Investigating the Properties of Superfluid He-4 through Density Functional Calculations

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Matthew Francis Dutra

May 2017

© by Matthew Francis Dutra, 2017
All Rights Reserved.

Dedicated to Mom, Dad, and Kerry. You guys totally rock.

Acknowledgements

I wish to first and foremost express my thanks to my research advisor, Robert Hinde, whose guidance and mentorship over the past five years has been irrepleacable, and without whom my graduate studies would not exist. I would also like to extend my gratitude to the many other teachers and professors whom I have been privileged to meet and learn from throughout my academic journey, most particularly Dr. Yosuke Kanai and Mr. Robert Gotwals - you are both responsible for the birth of my interest in computational chemistry, and for that I cannot thank you enough. Furthermore, I would like to thank the members of my graduate committee for seeing me through the whole dissertation process.

I have also had the very good fortune of interacting with a wonderful research group in my day-to-day graduate life. To the past and present members of the Hinde lab - Ashleigh Barnes, Kiran Dhah, Dan D'Andrea, Randi Beil, Paul Mott, James Smith, Matt Wilson, Tim Lillestolen, and Jacob Tande - thanks for five years of memories. You have given me countless hours of laughs (and baked goods!), and been a constant source of fun in otherwise stressful times. You've also kept me grounded - I haven't forgotten the kind criticisms from each of you after listening to all of my practice talks and presentation runthroughs - and I feel like I've become a better scientist because of you.

It would be remiss of me to not acknowledge the network of friends I have made outside of the Hinde group as well, both for celebrating with me through good times and enduring me through the bad. Jesse, Wyatt, Jenn (both of you!), Eric, Rachel,

Nicole, Danielle, and many others - you have provided me with weekend escapes, game nights, dinner parties, and dancing, and I can't imagine the last five years without any of you. You all have my gratitude.

Finally, an enormous thank you to Mom, Dad, and Kerry, my loving family who have provided me with motivation and support for as long as I can remember.

The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. - P.A.M. Dirac

Abstract

We present a study of isotopically pure He-4 systems evaluated using helium density functional theory (He-DFT) with the intent of better understanding their ground state structural and energetic properties, particularly within the scope of singularly-doped helium droplets. We self-consistently solve for the density profiles and chemical potentials for a wide range of pure helium droplet sizes (up to 9500 atoms) via an imaginary time propagation method, and fit the resultant energetic data to a power law formula to be able to extrapolate values for even larger droplets. Subsequent calculations on singularly-doped droplets within the same size range yield accurate binding energies for atomic dopants. We then suggest a method of predicting droplet size distributions after pickup of a dopant atom based on an initial distribution and a particular quantity of imparted energy using chemical potential values predicted from our He-DFT calculations, with the intent of providing a means for understanding data generated by helium droplet calorimetry experiments. Along the way, we also rigorously analyze four of the most popular helium density functionals published in the literature in terms of their robustness, computational cost, and accuracy. To conclude, we perform He-DFT simulations of systems that deviate far from bulk helium behavior in order to evaluate the capabilities of the He-DFT method in such scenarios. We find that a density functional treatment of both small and strongly bound droplets is a surprisingly adequate and efficient alternative to more powerful quantum Monte Carlo calculations.

Table of Contents

1 Introduction **1**

1.1 Document Outline 1

1.2 Helium-4 as a Superfluid 2

1.3 Helium Spectroscopy 5

1.4 Quantum Monte Carlo Techniques 6

1.5 Helium Density Functional Theory 9

2 Planar He-4 Films and Attractive Surfaces **13**

2.1 Stringari-Treiner and Orsay-Paris Functionals 15

2.2 Self-Consistent Methods 17

2.3 Attractive Planar Surfaces 23

2.4 Framework for Spherical Systems 35

3 Doped He-4 Droplets **45**

3.1 Orsay-Paris, Catalonia, and Orsay-Trento Functionals 46

3.2 Self-Consistent Approach 59

3.3 Analysis of System Discretization 66

3.4 Pure Droplet Results 69

3.5 Droplets with Noble Gas Dopants 76

3.6 Droplets with Transition Metal Dopants 85

3.7 Application to He Droplet Calorimetry 92

3.8 Discussion and Conclusions 104

4 He-DFT in Unconventional Environments **107**

4.1 Small Droplets 107

4.2 Super-bulk Densities 117

5 Conclusions **125**

5.1 Conclusions 125

5.2 Outlook and Future Directions 127

Bibliography **129**

Appendix **137**

A 3-D Droplet He-DFT Code **138**

A.1 Main Program and Functionals 138

A.2 ITP Algorithm for 3-D He-DFT Code 157

B 1-D Droplet He-DFT Code **160**

Vita **179**

List of Tables

2.1 Chemical potentials calculated for N = 40 and N = 112 atom droplets using the 3-D coordinate system. The numerical subscripts on the chemical potentials indicate the number of sampling points used in each direction for that particular calculation. The last two columns are chemical potential values calculated by Barranco and Hernandez [1] and by us using a 1-D sampling scheme, respectively. Voided entries indicate simulations that did not complete in the available computing time, and so have no converged chemical potential data. 42

3.1 Functional parameters for the functionals given in [2], [1], and [3]. Voided entries indicate that a particular parameter is not applicable to the given functional. 49

3.2 Energetic quantities predicted for an N=1000 atom droplet with various time step parameters. The green entry indicates the final time step chosen for our simulations, while the red entry represents an example of a particularly 'bad' time step based on the lack of convergence seen in the E/N and μ values. 61

