J. Giese, A. Roitberg, D. A. Case, R. C. Walker, D. M. York, *Journal of chemical information and modeling* **2018**, *58*, 2043–2050.

[18] X. He, S. Liu, T.-S. Lee, B. Ji, V. H. Man, D. M. York, J. Wang, *ACS omega* **2020**, *5*, 4611–4619.

[19] H. Chen, J. D. Maia, B. K. Radak, D. J. Hardy, W. Cai, C. Chipot, E. Tajkhorshid, *Journal of chemical information and modeling* **2020**, *60*, 5301–5307.

[20] R. Qian, J. Xue, Y. Xu, J. Huang, *Journal of Chemical Information and Modeling* **2024**, *64*, 7214–7237.

[21] S. Seritan, C. Bannwarth, B. S. Fales, E. G. Hohenstein, C. M. Isborn, S. I. Kokkila-Schumacher, X. Li, F. Liu, N. Luehr, J. W. Snyder Jr, et al., *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2021**, *11*, e1494.

[22] G. M. J. Barca, D. L. Poole, J. L. G. Vallejo, M. Alkan, C. Bertoni, A. P. Rendell, M. S. Gordon in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, **2020**, pp. 1–14.

[23] G. M. Barca, J. L. G. Vallejo, D. L. Poole, M. Alkan, R. Stocks, A. P. Rendell, M. S. Gordon in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, **2021**, pp. 1–15.

[24] G. M. Barca, C. Snowdon, J. L. G. Vallejo, F. Kazemian, A. P. Rendell, M. S. Gordon in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, **2022**, pp. 1–14.

[25] J. L. Galvez Vallejo, G. M. Barca, M. S. Gordon, *Molecular Physics* **2023**, *121*, e2112987.

[26] J. L. Galvez Vallejo, C. Snowdon, R. Stocks, F. Kazemian, F. Chuo, Y. Yu, C. Seidl, Z. Seeger, M. Alkan, D. Poole, B. M. Westheimer, ; Mehaboob Basha, M. De, L. Pierre, A. Rendell, E. I. Izgorodina, M. S. Gordon, G. M. J. Barca, M. Basha, *The Journal of Chemical Physics* **2023**, *159*, 44112.

[27] R. Stocks, E. Palethorpe, G. M. J. Barca, *Journal of Chemical Theory and Computation* **2024**, *20*, PMID: 38456899, 2505–2519.

[28] E. Palethorpe, R. Stocks, G. M. J. Barca, Advanced Techniques for High-Performance Fock Matrix Construction on GPU Clusters, **2024**.

[29] R. Stocks, J. L. G. Vallejo, F. C. Y. Yu, C. Snowdon, E. Palethorpe, J. Kurzak, D. Bykov, G. M. J. Barca in SC24: International Conference for High Performance Computing, Networking, Storage and Analysis, **2024**, pp. 1–12.

[30] X. Wu, Q. Sun, Z. Pu, T. Zheng, W. Ma, W. Yan, Y. Xia, Z. Wu, M. Huo, X. Li, et al., *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2025**, *15*, e70008.

[31] C. Seidl, G. M. Barca, *Journal of Chemical Theory and Computation* **2022**, *18*, 4164–4176.

[32] E. Palethorpe, G. M. Barca, *Journal of Chemical Theory and Computation* **2025**, *21*, 9388–9403.

[33] F. C. Yu, J. L. G. Vallejo, G. M. Barca, *Journal of Cheminformatics* **2024**, *16*, 102.

[34] J. L. Galvez Vallejo, C. Snowdon, R. Stocks, F. Kazemian, F. C. Yan Yu, C. Seidl, Z. Seeger, M. Alkan, D. Poole, B. M. Westheimer, et al., *The Journal of Chemical Physics* **2023**, *159*.

[35] C. Li, M. Petukh, L. Li, E. Alexov, *Journal of computational chemistry* **2013**, *34*, 1949–1960.

[36] C. Li, L. Li, J. Zhang, E. Alexov, *Journal of computational chemistry* **2012**, *33*, 1960–1966.

[37] L. Li, C. Li, S. Sarkar, J. Zhang, S. Witham, Z. Zhang, L. Wang, N. Smith, M. Petukh, E. Alexov, *BMC biophysics* **2012**, *5*, 1–11.

[38] C. Li, Z. Jia, A. Chakravorty, S. Pahari, Y. Peng, S. Basu, M. Koirala, S. K. Panday, M. Petukh, L. Li, et al., *Journal of computational chemistry* **2019**, *40*, 2502–2508.

[39] A. Nicholls, B. Honig, *Journal of Computational Chemistry* **1991**, *12*, 435–445.

[40] R. Qi, R. Luo, *Journal of chemical information and modeling* **2018**, *59*, 409–420.

[41] W. Geng, F. Jacob, *Computer Physics Communications* **2013**, *184*, 1490–1496.

[42] J. Colmenares, J. Ortiz, W. Rocchia, *Bioinformatics* **2014**, *30*, 569–570.

[43] J. Colmenares, A. Galizia, J. Ortiz, A. Clematis, W. Rocchia, *BioMed Research International* **2014**, *2014*, 560987.

[44] E. Jurrus, D. Engel, K. Star, K. Monson, J. Brandi, L. E. Felberg, D. H. Brookes, L. Wilson, J. Chen, K. Liles, et al., *Protein Science* **2018**, *27*, 112–128.

[45] D. Petrov, *Journal of Chemical Information and Modeling* **2021**, *61*, 4382–4390.

[46] E. E. Guest, L. F. Cervantes, S. D. Pickett, C. L. Brooks III, J. D. Hirst, *Journal of Chemical Information and Modeling* **2022**, *62*, 1458–1470.

[47] D. M. York, *ACS Physical Chemistry Au* **2023**, *3*, 478–491.

[48] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, T. E. Ferrin, *Journal of molecular biology* **1982**, *161*, 269–288.

[49] S. Skariyachan, S. Garka in *Fullerens, Graphenes and Nanotubes*, (Ed.: A. M. Grumezescu), William Andrew Publishing, **2018**, pp. 1–29.

[50] W. Xu, A. J. Lucke, D. P. Fairlie, *Journal of Molecular Graphics and Modelling* **2015**, *57*, 76–88.

[51] P. Ferrara, H. Gohlke, D. J. Price, G. Klebe, C. L. Brooks, *Journal of medicinal chemistry* **2004**, *47*, 3032–3047.

[52] Z. Wang, H. Sun, X. Yao, D. Li, L. Xu, Y. Li, S. Tian, T. Hou, *Physical Chemistry Chemical Physics* **2016**, *18*, 12964–12975.

[53] G. L. Warren, C. W. Andrews, A.-M. Capelli, B. Clarke, J. LaLonde, M. H. Lambert, M. Lindvall, N. Nevins, S. F. Semus, S. Senger, et al., *Journal of medicinal chemistry* **2006**, *49*, 5912–5931.

[54] K. J. Cutrona, A. S. Newton, S. G. Krimmer, J. Tirado-Rives, W. L. Jorgensen, *Journal of chemical information and modeling* **2020**, *60*, 4403–4415.

[55] L. Wang, Y. Wu, Y. Deng, B. Kim, L. Pierce, G. Krilov, D. Lupyan, S. Robinson, M. K. Dahlgren, J. Greenwood, et al., *Journal of the American Chemical Society* **2015**, *137*, 2695–2703.

[56] W. Jespers, M. Esguerra, J. Åqvist, H. Gutiérrez-de-Terán, *Journal of cheminformatics* **2019**, *11*, 26.

[57] L. Carvalho Martins, E. A. Cino, R. S. Ferreira, *Journal of Chemical Theory and Computation* **2021**, *17*, 4262–4273.

[58] H. H. Loeffler, J. Michel, C. Woods, *Journal of Chemical Information and Modeling* **2015**, *55*, PMID: 26544598, 2485–2490.

[59] E. Wang, H. Sun, J. Wang, Z. Wang, H. Liu, J. Z. H. Zhang, T. Hou, *Chemical Reviews* **2019**, *119*, 9478–9508.

[60] K. Furui, M. Ohue, *The Journal of Supercomputing* **2024**, 1–16.

[61] J. E. Crivelli-Decker, Z. Beckwith, G. Tom, L. Le, S. Khuttan, R. Salomon-Ferrer, J. Beall, R. Gómez-Bombarelli, A. Bortolato, *Journal of Chemical Theory and Computation* **2024**, *20*, 7188–7198.

[62] S. J. Fox, J. Dziedzic, T. Fox, C. S. Tautermann, C.-K. Skylaris, *Proteins: Structure Function and Bioinformatics* **2014**, *82*, 3335–3346.

[63] S. K. Mishra, J. Koča, *The Journal of Physical Chemistry B* **2018**, *122*, 8113–8121.

[64] J. Liu, X. Wang, J. Z. Zhang, X. He, *Rsc Advances* **2015**, *5*, 107020–107030.

[65] Y. Zhang, W. Xia, J. Xiao, J. Z. Zhang, *Journal of chemical theory and computation* **2025**, *21*, 2129–2139.

[66] L. Gundelach, T. Fox, C. S. Tautermann, C.-K. Skylaris, *Physical Chemistry Chemical Physics* **2022**, *24*, 25240–25249.

[67] Z. Yuan, X. Chen, S. Fan, L. Chang, L. Chu, Y. Zhang, J. Wang, S. Li, J. Xie, J. Hu, et al., *International Journal of Molecular Sciences* **2024**, *25*, 671.

[68] N. Okimoto, T. Otsuka, Y. Hirano, M. Taiji, *ACS omega* **2018**, *3*, 4475–4485.

[69] P. Soderhjelm, J. Kongsted, U. Ryde, *Journal of Chemical Theory and Computation* **2010**, *6*, 1726–1737.

[70] P. A. M. Dirac, *CUDA C++ Programming Guide*, NVIDIA, **2021**.

[71] T. M. John Cheng, Max Grossman, *Professional CUDA C Programming*, John Wiley & Sons, Inc., **2014**.

[72] H. Jun, J. Cho, K. Lee, H.-Y. Son, K. Kim, H. Jin, K. Kim in 2017 IEEE International Memory Workshop (IMW), IEEE, **2017**, pp. 1–4.

[73] *OpenMP application program interface 5.0*, OpenMP Architecture Review Board, **2018**.

[74] L. Chen, O. Villa, S. Krishnamoorthy, G. R. Gao, *Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing IPDPS 2010* **2010**, DOI 10.1109/IPDPS.2010.5470413.

[75] M. Karplus, G. A. Petsko, *Nature* **1990**, *347*, 631–639.

[76] S. A. Adcock, J. A. McCammon, *Chemical reviews* **2006**, *106*, 1589–1615.

[77] J. A. Lemkul, J. Huang, B. Roux, A. D. MacKerell Jr, *Chemical reviews* **2016**, *116*, 4983–5013.

[78] A. Perera, R. Mazighi, *The Journal of Chemical Physics* **2015**, *143*.

[79] J. Meller et al., *Encyclopedia of life sciences* **2001**, *18*.

[80] R. Schneider, A. R. Sharma, A. Rai in *Computational Many-Particle Physics*, Springer, **2008**, pp. 3–40.

[81] C. M. Baker, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2015**, *5*, 241–254.

[82] A. Warshel, M. Levitt, *Journal of Molecular Biology* **1976**, *103*, 227–249.

[83] Z. Jing, C. Liu, S. Y. Cheng, R. Qi, B. D. Walker, J.-P. Piquemal, P. Ren, *Annual Review of biophysics* **2019**, *48*, 371–394.

[84] X. He et al., *Biophysical Reviews* **2025**, 1–13.

[85] D. A. McQuarrie, J. D. Simon, *Physical chemistry: a molecular approach*, *Vol. 1*, University science books Sausalito, CA, **1997**.

[86] Z. Cournia, B. Allen, W. Sherman, *Journal of Chemical Information and Modeling* **2017**, *57*, PMID: 29243483, 2911–2937.

[87] R. A. Goodnow, *Drug Discovery Today: Technologies* **2006**, *3*, 367–375.

[88] T. P. Kenakin in *A Pharmacology Primer* (*Third Edition*), (Ed.: T. P. Kenakin), Academic Press, New York, **2009**, pp. 1–19.

[89] D. R. Silva, J. de Cássia Orlandi Sardi, I. A. Freires, A. C. B. Silva, P. L. Rosalen, *European Journal of Pharmacology* **2019**, *842*, 64–69.

[90] P. K. Deb, O. Al-Attraqchi, M. N. Al-Qattan, M. Raghu Prasad, R. K. Tekade in *Dosage Form Design Parameters*, (Ed.: R. K. Tekade), Advances in Pharmaceutical Product Development and Research, Academic Press, **2018**, pp. 665–703.

[91] R. Wang, Y. Lu, S. Wang, *Journal of Medicinal Chemistry* **2003**, *46*, PMID: 12773034, 2287–2303.

[92] I. A. Guedes, F. S. S. Pereira, L. E. Dardenne, *Frontiers in Pharmacology* **2018**, *Volume 9 - 2018*, DOI 10.3389/fphar.2018.01089.

[93] V. Salmaso, S. Moro, *Frontiers in Pharmacology* **2018**, *Volume 9 - 2018*, DOI 10.3389/fphar.2018.00923.

[94] S.-Y. Huang, X. Zou, *International Journal of Molecular Sciences* **2010**, *11*, 3016–3034.

[95] V. Salmaso, S. Moro, *Frontiers in pharmacology* **2018**, *9*, 923.

[96] A. Shirali, V. Stebliankin, U. Karki, J. Shi, P. Chapagain, G. Narasimhan, *BMC bioinformatics* **2025**, *26*, 25.

[97] K. J. Fujimoto, S. Minami, T. Yanai, *ACS omega* **2022**, *7*, 19030–19039.

[98] S.-Y. Huang, S. Z. Grinter, X. Zou, *Physical Chemistry Chemical Physics* **2010**, *12*, 12899–12908.