3.3 Effect of system discretization on the energetic properties of pure droplets. 68

3.4 Energetic data for droplets containing non-integer numbers of atoms. Note that the approximations (dE/dN) to the chemical potential at the m±1 and m±2 values (where the m indicates the ingeter-value droplet, and m±1, m±2 are that many 'steps' from the integer droplet) are in agreement with one another to 6+ decimal places, and in agreement to the chemical potential at N to within ≈0.01 K. 70
3.5 Energetic data for select droplet sizes using the Catalonia density functional. 72
3.6 Energetic data for select droplet sizes using the Orsay-Paris and Orsay-Trento density functionals. The entries are taken relative to the Catalonia values of Table 3.5, and color-coordinated to show stronger (blue) versus weaker (red) binding behavior relative to the Catalonia functional. The last column shows the ratio of densities of the centers of the droplets (ρ_c) to that of the bulk value (ρ_0). 73
3.7 Fitting parameters for Eq. 3.33 75
3.8 Asymptotic binding energies for noble gas dopants. The uncertainty in the values of [4] stems from visual estimation of their Fig. 5. The voided entries result from a lack of available literature data, while the entry with an asterisk corresponds to the value for an N=500 droplet. 79
3.9 Composition of the total rare gas-doped droplet energies relative to the pure droplet for the N=5000 case. Destabilizing effects are indicated by a negative sign. 82
3.10 Asymptotic binding energies of gold, silver, and copper dopants for all functionals. 87
3.11 Composition of the total transition metal-doped droplet energy relative to the pure droplet for the N=5000 case. Destabilizing effects are indicated by a negative sign. 89

3.12 Values of ΔN_{evap} predicted for lognormal (σ=0.468) distributions with various average droplet sizes (represented in the left-hand column by their α parameter) and injected energies. The values in parentheses correspond to the observed $\langle N_1 \rangle$ for the particular distribution generated. Voided entries indicate complete evaporation of the initial average droplet by the energy transferred. [a]Energetic values taken from Table III of reference [5]. [b]Energetic values taken from Table I of reference [6]. 95

3.13 Values of ΔN_{evap} predicted for lognormal (σ=0.573) distributions with various average droplet sizes (represented in the left-hand column by their α parameter) and injected energies. The values in parentheses correspond to the observed $\langle N_1 \rangle$ for the particular distribution generated. Voided entries indicate complete evaporation of the initial average droplet by the energy transferred. [a]Energetic values taken from Table III of reference [5]. [b]Energetic values taken from Table I of reference [6]. 96

3.14 Values of ΔN_{evap} predicted for lognormal (σ=0.612) distributions with various average droplet sizes (represented in the left-hand column by their α parameter) and injected energies. The values in parentheses correspond to the observed $\langle N_1 \rangle$ for the particular distribution generated. Voided entries indicate complete evaporation of the initial average droplet by the energy transferred. [a]Energetic values taken from Table III of reference [5]. [b]Energetic values taken from Table I of reference [6]. 97

4.1 Energies for small He_N clusters calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals and an imaginary time step $\Delta\tau = 2500$ a.u. Benchmarking values are calculated using DMC methods by various authors (see text); the uncertainty in these reported values is maintained in the conversion from cm^{-1} to K via conservation of significant figures. Parenthetical quantities indicate percent error with respect to the DMC calculations. 109

4.2 Chemical potentials for small He_N clusters calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals with an imaginary time step $\Delta\tau = 2500$ a.u. Benchmarking values are derived from calculated DMC values by various authors (see text) according to Eq. 4.1, where precision is maintained to the lowest number of decimal places reported. Parenthetical quantities indicate percent error with respect to the DMC calculations. 111

4.3 Binding energies for small He_N clusters doped with a single central Ag atom, calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals and an imaginary time step $\Delta\tau = 2500$ a.u. Benchmarking values are DMC values by Mella et al[7]. Parenthetical quantities indicate percent error with respect to the DMC calculations. 114

4.4 Binding energies for small He_N clusters doped with a single central Ag atom described by a modern potential fitted to ab initio data, calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals and an imaginary time step $\Delta\tau = 2500$ a.u. 118

4.5 **Upper Half:** Calculated structural properties of helium droplets solvating a single ionic dopant for an N=5000 atom droplet and a time step $\Delta\tau = 100$ a.u. **Lower Half:** Literature values for the same quantities. VMC data is from reference [8], DMC data is from reference [9], PIMC data is from reference [10], PIGS data is from reference [11], and experimental data is from reference [12]. Quantities with an asterisk were visually approximated from graphical He radial density distribution functions with an approximate accuracy of ±0.1. Quantities with a double asterisk are from size N=128 atom droplets at T=1 K from reference [10]. 120

List of Figures

2.1 Cartoon depicting the one-dimensional sampling scheme of a free helium slab. The $\vec{x}$ and $\vec{y}$ directions are assumed to be infinite, while $\vec{z}$ spans the finite width of the slab. The grey area is representative of an attractive surface, while the shades of blue indicate the various helium densities that might be found in solvation layers adjacent to the surface. The lightest blue block at the top denotes bulk helium density. 14

2.2 Pictorial representation of the 'mixing' algorithm used in our density functional calculations for planar slabs of helium. 20

2.3 Pictorial representation of the ITP algorithm used in our density functional calculations for planar slabs of helium. FFT = Fast Fourier Transform 24

2.4 Density profile of a free planar slab of He-4 calculated using the Stringari-Treiner functional (green, left axis). Also shown is the corresponding mean field potential (red, right axis). The region bound by the black brackets contains the points that were checked to reproduce available literature data. 25