[99] R. W. Zwanzig, *The Journal of Chemical Physics* **1954**, *22*, 1420–1426.

[100] W. L. Jorgensen, C. Ravimohan, *The Journal of chemical physics* **1985**, *83*, 3050–3054.

[101] D. L. Beveridge, F. M. Dicapua, *Annual review of biophysics and biophysical chemistry* **1989**, *18*, 431–492.

[102] C. H. Bennett, *Journal of Computational Physics* **1976**, *22*, 245–268.

[103] A. de Ruiter, S. Boresch, C. Oostenbrink, *Journal of computational chemistry* **2013**, *34*, 1024–1034.

[104] V. A. Adediwura, K. Koirala, H. N. Do, J. Wang, Y. Miao, *Expert Opinion on Drug Discovery* **2024**, *19*, PMID: 38722032, 671–682.

[105] L. Wang, PhD thesis, The Australian National University, **2022**.

[106] Y. Matsunaga, M. Kamiya, H. Oshima, J. Jung, S. Ito, Y. Sugita, *Biophysical Reviews* **2022**, *14*, 1503–1512.

[107] A. P. Bhati, S. Wan, D. W. Wright, P. V. Coveney, *Journal of chemical theory and computation* **2017**, *13*, 210–222.

[108] A. D. Wade, A. P. Bhati, S. Wan, P. V. Coveney, *Journal of Chemical Theory and Computation* **2022**, *18*, 3972–3987.

[109] H. Ioannidis, A. Drakopoulos, C. Tzitzoglaki, N. Homeyer, F. Kolarov, P. Gkeka, K. Freudenberger, C. Liolios, G. Gauglitz, Z. Cournia, et al., *Journal of Chemical Information and Modeling* **2016**, *56*, 862–876.

[110] V. Gapsys, L. Pérez-Benito, M. Aldeghi, D. Seeliger, H. Van Vlijmen, G. Tresadern, B. L. De Groot, *Chemical Science* **2020**, *11*, 1140–1152.

[111] Y. Hu, I. Muegge, *Journal of Chemical Information and Modeling* **2022**, *62*, 4448–4459.

[112] G. A. Ross, C. Lu, G. Scarabelli, S. K. Albanese, E. Houang, R. Abel, E. D. Harder, L. Wang, *Communications Chemistry* **2023**, *6*, 222.

[113] A. Ghysels, V. Van Speybroeck, E. Pauwels, S. Catak, B. R. Brooks, D. Van Neck, M. Waroquier, *Journal of computational chemistry* **2010**, *31*, 994–1007.

[114] S. Genheden, U. Ryde, *Expert opinion on drug discovery* **2015**, *10*, 449–461.

[115] C.-Y. Yang, H. Sun, J. Chen, Z. Nikolovska-Coleska, S. Wang, *Journal of the American Chemical Society* **2009**, *131*, 13709–13721.

[116] I. Massova, P. A. Kollman, *Journal of the American Chemical Society* **1999**, *121*, 8133–8143.

[117] N. Špačková, T. E. Cheatham, F. Ryjáček, F. Lankaš, L. Van Meervelt, P. Hobza, J. Šponer, *Journal of the American Chemical Society* **2003**, *125*, 1759–1769.

[118] M. Lepšík, Z. Kříž, Z. Havlas, *Proteins: Structure Function and Bioinformatics* **2004**, *57*, 279–293.

[119] N. H. March, W. H. Young, S. Sampanthar, *The many-body problem in quantum mechanics*, Courier Corporation, **1995**.

[120] M. Born, W. Heisenberg in *Original Scientific Papers Wissenschaftliche Originalarbeiten*, Springer, **1985**, pp. 216–246.

[121] D. R. Hartree in Mathematical Proceedings of the Cambridge Philosophical Society, *Vol. 24*, Cambridge university press, **1928**, pp. 89–110.

[122] J. C. Slater, *Physical review* **1951**, *81*, 385.

[123] P. W. Atkins, R. S. Friedman, *Molecular quantum mechanics*, Oxford university press, **2011**.

[124] P. M. Gill in *Advances in quantum chemistry*, *Vol. 25*, Elsevier, **1994**, pp. 141–205.

[125] C. C. J. Roothaan, *Reviews of modern physics* **1951**, *23*, 69.

[126] C. Roothaan, *Reviews of modern physics* **1960**, *32*, 179.

[127] H. Li, D. J. Yaron, *Journal of chemical theory and computation* **2016**, *12*, 5322–5332.

[128] I. Snook, M. C. Per, S. P. Russo, *The Journal of chemical physics* **2008**, *129*.

[129] F. Jensen, *Introduction to Computational Chemistry*, 2nd ed., John Wiley & Sons Ltd, Chichester, West Sussex PO19 8SQ, England, **2007**.

[130] C. Møller, M. S. Plesset, *Physical review* **1934**, *46*, 618.

[131] O. Vahtras, J. Almlöf, M. Feyereisen, *Chemical Physics Letters* **1993**, *213*, 514–518.

[132] O. Vahtras, J. Almlöf, M. W. Feyereisen, *Chemical Physics Letters* **1993**, *213*, 514–518.

[133] R. Stocks, J. L. G. Vallejo, C. Fiona, C. Snowdon, E. Palethorpe, J. Kurzak, D. Bykov, G. M. Barca in SC24: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, **2024**, pp. 1–12.

[134] U. Ryde, *Methods in enzymology* **2016**, *577*, 119–158.

[135] P. Hohenberg, W. Kohn, *Physical review* **1964**, *136*, B864.

[136] W. Kohn, L. J. Sham, *Physical review* **1965**, *140*, A1133.

[137] S. Y. Haoyu, X. He, S. L. Li, D. G. Truhlar, *Chemical science* **2016**, *7*, 5032–5051.

[138] A. D. Becke, *The Journal of Chemical Physics* **1993**, *98*, 5648–5652.

[139] C. Lee, W. Yang, R. G. Parr, *Physical review B* **1988**, *37*, 785.

[140] S. H. Vosko, L. Wilk, M. Nusair, *Canadian Journal of physics* **1980**, *58*, 1200–1211.

[141] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, M. J. Frisch, *The Journal of physical chemistry* **1994**, *98*, 11623–11627.

[142] R. M. Richard, K. U. Lao, J. M. Herbert, *Accounts of Chemical Research* **2014**, *47*, 2828–2836.

[143] N. J. Mayhall, K. Raghavachari, *Journal of Chemical Theory and Computation* **2012**, *8*, 2669–2675.

[144] M. A. Collins, R. P. Bettens, *Chemical Reviews* **2015**, *115*, 5607–5642.

[145] D. Schmitt-Monreal, C. R. Jacob, *Journal of Chemical Theory and Computation* **2021**, *17*, 4144–4156.

[146] J. N. M. Vitorino, Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2024-10-31, PhD thesis, **2022**, p. 87.

[147] K. A. Sharp, B. Honig, *Journal of Physical Chemistry* **1990**, *94*, 7684–7692.

[148] M. E. Davis, J. A. McCammon, *Journal of Computational Chemistry* **1989**, *10*, 386–391.

[149] P. I. Frazier in *Recent advances in optimization and modeling of contemporary problems*, Informs, **2018**, pp. 255–278.

[150] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.

[151] D. Kobler, *Encyclopedia of Optimization* **2008**, 950–959.

[152] J. Kalinowski, F. Wennmohs, F. Neese, *Journal of Chemical Theory and Computation* **2017**, *13*, 3160–3170.

[153] G. M. Barca, D. L. Poole, J. L. Vallejo, M. Alkan, C. Bertoni, A. P. Rendell, M. S. Gordon, *International Conference for High Performance Computing Networking Storage and Analysis SC* **2020**, *2020-November*, DOI 10.1109/SC41405.2020.00085.

[154] S. R. Gadre, R. N. Shirsat, A. C. Limaye, *J. Phys. Chem* **1994**, *98*, 9165–9169.

[155] S. S. Khire, N. Sahu, S. R. Gadre, *Computer Physics Communications* **2022**, *270*, 108175.

[156] V. Deev, M. A. Collins, *The Journal of Chemical Physics* **2005**, *122*, 154102.

[157] M. A. Collins, *Physical Chemistry Chemical Physics* **2012**, *14*, 7744–7751.

[158] M. A. Collins, *The Journal of Chemical Physics* **2014**, *141*, 094108.

[159] S. Hua, W. Hua, S. Li, *Journal of Physical Chemistry A* **2010**, *114*, 8126–8134.

[160] S. Hua, W. Li, S. Li, *ChemPhysChem* **2013**, *14*, 108–115.

[161] S. Li, W. Li, J. Ma, *Accounts of Chemical Research* **2014**, *47*, 2712–2720.

[162] K. Kitaura, E. Ikeo, T. Asada, T. Nakano, M. Uebayasi, *Chemical Physics Letters* **1999**, *313*, 701–706.

[163] D. G. Fedorov, K. Ishimura, T. Ishida, K. Kitaura, P. Pulay, S. Nagase, *Journal of Computational Chemistry* **2007**, *28*, 1476–1484.

[164] Y. Nishimoto, D. G. Fedorov, S. Irle, *Journal of Chemical Theory and Computation* **2014**, *10*, 4801–4812.

[165] R. Wang, X. Fang, Y. Lu, S. Wang, *Journal of Medicinal Chemistry* **2004**, *47*, 2977–2980.

[166] B. Cordero, V. Gómez, A. E. Platero-Prats, M. Revés, J. Echeverría, E. Cremades, F. Barragán, S. Alvarez, *Dalton Transactions* **2008**, 2832–2838.

[167] G. M. Barca, M. Alkan, J. L. Galvez-Vallejo, D. L. Poole, A. P. Rendell, M. S. Gordon, *Journal of Chemical Theory and Computation* **2021**, *17*, 7486–7503.

[168] R. Stocks, E. Palethorpe, G. M. Barca, *Journal of Chemical Theory and Computation* **2024**, *20*, 2505–2519.

[169] R. Stocks, E. Palethorpe, G. M. J. Barca, Multi-GPU RI-HF Energies and Analytic Gradients − Towards High Throughput Ab Initio Molecular Dynamics, **2024**.

[170] D. G. Fedorov, J. H. Jensen, R. C. Deka, K. Kitaura, *Journal of Physical Chemistry A* **2008**, *112*, 11808–11816.

[171] D. G. Fedorov, L. V. Slipchenko, K. Kitaura, *Journal of Physical Chemistry A* **2010**, *114*, 8742–8753.

[172] G. M. J. Barca, C. Bertoni, L. Carrington, D. Datta, N. De Silva, J. E. Deustua, D. G. Fedorov, J. R. Gour, A. O. Gunina, E. Guidez, T. Harville, S. Irle, J. Ivanic, K. Kowalski, S. S. Leang, H. Li, W. Li, J. J. Lutz, I. Magoulas, J. Mato, V. Mironov, H. Nakata, B. Q. Pham, P. Piecuch, D. Poole, S. R. Pruitt, A. P. Rendell, L. B. Roskop, K. Ruedenberg, T. Sattasathuchana, M. W. Schmidt, J. Shen, L. Slipchenko, M. Sosonkina, V. Sundriyal, A. Tiwari, J. L. Galvez Vallejo, B. Westheimer, M. Wloch, P. Xu, F. Zahariev, M. S. Gordon, *The Journal of Chemical Physics* **2020**, *152*, 154102.

[173] J. M. Amigó, J. Gálvez, V. M. Villar, *Naturwissenschaften* **2009**, *96*, 749–761.

[174] J. A. Bilbrey, J. P. Heindel, M. Schram, P. Bandyopadhyay, S. S. Xantheas, S. Choudhury, *Journal of Chemical Physics* **2020**, *153*, 16.

[175] K. Atz, F. Grisoni, G. Schneider, *Nature Machine Intelligence 2021 3:12* **2021**, *3*, 1023–1032.

[176] Russell D. Johnson III, NIST Computational Chemistry Comparison and Benchmark Database, `http://cccbdb.nist.gov/` Accessed 2 Jan 2024., **2022**.

[177] B. Milián-Medina, J. Gierschner, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2012**, *2*, 513–524.

[178] I. V. Alabugin, G. dos Passos Gomes, M. A. Abdo, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2019**, *9*, e1389.

[179] A. K. Rappé, C. J. Casewit, K. S. Colwell, W. A. Goddard, W. M. Skiff, *Journal of the American Chemical Society* **1992**, *114*, 10024–10035.

[180] M. G. Martin, *https://doi.org/10.1080/08927022.2013.828208* **2013**, *39*, 1212–1222.

[181] Á. Jász, Á. Rák, I. Ladjánszki, G. Cserey, *Journal of Molecular Structure* **2019**, *1188*, 227–233.

[182] L. M. McDowall, R. A. Dampney, *American Journal of Physiology-Heart and Circulatory Physiology* **2006**, *291*, H2003–H2007.

[183] C. M. Wolf, L. Guio, S. Scheiwiller, V. Pakhnyuk, C. Luscombe, L. D. Pozzo, *ACS Polymers Au* **2021**, *1*, 134–152.

[184] A. S. Christensen, T. Kubař, Q. Cui, M. Elstner, *Chemical Reviews* **2016**, *116*, 5301.

[185] J. A. Grant, B. T. Pickup, *Journal of Physical Chemistry* **1995**, *99*, 3503–3510.

[186] M. J. Vainio, J. S. Puranen, M. S. Johnson, *Journal of Chemical Information and Modeling* **2009**, *49*, 492–502.

[187] A. Shukla, H. M. Pandey, D. Mehrotra, *2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management ABLAZE 2015* **2015**, 515–519.

[188] J. L. Pachuau, A. Roy, A. Kumar Saha, *Smart Innovation Systems and Technologies* **2021**, *206*, 581–598.

[189] M. A. Collins, V. A. Deev, *The Journal of Chemical Physics* **2006**, *125*, 104104.