2.5 Helium density profiles for systems involving an artificial Lennard-Jones surface located at z=0 using the Stringari-Treiner functional. The vertical black line represents the LJ sigma parameter, where the He-surface interaction switches from repulsive to attractive. 28

2.6 Helium density profiles for systems involving an artificial Lennard-Jones surface located at z=0 using the Orsay-Paris functional. The vertical black line represents the LJ sigma parameter, where the He-surface interaction switches from repulsive to attractive. 29
2.7 Solvation layer region of a He-4 density profile adsorbed to a planar graphite surface, calculated using the Orsay-Paris functional and the 'mixing' method. The graphite surface is located at z=0. Inset: Computed values for the first and first+second solvation layer densities, as well as for the He-graphite distance. The literature values represent neutron studies performed by Carneiro et al. [13] 31
2.8 Consecutive density profiles of helium adsorbed onto a graphite surface, calculated using the 'mixing' method. The red line represents the nearly-converged density of an arbitrary (m-1)th iteration, while the dashed blue line shows the newly calculated density profile of iteration m. Note that this density has been pushed away from the surface entirely; we refer to this as the 'sloshing' of sequentially calculated density profiles, and it is indicative of a convergence issue with the 'mixing' method as a whole. 33
2.9 Surface regions of multiple density profiles of free planar slabs of He-4 calculated using the Orsay-Paris functional with the ITP method. Inset: Calculated chemical potential values corresponding to the various sampling meshes tested. 34
2.10 Solvation layer regions of He-4 density profiles adsorbed to a planar graphite surface, calculated using the Orsay-Paris functional. The graphite surface is located at z=0. Inset: Computed values for the first and first+second solvation layer densities, as well as for the He-graphite distance using both the 'mixing' and ITP profiles. The literature values represent neutron studies performed by Carneiro et al. 36

2.11 Two-dimensional depiction of the gridding problem incurred by attempting to describe the radial distance cutoff h_{OP} using Cartesian coordinates. The point of origin is at (10,10), and the length of the radius h_{OP} is not to scale. The green box highlights a problematic zone where the form of the interaction potential changes within a single integrable region. 38

2.12 Calculated chemical potentials as a function of finer sampling mesh sizes for a 40-atom droplet using the Catalonia density functional. The horizontal green and blue lines represent the chemical potential for a 40-atom droplet calculated by Barranco and Hernandez[1] and by us using a 1-D droplet scheme, respectively. 43

2.13 Calculated chemical potentials as a function of finer sampling mesh sizes for a 112-atom droplet using the Catalonia density functional. The horizontal green and blue lines represent the chemical potential for a 112-atom droplet calculated by Barranco and Hernandez[1] and by us using a 1-D droplet scheme, respectively. 44

3.1 Cross-section of a helium droplet (blue circle) showing the distinct integration regions of Eqs. 3.12 - 3.14. The short-range interactions are contained within the red circle, while the dashed circles show the radial integral boundaries. In this instance, the only boundary aligns with the short-range cutoff distance, h_{cut}. Note that, in the Catalonia model, the parameter h_{cut} should be replaced with the Lennard-Jones sigma for He-He interactions. 52

3.2 Cross-section of a helium droplet (blue circle) showing the distinct integration regions of Eqs. 3.15 - 3.17. The short-range interactions are contained within the red circle, while the dashed circles show the radial integral boundaries. The angular portion is omitted for clarity. Note that, in the Catalonia model, the parameter h_{cut} should be replaced with the Lennard-Jones sigma for He-He interactions. 54

3.3 Cross-section of a helium droplet (blue circle) showing the distinct integration regions of Eqs. 3.18 - 3.20. The short-range interactions are contained within the red circle, while the dashed circles show the radial integral boundaries. The angular portion is represented by a green arc. Note that, in the Catalonia model, the parameter h_{cut} should be replaced with the Lennard-Jones sigma for He-He interactions. . . 57

3.4 A portion of the surface region of resultant density profiles for an N=1000 atom droplet after 5000 iterations with varying imaginary time step parameters using the Orsay-Paris functional. Note that the values for $\Delta\tau$ = 1000-5000 a.u. produce density profiles that are all aligned with one another. 62

3.5 Resultant density profiles for an N=1000 atom droplet calculated with the Orsay-Paris functional after 5000 iterations with two time step parameters, $\Delta\tau$=2500 a.u. and $\Delta\tau$=9000 a.u. Note that the latter of these values produces an unstable and unconverged density profile. . . 63

3.6 Diagram of the iterative procedure used to calculate self-consistent helium density profiles. ITP = Imaginary Time Propagation, HT = Hankel Transform 67

3.7 Energy as a function of droplet size for each functional. The colored lines correspond to a fit of Eq. 3.33 to the appropriate points. Some values are omitted for clarity. 74

3.8 Density profiles of Ne-He_{1000}, Ar-He_{1000}, and Xe-He_{1000} droplets using the Orsay-Paris functional. The pure 1000 helium atom droplet profile is included for reference. 78
3.9 Binding energies for Ne, Ar, and Xe doped droplets. 81
3.10 Chemical potential of Ne, Ar, and Xe doped droplets as a function of droplet size, calculated using the Orsay-Paris functional. The solid line connects the calculated chemical potential values of the pure droplet case, and is drawn to guide the eye. The inset focuses on the large droplet limit; the agreement with the undoped chemical potential values (black line) is excellent. 83
3.11 Density profiles of Ne-doped droplets with various particle numbers, calculated using the Orsay-Paris functional. The horizontal black line represents the theoretical He-4 bulk density. Note the buildup of the bulk region beyond the solvation layers as the droplet size increases. . 84
3.12 Density profiles of Au-He_{1000}, Ag-He_{1000}, and Cu-He_{1000} droplets using the Orsay-Paris functional. The pure 1000 helium atom droplet profile is included for reference. 86
3.13 Binding energies for Au, Ag, and Cu doped droplets. 88
3.14 Chemical potential of Au, Ag, and Cu doped droplets as a function of droplet size, calculated using the Orsay-Paris functional. The solid line connects the chemical potential values of the pure droplet case, and is drawn to guide the eye. The inset again focuses on the large droplet limit; the agreement with the undoped chemical potential values (black line) is again excellent. 90
3.15 Density profiles of Au-doped droplets with various particle numbers, calculated using the Orsay-Paris functional. The horizontal black line represents the theoretical He-4 bulk density. Note the buildup of the bulk region beyond the solvation layers as the droplet size increases. . 91