[190] E. I. Izgorodina, J. Rigby, D. R. Mac Farlane, *Chemical Communications* **2012**, *48*, 1493–1495.

[191] J. M. Herbert, *Journal of Chemical Physics* **2019**, *151*, DOI 10.1063/1.5126216.

[192] F. Ballesteros, K. U. Lao, *Journal of Chemical Theory and Computation* **2022**, *18*, 179–191.

[193] D. W. Zhang, X. H. Chen, J. Z. Zhang, *Journal of Computational Chemistry* **2003**, *24*, 1846–1852.

[194] M. Isegawa, B. Wang, D. G. Truhlar, *Journal of Chemical Theory and Computation* **2013**, *9*, 1381–1393.

[195] A. Saha, K. Raghavachari, *Journal of Chemical Theory and Computation* **2015**, *11*, 2012–2023.

[196] K. F. Y Mochizuki, S Tanaka, *Recent Advances of the Fragment Molecular Orbital Method Enhanced Performance and Applicability*, Springer Nature Singapore Pte Ltd, 152 Beach Road, #21-01/04 Gateway East, Singapore, **2021**.

[197] Y. Nishimoto, D. G. Fedorov, *Journal of Chemical Physics* **2018**, *148*, 64115.

[198] E. E. Dahlke, D. G. Truhlar, *Journal of Chemical Theory and Computation* **2007**, *3*, 46–53.

[199] X. Wang, J. Liu, J. Z. Zhang, X. He, *Journal of Physical Chemistry A* **2013**, *117*, 7149–7161.

[200] J. Almlöf, K. Faegri Jr., K. Korsell, *Journal of Computational Chemistry* **1982**, *3*, 385–399.

[201] J. H. Van Lenthe, R. Zwaans, H. J. Van Dam, M. F. Guest, *Journal of Computational Chemistry* **2006**, *27*, 926–932.

[202] X. He, K. M. Merz, *Journal of Chemical Theory and Computation* **2010**, *6*, 405–411.

[203] S. Lehtola, *Journal of Chemical Theory and Computation* **2019**, *15*, 1593–1604.

[204] Z. Szekeres, P. G. Mezey, P. R. Surján, *Chemical Physics Letters* **2006**, *424*, 420–424.

[205] B. Jansík, S. Høst, M. P. Johansson, J. Olsen, P. Jørgensen, T. Helgaker, *Physical Chemistry Chemical Physics* **2009**, *11*, 5805–5813.

[206] J. Zhang, A. L. Weisman, P. Saitta, R. A. Friesner, *International Journal of Quantum Chemistry* **2016**, *116*, 357–368.

[207] Z. Wang, W. Liu, *Journal of Chemical Theory and Computation* **2021**, *17*, PMID: 34240856, 4831–4845.

[208] B. Hégely, M. Kállay, *International Journal of Quantum Chemistry* **2022**, *122*, DOI 10.1002/QUA.26782.

[209] F. Ballesteros, J. A. Tan, K. U. Lao, *The Journal of chemical physics* **2023**, *159*, 74107.

[210] M. E. Mura, P. J. Knowles, *The Journal of Chemical Physics* **1996**, *104*, 9848–9858.

[211] S. Lehtola, C. Steigemann, M. J. Oliveira, M. A. Marques, *SoftwareX* **2018**, *7*, 1–5.

[212] D. Poole, D. B. Williams-Young, A. Jiang, Z. L. Glick, C. D. Sherrill, *The Journal of Chemical Physics* **2024**, *161*, 052503.

[213] T. H. Dunning Jr, *The Journal of chemical physics* **1989**, *90*, 1007–1023.

[214] D. E. Woon, T. H. Dunning Jr, *The Journal of chemical physics* **1993**, *98*, 1358–1371.

[215] F. Jensen, *The Journal of Chemical Physics* **2001**, *115*, 9113–9125.

[216] F. Jensen, *The Journal of chemical physics* **2002**, *116*, 7372–7379.

[217] F. Jensen, T. Helgaker, *The Journal of chemical physics* **2004**, *121*, 3463–3470.

[218] F. Jensen, *The Journal of Physical Chemistry A* **2007**, *111*, 11198–11204.

[219] A. Schäfer, C. Huber, R. Ahlrichs, *The Journal of Chemical Physics* **1994**, *100*, 5829–5835.

[220] F. Weigend, R. Ahlrichs, *Physical Chemistry Chemical Physics* **2005**, *7*, 3297–3305.

[221] P. Pulay, *Chemical Physics Letters* **1980**, *73*, 393–398.

[222] P. Pulay, *Journal of Computational Chemistry* **1982**, *3*, 556–560.

[223] E. Epifanovsky, A. T. B. Gilbert, X. Feng, J. Lee, Y. Mao, N. Mardirossian, P. Pokhilko, A. F. White, M. P. Coons, A. L. Dempwolff, Z. Gan, D. Hait, P. R. Horn, L. D. Jacobson, I. Kaliman, J. Kussmann, A. W. Lange, K. U. Lao, D. S. Levine, J. Liu, S. C. McKenzie, A. F. Morrison, K. D. Nanda, F. Plasser, D. R. Rehn, M. L. Vidal, Z.-Q. You, Y. Zhu, B. Alam, B. J. Albrecht, A. Aldossary, E. Alguire, J. H. Andersen, V. Athavale, D. Barton, K. Begam, A. Behn, N. Bellonzi, Y. A. Bernard, E. J. Berquist, H. G. A. Burton, A. Carreras, K. Carter-Fenk, R. Chakraborty, A. D. Chien, K. D. Closser, V. Cofer-Shabica, S. Dasgupta, M. de Wergifosse, J. Deng, M. Diedenhofen, H. Do, S. Ehlert, P.-T. Fang, S. Fatehi, Q. Feng, T. Friedhoff, J. Gayvert, Q. Ge, G. Gidofalvi, M. Goldey, J. Gomes, C. E. González-Espinoza, S. Gulania, A. O. Gunina, M. W. D. Hanson-Heine, P. H. P. Harbach, A. Hauser, M. F. Herbst, M. Hernández Vera, M. Hodecker, Z. C. Holden, S. Houck, X. Huang, K. Hui, B. C. Huynh, M. Ivanov, A. Jász, H. Ji, H. Jiang, B. Kaduk, S. Kähler, K. Khistyaev, J. Kim, G. Kis, P. Klunzinger, Z. Koczor-Benda, J. H. Koh, D. Kosenkov, L. Koulias, T. Kowalczyk, C. M. Krauter, K. Kue, A. Kunitsa, T. Kus, I. Ladjánszki, A. Landau, K. V. Lawler, D. Lefrancois, S. Lehtola, R. R. Li, Y.-P. Li, J. Liang, M. Liebenthal, H.-H. Lin, Y.-S. Lin, F. Liu, K.-Y. Liu, M. Loipersberger, A. Luenser, A. Manjanath, P. Manohar, E. Mansoor, S. F. Manzer, S.-P. Mao, A. V. Marenich, T. Markovich, S. Mason, S. A. Maurer, P. F. McLaughlin, M. F. S. J. Menger, J.-M. Mewes, S. A. Mewes, P. Morgante, J. W. Mullinax, K. J. Oosterbaan, G. Paran, A. C. Paul, S. K. Paul, F. Pavošević, Z. Pei, S. Prager, E. I. Proynov, A. Rák, E. Ramos-Cordoba, B. Rana, A. E. Rask, A. Rettig, R. M. Richard, F. Rob, E. Rossomme, T. Scheele, M. Scheurer, M. Schneider, N. Sergueev, S. M. Sharada, W. Skomorowski, D. W. Small, C. J. Stein, Y.-C. Su, E. J. Sundstrom, Z. Tao, J. Thirman, G. J. Tornai, T. Tsuchimochi, N. M. Tubman, S. P. Veccham, O. Vydrov, J. Wenzel, J. Witte, A. Yamada, K. Yao, S. Yeganeh, S. R. Yost, A. Zech, I. Y. Zhang, X. Zhang, Y. Zhang, D. Zuev, A. Aspuru-Guzik, A. T. Bell, N. A. Besley, K. B. Bravaya, B. R. Brooks, D. Casanova, J.-D. Chai, S. Coriani, C. J. Cramer, G. Cserey, I. DePrince, A. Eugene, J. DiStasio, Robert A., A. Dreuw, B. D. Dunietz, T. R. Furlani, I. Goddard, William A., S. Hammes-Schiffer, T. Head-Gordon, W. J. Hehre, C.-P. Hsu, T.-C. Jagau, Y. Jung, A. Klamt, J. Kong, D. S. Lambrecht, W. Liang, N. J. Mayhall, C. W. McCurdy, J. B. Neaton, C. Ochsenfeld, J. A. Parkhill, R. Peverati, V. A. Rassolov, Y. Shao, L. V. Slipchenko, T. Stauch, R. P. Steele, J. E. Subotnik, A. J. W. Thom, A. Tkatchenko, D. G. Truhlar, T. Van Voorhis, T. A. Wesolowski, K. B. Whaley, I. Woodcock, H. Lee, P. M. Zimmerman, S. Faraji, P. M. W. Gill, M. Head-Gordon, J. M. Herbert, A. I. Krylov, *The Journal of Chemical Physics* **2021**, *155*, 084801.

[224] R. Ditchfield, W. J. Hehre, J. A. Pople, *The Journal of Chemical Physics* **1971**, *54*, 724–728.

[225] W. J. Hehre, R. Ditchfield, J. A. Pople, *The Journal of Chemical Physics* **1972**, *56*, 2257–2261.

[226] J. D. Dill, J. A. Pople, *The Journal of Chemical Physics* **1975**, *62*, 2921–2923.

[227] J. S. Binkley, J. A. Pople, W. J. Hehre, *Journal of the American Chemical Society* **1980**, *102*, 939–947.

[228] M. S. Gordon, J. S. Binkley, J. A. Pople, W. J. Pietro, W. J. Hehre, *Journal of the American Chemical Society* **1982**, *104*, 2797–2803.

[229] W. J. Hehre, R. F. Stewart, J. A. Pople, *The Journal of Chemical Physics* **1969**, *51*, 2657–2664.

[230] W. Hehre, R. Ditchfield, R. Stewart, J. Pople, *The Journal of Chemical Physics* **1970**, *52*, 2769–2773.

[231] R. Stocks, E. Palethorpe, G. M. J. Barca, Multi-GPU RI-HF Energies and Analytic Gradients — Towards High Throughput Ab Initio Molecular Dynamics, **2024**.

[232] H. Chen, J. Fan, Y. Fu, C.-L. Do-Thanh, X. Suo, T. Wang, I. Popovs, D.-e. Jiang, Y. Yuan, Z. Yang, S. Dai, *Advanced Materials* **2021**, *33*, 2008685.

[233] Y. Pei, Y. Zhang, J. Ma, M. Fan, S. Zhang, J. Wang, *Materials Today Nano* **2022**, *17*, 100159.

[234] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, *Nucleic Acids Research* **2000**, *28*, 235–242.

[235] S. Pérez, A. Sarkar, A. Rivet, S. Drouillard, C. Breton, A. Imberty, *A Practical Guide to Using Glycomics Databases* **2017**, 133.

[236] G. E. Bacon, N. A. Curry, S. A. Wilson, *Proceedings of the Royal Society of London. Series A Mathematical and Physical Sciences* **1964**, *279*, 98–110.

[237] A. Kozak, M. Grottel, A. E. Koziol, Z. Pajak, *Journal of Physics C: Solid State Physics* **1987**, *20*, 5433.

[238] J. D. Bernal, R. H. Fowler, *The Journal of Chemical Physics* **1933**, *1*, 515–548.

[239] S. F. Tang, V. Smetana, M. K. Mishra, S. P. Kelley, O. Renier, R. D. Rogers, A. V. Mudring, *Inorganic Chemistry* **2020**, *59*, 7227–7237.

[240] D. Broderick, J. Herbert, Delocalization error poisons the density-functional many-body expansion, **2024**.

[241] D. A. Case, H. M. Aktulga, K. Belfon, D. S. Cerutti, G. A. Cisneros, V. W. D. Cruzeiro, N. Forouzesh, T. J. Giese, A. W. Götz, H. Gohlke, S. Izadi, K. Kasavajhala, M. C. Kaymak, E. King, T. Kurtzman, T.-S. Lee, P. Li, J. Liu, T. Luchko, R. Luo, M. Manathunga, M. R. Machado, H. M. Nguyen, K. A. O'Hearn, A. V. Onufriev, F. Pan, S. Pantano, R. Qi, A. Rahnamoun, A. Risheh, S. Schott-Verdugo, A. Shajan, J. Swails, J. Wang, H. Wei, X. Wu, Y. Wu, S. Zhang, S. Zhao, Q. Zhu, T. E. I. Cheatham, D. R. Roe, A. Roitberg, C. Simmerling, D. M. York, M. C. Nagan, K. M. J. Merz, *Journal of Chemical Information and Modeling* **2023**, *63*, PMID: 37805934, 6183–6191.

[242] A. P. Niranjan Kumar, Rakesh Srivastava, A. M. Lynn, *Journal of Biomolecular Structure and Dynamics* **2020**, *38*, 3396–3410.

[243] H. K. Srivastava, G. N. Sastry, *Journal of Chemical Information and Modeling* **2012**, *52*, 3088–3098.

[244] H. Sun, Y. Li, M. Shen, S. Tian, L. Xu, P. Pan, Y. Guan, T. Hou, *Phys. Chem. Chem. Phys.* **2014**, *16*, 22035–22045.

[245] N. Chéron, E. I. Shakhnovich, *Journal of Computational Chemistry* **2017**, *38*, 1941–1951.

[246] F. Chen, H. Liu, H. Sun, P. Pan, Y. Li, D. Li, T. Hou, *Phys. Chem. Chem. Phys.* **2016**, *18*, 22129–22139.

[247] C. Seidl, G. M. Barca, *Journal of Chemical Theory and Computation* **2022**, *18*, 4164–4176.