3.16 Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 2600$ injected with an energy of 1323 K. The difference in average values between the slow and fast distributions is 1 atom. 99
3.17 Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 2600$ injected with an energy of 2565 K. The difference in average values between the slow and fast distributions is 2.3 atoms. 100
3.18 Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 2600$ injected with an energy of 4398 K. The difference in average values between the slow and fast distributions is 3 atoms. 101
3.19 Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 12000$ injected with an energy of 69630 K. The difference in average values between the slow and fast distributions is 233 atoms. 102
3.20 Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 25000$ injected with an energy of 150860 K. The difference in average values between the slow and fast distributions is 365 atoms. 103

4.1 Density profiles for select small droplets ranging in size from N=5 to N=40 atoms calculated using the Orsay-Paris functional. Note that even the largest droplet shown here still never achieves bulk density (represented by the horizontal black line) at any point. 112
4.2 Comparison of Ag-He potentials used to model small silver-doped helium clusters. Note that the potential of Tong et al. predicts an attractive well similar in magnitude to that of the He-He Lennard-Jones well. 116

4.3 Density profiles for the cation-containing helium droplets with N=1000 atoms. The dotted points indicate the 'snowball' solvation peak. . . . 121
4.4 Density profiles for the anion-containing helium droplets with N=1000 atoms. The dotted points indicate the 'snowball' solvation peak. Note that the chloride ion does not induce a high-density first solvation shell similar to the other ions studied; we interpret this as a lack of a compact, solid-like snowball in the center of the droplet. 122

Chapter 1

Introduction

1.1 Document Outline

The content of this document is outlined in the following manner. Chapter one presents an introduction to superfluids and superfluid behavior, with a focus on the role of superfluid He-4 in the history and development of superfluid theory. Furthermore, two modern computational techniques - quantum Monte Carlo and helium density functional theory - are introduced to provide context for current theoretical research and motivation for the use of helium density functional theory in this particular project.

Chapter two discusses the pair of self-consistent algorithms used in calculating the ground state helium densities, focusing on the efficiency and accuracy of each. A section devoted to exploring, in thorough detail, two different helium density functionals, known as the Stringari-Treiner and Orsay-Paris functionals, completes the description of the methodologies used in the remainder of the chapter. Next, bulk slabs of He-4 both in free space and adsorbed on structureless Lennard-Jones attractive surfaces are used to qualitatively distinguish both the self-consistent methods and the functionals from one another. We complete the comparative analysis by using the methods to carry out calculations on a realistic test case - helium

adsorbed to a planar graphite surface. Finally, a section devoted to establishing the framework for the computation of spherically symmetric droplet systems rounds out the chapter.

Chapter three explores the realm of atomically doped droplets using the computational setup laid out in chapter two. To do this, the Catalonia and Orsay-Trento density functionals are described in detail, along with the aforementioned Orsay-Paris functional. An assessment of the structural and energetic properties of both pure and doped He-4 droplets in the small and large limits is presented, and a further comparison of the functionals used helps to distinguish their utility.

Chapter four underscores the importance of the helium-dopant pair potential on the calculation of doped droplet systems. A variety of resident atoms are used to highlight the effect of changing the external potential felt by the helium system, with particular focus on silver as the atomic dopant. Two Ag-He pair potentials are investigated, and the resultant droplet data compared to one another, as well as to quantum Monte Carlo results available in the literature. Conclusions are then drawn as to the importance of accurate pair potentials in the He-DFT approach, as well as the accuracy of the He-DFT method as a whole.

Lastly, chapter five finishes the document by presenting overall conclusions regarding the work discussed in chapters two, three, and four, and provides insight into future interests and possible directions for the project. Copies of the Fortran codes developed and used in this work are made available in the subsequent appendices.

1.2 Helium-4 as a Superfluid

As the only element to exhibit fluid behavior at absolute zero temperature and ambient pressure, helium has garnered attention from physicists and chemists alike in the 20th and 21st centuries. Perhaps the most interesting property of helium-4 (the dominant isotope of helium, abbreviated herein as He-4) is that it undergoes a phase transition into a superfluid state below the lambda line at T=2.17 K - a

phenomenon first observed in the late 1930s by Kapitza [14] and by Allen and Misener [15]. From these studies, it was determined that He-4 in the superfluid phase exhibits high thermal conductivity and negligible viscosity, properties that are today known to be characteristic of superfluids in general. London subsequently postulated that this phase transition was a real manifestation of Bose-Einstein condensation (BEC) [16], a phenomenon described by Bose and Einstein in which a collection of non-interacting, identical, and indistinguishable bosonic (integer spin) particles collapse into their energetic single-particle ground state as they cool [17, 18, 19] (as a brief aside, He-3 - a fermion - also undergoes a superfluid transition via Cooper pairing, a phenomenon that can be thought of as an analogue to BEC for fermions[20]). Independent works by London [16] and by Landau [21] further characterized liquid He-4 as a two-component system, in which there exist classical and superfluid components. This characterization was later verified by Peshkov in his measurement of second sound in He-4 [22] - a phenomenon that by nature necessitates the presence of normal and superfluid components.

From this brief history of the discoveries regarding superfluid helium (for a more thorough review, consider [23, 24]), a number of distinct properties regarding the nature of quantum superfluids can be extracted. Firstly, they are accurately described as a collection of indistinguishable particles/quasiparticles, a large fraction of which occupy some single-particle ground state as a result of Bose-Einstein condensation in bosonic systems or Cooper pairing in fermionic systems. This heavily populated single-particle ground state gives rise to macroscopically measurable quantum effects. As an illustrative example, consider a macroscopic wavefunction descriptive of a system of superfluid He-4 atoms in their collective ground state. Because of the quantum nature of the constituent particles, their collective angular momentum is necessarily quantized, meaning that a flask of superfluid He-4 can only rotate with certain allowed values of the angular momentum - an example of quantum mechanics observed at a traditionally classical scale. Furthermore, quantum fluids flow with

zero viscosity and have zero entropy, meaning that they do not experience friction as normal liquids do.

In He-4 specifically, the superfluid phase is complemented by a normal component, leading to the two-phase model described earlier. This duality gives rise to two kinds of 'sound' waves: a density wave, propagated by fluctuations in pressure throughout both components of the liquid, and a fluctuation of entropy within the normal fluid component, driven by a temperature gradient. The former scenario corresponds to 'first' sound, in which the normal and superfluid components are in-phase, while the latter is the 'second' sound indicative of a two-fluid system, in which the normal and superfluid components are out-of-phase with one another. Because of its role in the discoveries pertaining to superfluid He-4, second sound has been described extensively [23].

Related to the two-component description of ultracold He-4 is the excitation spectrum of the superfluid, again established by Landau [21]. Two primary types of excitations can be seen in the dispersion curve [25] - low energy phonons and higher energy rotons. The phonons are merely longitudinal fluctuations in density, while the rotons describe the backflow of helium caused by a particle passing through it with a certain threshold velocity (as a corollary, this means that particles moving with a velocity less than the threshold cannot dissipate energy into the superfluid). The behavior of rotons as related to the generation of quantum vortices in superfluid He-4 was theorized by Feynman [26] in the 1950s, although there exist alternative characterizations of rotons as bound He-clusters [27]. Nevertheless, quantized vortex rings were found experimentally by Rayfield and Reif [28] not long after their predicted existence by Feynman, and have since become an active field of research in and of themselves [29, 30, 31, 32].

1.3 Helium Spectroscopy

The physics alone of superfluid He-4 has offered a unique playground for scientists over the past century; however, it has also gained more recent popularity as a quantum solvent in many spectroscopic experiments. Atomic He-4 is incredibly stable, having only two paired electrons in a closed 1s-shell, meaning that it is completely chemically inert and only weakly interacts with other atoms (including other helium atoms) through slight van der Waals forces. It is exactly this property that allows helium to maintain its fluidity at the lowest of temperatures, and is also what makes He-4 such an excellent cryogenic matrix for a wide variety of guest atoms and molecules. Indeed, there exists a broad class of spectroscopies known as HENDI (**HE**lium **N**ano**D**roplet **I**solation) spectroscopies that take advantage of the fluid behavior of ultracold helium droplets to solvate and stabilize dopant complexes of interest, even if the dopants are otherwise unstable at room temperature. Helium droplets are also transparent to most forms of radiation, meaning that HENDI experiments can be used to probe both electronic and rovibrational transitions of solvated molecules.

In general, HENDI techniques begin by generating a beam of helium droplets from the rapid expansion of an ultracold gas or liquid source into a vacuum (see for example, Fig. 2 of [33]) which, in the former case at least, provides a beam of droplets with a lognormal size distribution [34]. The droplets then collide with the gaseous-phase dopant to be collected, whereupon the dopant-droplet system rapidly cools to the original temperature of the droplet, 0.37 K [35, 36]. Optical excitation of the resident atom/molecule then evaporates helium atoms from the droplet, with the number of atoms evaporated being determined by the photon energy and the helium chemical potential, and measurement of the size of the droplet after evaporation yields information as to the energy of the excitation. Because the measurement being made in these experiments is the helium evaporation from droplets due to an injection of energy, they are not restricted to single molecule excitations - reaction energetics can be studied via collection of multiple reactive

species within the droplet. This methodology has been used to examine the binding of water clusters [37], C+H_2 reactions [38], and reactions involving alkaline earth metals [39, 40], to name but a few. Helium droplet experiments have also been used to observe other interesting physics, including the dependence of superfluidity on He-cluster size [33], creation of ion-induced helium 'snowballs' [41], surface-solvated versus interior-solvated impurities [42], and even as a way to hypothetically observe the fission of an electronic wavefunction [43]. The interested reader is directed to comprehensive reviews of helium droplet spectroscopic experiments available in the literature [44, 45].

Understanding the dependence of the energetics of helium droplets on their size is one of the primary goals of this work. The practice of calculating an imparted energy from droplet distributions before and after collision with foreign particles is nontrivial, and current assumptions do not account for finite droplet size when predicting energetic quantities. We seek to improve upon these assumptions by tackling the helium-dopant system theoretically, as described in detail in Chapter 3. A brief survey of two of the most popular theoretical methods is presented in the subsequent sections of this chapter.