[248] A. Abdelfattah, D. Keyes, H. Ltaief, *ACM Trans. Math. Softw.* **2016**, *42*, DOI 10.1145/2818311.

[249] A. V. Marenich, C. P. Kelly, J. D. Thompson, G. D. Hawkins, C. C. Chambers, D. J. Giesen, P. Winget, C. J. Cramer, D. G. Truhlar, Minnesota Solvation Database version 2012, Accessed June 12, 2024, University of Minnesota: Minneapolis, MN, **2012**.

[250] D. F. Hahn, J. R. Wagner, Protein-Ligand Benchmark Dataset for Free Energy Calculations, version 0.2.1, **2022**.

[251] D. Hahn, C. Bayly, M. L. Boby, H. Bruce Macdonald, J. Chodera, V. Gapsys, A. Mey, D. Mobley, L. Perez Benito, C. Schindler, G. Tresadern, G. Warren, *Living Journal of Computational Molecular Science* **2022**, *4*, 1497.

[252] P. Mittal, S. Singh, R. Sinha, A. Shrivastava, A. Singh, I. K. Singh, *International Journal of Biological Macromolecules* **2021**, *187*, 999–1018.

[253] S. Danese, L. Peyrin-Biroulet, *Inflammatory Bowel Diseases* **2021**, *27*, 2023–2030.

[254] Y. Duan, C. Wu, S. Chowdhury, M. C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, et al., *Journal of computational chemistry* **2003**, *24*, 1999–2012.

[255] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, D. A. Case, *Journal of computational chemistry* **2004**, *25*, 1157–1174.

[256] X. He, V. H. Man, W. Yang, T.-S. Lee, J. Wang, *The Journal of chemical physics* **2020**, *153*.

[257] C. M. Breneman, K. B. Wiberg, *Journal of Computational Chemistry* **1990**, *11*, 361–373.

[258] B. Hess, H. Bekker, H. J. Berendsen, J. G. Fraaije, *Journal of computational chemistry* **1997**, *18*, 1463–1472.

[259] G. Bussi, D. Donadio, M. Parrinello, *The Journal of Chemical Physics* **2007**, *126*, 014101.

[260] H. J. Berendsen, J. v. Postma, W. F. Van Gunsteren, A. DiNola, J. R. Haak, *The Journal of chemical physics* **1984**, *81*, 3684–3690.

[261] Y.-j. Sheng, Y.-w. Yin, Y.-q. Ma, H.-m. Ding, *Journal of Chemical Information and Modeling* **2021**, *61*, 2454–2462.

[262] Y.-X. Zhu, Y.-J. Sheng, Y.-Q. Ma, H.-M. Ding, *The Journal of Physical Chemistry B* **2022**, *126*, 1700–1708.

[263] H. Sun, Y. Li, S. Tian, L. Xu, T. Hou, *Physical Chemistry Chemical Physics* **2014**, *16*, 16719–16729.

[264] D. Case, H. Aktulga, K. Belfon, I. Ben-Shalom, J. Berryman, S. Brozell, D. Cerutti, T. Cheatham, III, G. Cisneros, V. Cruzeiro, T. Darden, N. Forouzesh, M. Ghazimirsaeed, G. Giambaşu, T. Giese, M. Gilson, H. Gohlke, A. Goetz, J. Harris, Z. Huang, S. Izadi, S. Izmailov, K. Kasavajhala, M. Kaymak, A. Kovalenko, T. Kurtzman, T. Lee, P. Li, Z. Li, C. Lin, J. Liu, T. Luchko, R. Luo, M. Machado, M. Manathunga, K. Merz, Y. Miao, O. Mikhailovskii, G. Monard, H. Nguyen, K. O'Hearn, A. Onufriev, F. Pan, S. Pantano, A. Rahnamoun, D. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, A. Shajan, J. Shen, C. Simmerling, N. Skrynnikov, J. Smith, J. Swails, R. Walker, J. Wang, J. Wang, X. Wu, Y. Wu, Y. Xiong, Y. Xue, D. York, C. Zhao, Q. Zhu, P. Kollman, Amber 2022, University of California, San Francisco, **2022**.

[265] A. v. Bondi, *The Journal of physical chemistry* **1964**, *68*, 441–451.

[266] Á. González, *Mathematical geosciences* **2010**, *42*, 49–64.

[267] S. Genheden, T. Luchko, S. Gusarov, A. Kovalenko, U. Ryde, *Journal of Physical Chemistry B* **2010**, *114*, 8505–8516.

[268] J. L. Whitten, *The Journal of Chemical Physics* **1973**, *58*, 4496–4501.

[269] M. Häser, R. Ahlrichs, *Journal of Computational Chemistry* **1989**, *10*, 104–111.

[270] P. M. Gill in (Eds.: J. R. Sabin, M. C. Zerner), Advances in Quantum Chemistry, Academic Press, **1994**, pp. 141–205.

[271] S. A. Slattery, K. A. Surjuse, C. C. Peterson, D. A. Penchoff, E. F. Valeev, *Physical Chemistry Chemical Physics* **2024**.

[272] N. J. Mayhall, K. Raghavachari, *Journal of chemical theory and computation* **2011**, *7*, 1336–1343.

[273] M. A. Collins, M. W. Cvitkovic, R. P. Bettens, *Accounts of chemical research* **2014**, *47*, 2776–2785.

[274] A. K. Gupta, S. Maier, B. Thapa, K. Raghavachari, *Journal of Chemical Theory and Computation* **2024**, *20*, 2774–2785.

[275] F. Weigend, M. Häser, *Theoretical Chemistry Accounts* **1997**, *97*, 331–340.

[276] S. Tadesse, A. T. Anshabo, N. Portman, E. Lim, W. Tilley, C. E. Caldon, S. Wang, *Drug discovery today* **2020**, *25*, 406–413.

[277] R. Hantani, S. Hanawa, S. Oie, K. Umetani, T. Sato, Y. Hantani, *SLAS DISCOVERY: Advancing Life Sciences R&D* **2019**, *24*, 854–862.

[278] A. Heifetz, G. Trani, M. Aldeghi, C. H. MacKinnon, P. A. McEwan, F. A. Brookfield, E. I. Chudyk, M. Bodkin, Z. Pei, J. D. Burch, et al., *Journal of medicinal chemistry* **2016**, *59*, 4352–4363.

[279] M. Kuhn, S. Firth-Clark, P. Tosco, A. S. Mey, M. Mackey, J. Michel, *Journal of Chemical Information and Modeling* **2020**, *60*, 3120–3130.

[280] H.-C. Tsai, J. Xu, Z. Guo, Y. Yi, C. Tian, X. Que, T. Giese, T.-S. Lee, D. M. York, A. Ganguly, et al., *Journal of Chemical Information and Modeling* **2024**, *64*, 7046–7055.

[281] L. F. Song, T.-S. Lee, C. Zhu, D. M. York, K. M. Merz Jr, *Journal of chemical information and modeling* **2019**, *59*, 3128–3135.