1.4 Quantum Monte Carlo Techniques

Often considered the gold standard of many-body quantum calculations, quantum Monte Carlo (QMC) methods are a class of numerical techniques that evaluate the Schrödinger equation by probabilistically sampling the full vector space of a system and returning averages of its physical properties. Interacting helium particles provide excellent cases for study via QMC approaches due to the closed shell, spherical nature of the ground state atoms, low-energy bound state of the dimer, and bosonic nature of the wavefunction; it is therefore no coincidence that He-4 systems were among the first ones to which QMC methods were applied[46].

Early QMC simulations usually adopted a variational procedure for calculating the ground state many-body wavefunction from an initial trial function with properties appropriate to the system - in the case of He-4 liquids, the Jastrow-McMillan form of the trial wavefunction [46] was (and still is) often used (Eq. 1.1). Also necessary to these calculations is a choice of the He-He potential interaction function; Lennard-Jones descriptions were popular due to their simplicity (Eq. 1.2), although modern calculations prefer the more accurate Aziz potentials [47, 48] (the forms of which are cumbersome and thus omitted from this document).

$$f(\mathbf{r}) = exp[-(a_1/\mathbf{r})^{a_2}] \tag{1.1}$$

$$V(\mathbf{r}) = 4\epsilon\Big[\Big(\frac{\sigma}{\mathbf{r}}\Big)^{12} - \Big(\frac{\sigma}{\mathbf{r}}\Big)^{6}\Big] \tag{1.2}$$

Because of the variational nature of the algorithm, multiple simulations must be run for various guesses of the wavefunction parameters a_1 and a_2; the values that minimize the system energy $\langle E \rangle$ become the parameters of the variational ground state (an approximation to the true ground state of the system, but necessarily higher in energy according to the variational principle). The optimized wavefunction can then be used to generate many configurations representative of the variational ground state according to an accept-reject criterion based on changes in probability density, which in turn are used to calculate the desired average ground-state properties.

The method described above is known as variational Monte Carlo (VMC), and constitutes the simplest of the quantum Monte Carlo techniques; however, its ability to accurately model He-4 systems is inhibited by the variational principle - indeed, the pioneering results of McMillan [46] showed an 18% difference in calculated ground state energy and 12% difference in equilibrium density relative to experiment. Admittedly, these values were later improved by Schmidt et al. [49] by employing the then-known Aziz HFDHE2 potential and including a term to describe three-body correlations, although these results still suffered from inaccuracies from the variational

procedure. Modern Monte Carlo methods have improved beyond the limitations of the variational principle by eliminating the variational procedure entirely, opting instead for an imaginary time propagation algorithm (the details of which are discussed in later chapters, in the context of He-DFT). There are three such classes of QMC techniques: Green's Function Monte Carlo (GFMC), Diffusion Monte Carlo (DMC), and Variational Path Integral Monte Carlo (VPI-MC, or just VPI), although it is most relevant to discuss the latter here - for reviews of the other two, the reader is directed to external sources [50, 51].

VPI, also known as path integral ground state (PIGS) Monte Carlo, is a premier tool for calculating ground state properties of bosonic quantum systems. It is a method used at zero temperature, and exactly solves for the ground state many-body wavefunction by means of damping out the excited state contributions from a trial (many-body) wavefunction via propagation through imaginary time. By considering the initial trial wavefunction for an interacting system of He-4 atoms as a 'bead', M successive applications of the imaginary time propagator to the first bead generates an additional M beads, each representing the many-body wavefunction at a particular slice in imaginary time. By ensuring that M is a sufficiently large number, beads further along the chain begin to sample the ground state of the system by virtue of the imaginary time propagator. Then, the Monte Carlo process simply proceeds by attempting to move atoms one-by-one within each bead with an accept/reject criterion related to the change in energy of the system - a decrease in energy results in an accepted move, while an increase in energy is conditionally accepted with a probability inversely related to its magnitude. Average system properties are computed over many Monte Carlo moves, and convergence to the ground state properties can be checked by comparing the results obtained from each bead. For the interested reader, more rigorous descriptions of the VPI algorithm exist in the literature [52, 53, 54].

The strengths of the VPI algorithm are numerous - the exact ground state many-body probability density and wavefunction can be sampled without any knowledge of their forms, and convergence to the ground state is insensitive to the initial trial

wavefunction chosen, provided the calculation is run for a sufficiently large number of beads [55]. Also, system properties other than the ground state energy can be accurately calculated, despite the non-commutation of the quantum mechanical operators for these properties with the Hamiltonian [53]. The primary drawback of this method, then, is its computational demand. Calculation times are on the order of days for systems containing $N \sim 10^2$ atoms, even with highly parallel processing. Inclusion of three-body terms, a necessity for accurate treatments of both solid and liquid He-4 [49, 56], further exacerbates this problem. It is clear, then, that an alternative method is needed to accurately compute the properties of large-scale superfluid He-4 systems ($N \sim 10^3 - 10^5$ atoms), given the current limits of present-day technology. We thus turn to the phenomenologically-based approach of helium density functional theory.

1.5 Helium Density Functional Theory

It is convenient to introduce the atom-centric helium density functional theory (He-DFT) in the context of the more commonplace electronic structure variant, as it wasn't formulated until after the quantum mechanical framework was established by Hohenberg and Kohn [57] and, subsequently, Kohn and Sham (an anomalous occurrence in method development - most quantum mechanical descriptions were derived after their classical counterparts). In electronic structure theory, the electron density $\rho_e(\mathbf{r})$ of a system uniquely determines its ground state properties, and minimizing the energy functional corresponds to finding said ground state electron density. The energy functional is completely described by the kinetic (E_K) and potential (E_V, E_{XC}, E_{ext}) components (terms in bold face are understood to be three-dimensional vectors):

$$E[\rho_e(\mathbf{r})] = E_K[\rho_e(\mathbf{r})] + E_V[\rho_e(\mathbf{r})] + E_{XC}[\rho_e(\mathbf{r})] + E_{ext}[\rho_e(\mathbf{r})] \tag{1.3}$$

Using the notation of Eq. 1.3, the potential energy is a sum of the electron-electron repulsions (E_V), fermionic exchange-correlation interactions (E_{XC}), and - in the context of atoms and molecules - electron-nucleus attractions (E_{ext}). Although formally exact, the true energy functional of any given system is at present neither known nor obtainable, as a complete description of the exchange and correlation interactions between electrons does not yet exist. Approximations to these terms can be made (the equivalent of "choosing" a functional), and a minimized electron density can be self-consistently calculated via some iterative procedure, although there is no guarantee that the calculated electron density is representative of the true ground state.

An analogous formalism can be constructed with principles that mirror the ones described above, using He-4 nuclei in place of electrons. In this picture, the ground state properties of a system are uniquely determined by the macroscopic He-4 density $\rho_{He}(\mathbf{r})$ - a trait that holds true because the helium atoms are all in the same ground state, and can thus be described by a macroscopic wavefunction. Note that this restricts He-DFT calculations to the case of zero absolute temperature; at finite temperatures the two-fluid model described in Sec. 1.2 becomes relevant, and must be accounted for in the system description. The energy functional for a system at $T = 0$ is the sum of the kinetic and potential terms, and reads as follows:

$$E[\rho_{He}(\mathbf{r})] = E_K[\rho_{He}(\mathbf{r})] + E_{He-He}[\rho_{He}(\mathbf{r})] + E_{ext}[\rho_{He}(\mathbf{r})] \tag{1.4}$$

Note that the same bosonic properties responsible for the macroscopic He-4 wavefunction are also responsible for the omission of the exchange term from the energy expression above. In its stead is the helium-helium interaction energy functional, E_{He-He}, which is loosely described by the two-body He-He pair potential; however, as it incorporates the He-4 equivalent of electronic correlation, its exact form is not currently known. The practical approach adopted by many authors is to construct a functional that incorporates the attractive portion of the He-He pair

potential and combines it with a self-interaction correction at small distances that is parameterized phenomenologically based on fitting to bulk He-4 data. Once a helium functional is chosen, an iterative procedure is adopted to calculate ground state properties from the self-consistent helium density profile. This is done in general by evaluating a Schrödinger-like equation (Eq. 1.5) that results from variationally minimizing Eq. 1.4 with respect to the square root of the density, $\sqrt{\rho(\boldsymbol{r})}$:

$$\left(\frac{-\hbar^2}{2m}\nabla^2 + U_{He-He} + V_{external}\right)\sqrt{\rho(\boldsymbol{r})} = \mu\sqrt{\rho(\boldsymbol{r})} \tag{1.5}$$

The terms on the left-hand side of Eq. 1.5 correspond to the first functional derivative of the total kinetic energy, helium-helium interactions, and external potential terms of Eq. 1.4. In particular, the choice of $E_{He-He}[\rho(\boldsymbol{r})]$ determines the form of U_{He-He}.

One of the earliest successful helium functionals was developed by Stringari and Treiner in 1987 as a tool for studying planar free films and spherical helium droplets [58]. Mathematically simple, the Stringari-Treiner (ST) functional uses zero-range Skyrme-type interactions and a single density gradient term to describe the helium system. This approach has made possible accurate calculations of the surface thickness, chemical potential, and bulk density of free slabs of He-3 and He-4, and has been used to predict the energetics of asymptotically large He-3 and He-4 droplets [58, 59]; however, it fails to capture solvation effects present in doped helium systems. This shortcoming, among others, was addressed in subsequent works by Dupont-Roc et al. (1990) [2], Barranco et al. (1994) [1], and Dalfovo et al. (1995) [3], yielding the Orsay-Paris (OP), Catalonia (Cat), and Orsay-Trento (OT) functionals, respectively. In each of these functionals, He-He correlation is treated beyond the zero-range description of Stringari and Treiner, making the functionals non-local in nature. Each also has a unique screened-core potential to correct for self-interaction error and, in the case of the OT functional, an additional corrective term to account for quantum backflow effects within the helium system.

One of the main goals of this work is to assess each of the functionals mentioned above in terms of their ability to accurately describe the system properties of He-4 adsorbed onto attractive planar surfaces and of doped He-4 droplets, as this is of great importance in ensuring high quality calculations.

Chapter 2

Planar He-4 Films and Attractive Surfaces

One of the primary objectives of this chapter is to ascertain the ability of the Stringari-Treiner and Orsay-Paris density functionals to accurately compute the ground state properties of helium films both in free space and adsorbed onto attractive surfaces. It is thus convenient to first establish a conceptual picture of the systems of interest. We begin by immediately reducing the dimensionality of the helium slab from a three-dimensional description to a one-dimensional one by considering only films that have an infinitely flat uniform surface in the $\vec{x}$ and $\vec{y}$ directions, and finite thickness perpendicular to the surface in the $\vec{z}$ direction. Although this simplification discards the edge effects otherwise felt by a real helium slab, it allows us to probe the density only along the $\vec{z}$ axis (i.e. the direction in which it is allowed to vary), greatly reducing the required computational resources. A diagram depicting this setup can be found in Fig. 2.1. Note that because the helium slab is represented by a discretely sampled one-dimensional density function, there are no distinct atoms in the simulation.