[282] F. Sabanés Zariquiey, A. Pérez, M. Majewski, E. Gallicchio, G. De Fabritiis, *Journal of chemical information and modeling* **2023**, *63*, 2438–2444.

[283] M. Jalaie, J. Fanfrlík, A. Pecina, M. Lepšík, J. Řezáč, *Journal of Chemical Information and Modeling* **2025**, *65*, 8127–8136.

[284] J. A. Lemkul, *The Journal of Physical Chemistry B* **2024**, *128*, 9418–9435.

[285] J. N. Cumming, E. M. Smith, L. Wang, J. Misiaszek, J. Durkin, J. Pan, U. Iserloh, Y. Wu, Z. Zhu, C. Strickland, et al., *Bioorganic & medicinal chemistry letters* **2012**, *22*, 2444–2449.

[286] I. R. Hardcastle, C. E. Arris, J. Bentley, F. T. Boyle, Y. Chen, N. J. Curtin, J. A. Endicott, A. E. Gibson, B. T. Golding, R. J. Griffin, et al., *Journal of medicinal chemistry* **2004**, *47*, 3710–3722.

[287] B. G. Szczepankiewicz, C. Kosogof, L. T. Nelson, G. Liu, B. Liu, H. Zhao, M. D. Serby, Z. Xin, M. Liu, R. J. Gum, et al., *Journal of medicinal chemistry* **2006**, *49*, 3563–3580.

[288] A. Friberg, D. Vigil, B. Zhao, R. N. Daniels, J. P. Burke, P. M. Garcia-Barrantes, D. Camper, B. A. Chauder, T. Lee, E. T. Olejniczak, et al., *Journal of medicinal chemistry* **2013**, *56*, 15–30.

[289] D. M. Goldstein, M. Soth, T. Gabriel, N. Dewdney, A. Kuglstatter, H. Arzeno, J. Chen, W. Bingenheimer, S. A. Dalrymple, J. Dunn, et al., *Journal of Medicinal Chemistry* **2011**, *54*, 2255–2265.

[290] D. P. Wilson, Z.-K. Wan, W.-X. Xu, S. J. Kirincich, B. C. Follows, D. Joseph-McCarthy, K. Foreman, A. Moretto, J. Wu, M. Zhu, et al., *Journal of medicinal chemistry* **2007**, *50*, 4681–4698.

[291] B. Baum, M. Mohamed, M. Zayed, C. Gerlach, A. Heine, D. Hangauer, G. Klebe, *Journal of molecular biology* **2009**, *390*, 56–69.

[292] J. Liang, V. Tsui, A. Van Abbema, L. Bao, K. Barrett, M. Beresini, L. Berezhkovskiy, W. S. Blair, C. Chang, J. Driscoll, et al., *European journal of medicinal chemistry* **2013**, *67*, 175–187.

[293] J. Liang, A. van Abbema, M. Balazs, K. Barrett, L. Berezhkovsky, W. Blair, C. Chang, D. Delarosa, J. DeVoss, J. Driscoll, et al., *Journal of medicinal chemistry* **2013**, *56*, 4521–4536.

[294] M. Parrinello, A. Rahman, *Journal of Applied physics* **1981**, *52*, 7182–7190.

[295] D. van der Spoel, H. Berendsen, *Biophysical journal* **1997**, *72*, 2032–2041.

[296] M. Gao, J. Skolnick, *PLoS computational biology* **2013**, *9*, e1003302.

[297] K. Kriz, J. Rezac, *Journal of Chemical Information and Modeling* **2019**, *59*, 229–235.

[298] T. Hou, J. Wang, Y. Li, W. Wang, *Journal of chemical information and modeling* **2011**, *51*, 69–82.

[299] A. Klamt, G. Schüürmann, *Journal of the Chemical Society Perkin Transactions 2* **1993**, 799–805.

[300] A. Pecina, J. Fanfrlík, M. Lepšík, J. Řezáč, *Nature Communications* **2024**, *15*, 1127.

[301] J. Rezac, P. Hobza, *Journal of Chemical Theory and Computation* **2012**, *8*, 141–151.

[302] J. Řezáč, P. Hobza, *Chemical Physics Letters* **2011**, *506*, 286–289.

[303] J. J. Stewart, *Journal of Molecular modeling* **2007**, *13*, 1173–1213.

[304] J. Řezáč, Cuby: An integrative framework for computational chemistry, **2016**.

[305] J. Řezáč, O. V. Kontkanen, M. Nováček, *The Journal of Chemical Physics* **2024**, *160*.

[306] J. E. Moussa, J. J. P. Stewart, *MOPAC*, version 23.2, **2025**.

[307] J. J. Stewart, *International journal of quantum chemistry* **1996**, *58*, 133–146.

[308] K. Kříž, J. Řezáč, *Physical Chemistry Chemical Physics* **2022**, *24*, 14794–14804.

[309] P. G. Wyatt, A. J. Woodhead, V. Berdini, J. A. Boulstridge, M. G. Carr, D. M. Cross, D. J. Davis, L. A. Devine, T. R. Early, R. E. Feltell, E. J. Lewis, R. L. McMenamin, E. F. Navarro, M. A. O'Brien, M. O'Reilly, M. Reule, G. Saxty, L. C. A. Seavers, D.-M. Smith, M. S. Squires, G. Trewartha, M. T. Walker, A. J.-A. Woolford, *Journal of Medicinal Chemistry* **2008**, *51*, 4986–4999.

[310] M. Congreve, G. Chessari, D. Tisi, A. J. Woodhead, *Journal of medicinal chemistry* **2008**, *51*, 3661–3680.

[311] C. H. MacKinnon, K. Lau, J. D. Burch, Y. Chen, J. Dines, X. Ding, C. Eigenbrot, A. Heifetz, A. Jaochico, A. Johnson, et al., *Bioorganic & Medicinal Chemistry Letters* **2013**, *23*, 6331–6335.

[312] Z. Zhu, Z.-Y. Sun, Y. Ye, J. Voigt, C. Strickland, E. M. Smith, J. Cumming, L. Wang, J. Wong, Y.-S. Wang, et al., *Journal of medicinal chemistry* **2010**, *53*, 951–965.

[313] C. Wang, P. H. Nguyen, K. Pham, D. Huynh, T.-B. N. Le, H. Wang, P. Ren, R. Luo, *Journal of computational chemistry* **2016**, *37*, 2436–2446.

[314] T. Yang, J. C. Wu, C. Yan, Y. Wang, R. Luo, M. B. Gonzales, K. N. Dalby, P. Ren, *Proteins: Structure Function and Bioinformatics* **2011**, *79*, 1940–1951.

[315] F. Chen, H. Liu, H. Sun, P. Pan, Y. Li, D. Li, T. Hou, *Physical Chemistry Chemical Physics* **2016**, *18*, 22129–22139.

[316] P. Söderhjelm, F. Aquilante, U. Ryde, *The Journal of Physical Chemistry B* **2009**, *113*, 11085–11094.

[317] M. K. Gilson, L. E. Stewart, M. J. Potter, S. P. Webb, *Journal of Chemical Theory and Computation* **2024**, *20*, 6328–6340.