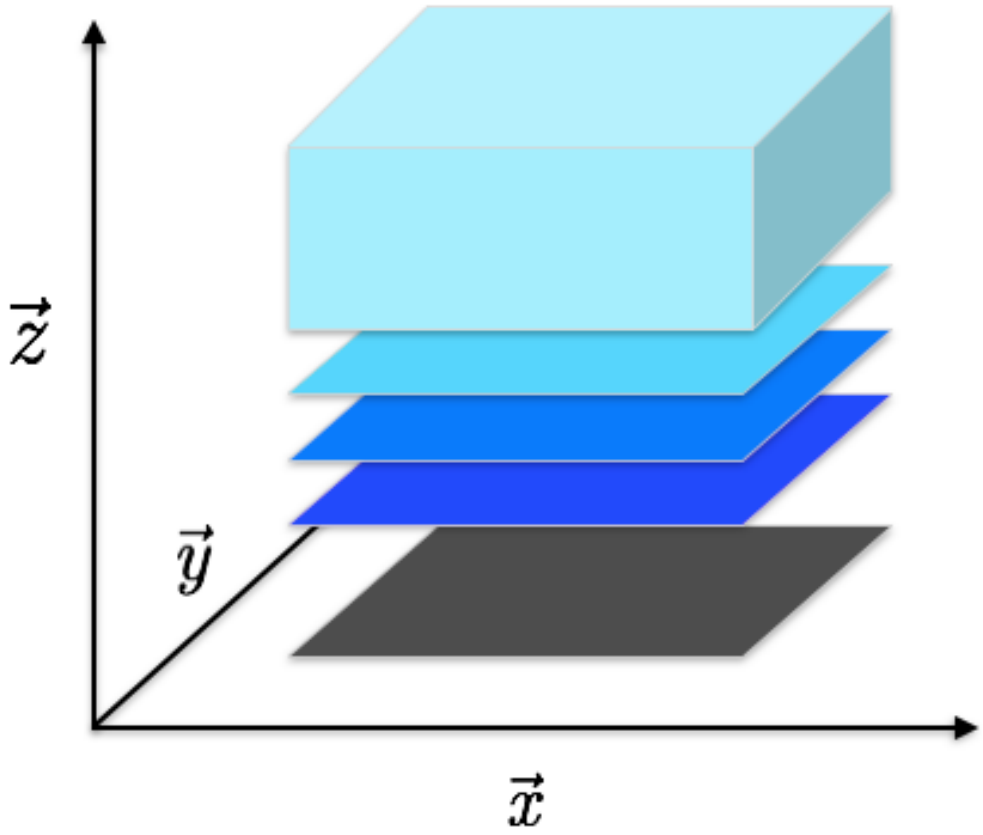


Figure 2.1: Cartoon depicting the one-dimensional sampling scheme of a free helium slab. The $\vec{x}$ and $\vec{y}$ directions are assumed to be infinite, while $\vec{z}$ spans the finite width of the slab. The grey area is representative of an attractive surface, while the shades of blue indicate the various helium densities that might be found in solvation layers adjacent to the surface. The lightest blue block at the top denotes bulk helium density.

2.1 Stringari-Treiner and Orsay-Paris Functionals

The earliest functional considered in this work is the zero-range Skyrme-type functional proposed by Stringari and Treiner [58], whose form is seem below for the case of a planar 1-D helium slab:

$$U_{He-He}(z) = b\rho(z) + \frac{2+\gamma}{2}c\rho^{1+\gamma}(z) - 2d\nabla^2\rho(z) \tag{2.1}$$

The values of the constants b, c, d, and γ are equal to -8.88810×10^2 KÅ^3, 1.04554×10^7 KÅ$^{3(1+\gamma)}$, 2.383×10^3 KÅ^5, and 2.8, respectively, and together compose the sum of the helium-helium interactions. The first ("b") and second ("c") terms represent the density-dependent long-range attractions and short-range repulsions, respectively, while the third ("d") term - nonzero only when the helium density varies - represents a repulsive surface contribution.

Modifications to this form of the helium functional come from Dupont-Roc et al. [2], who account for finite range helium-helium correlations by improving the "b" and "c" terms of Eq. 2.1, while omitting the "d" term entirely. This functional, referred to as the Orsay-Paris functional in the literature, has the general form seen in Eq. 2.2.

$$E_{He-He}(\boldsymbol{r}) = \frac{1}{2}\int \rho(\boldsymbol{r}')V_{OP}(|\boldsymbol{r}-\boldsymbol{r}'|)d\boldsymbol{r}' + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} \tag{2.2}$$

The "b" term - again representing long-range attraction (albeit without the actual parameter "b") - is now an integral over the entirety of the density profile, where interaction between the densities at coordinates $\boldsymbol{r}$ and $\boldsymbol{r}'$ is a function of their separation:

$$V_{OP}(|\boldsymbol{r}-\boldsymbol{r}'|) = \begin{cases} 4\epsilon\left[\left(\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|}\right)^{12} - \left(\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|}\right)^{6}\right] & |\boldsymbol{r}-\boldsymbol{r}'| \geq h_{OP} \\ V_{OP}(h_{OP})\left(\frac{|\boldsymbol{r}-\boldsymbol{r}'|}{h_{OP}}\right)^4 & |\boldsymbol{r}-\boldsymbol{r}'| < h_{OP} \end{cases} \tag{2.3}$$

The new parameter h_{OP} is a cutoff distance calculated by ensuring that the integral of Eq. 2.3 over all space recovers the value of the "b" parameter of Stringari

and Treiner, and further separates the interaction term into what are essentially attractive and repulsive components. The attractive portion ($|\boldsymbol{r} - \boldsymbol{r}'| \geq h_{OP}$) is simply the Lennard-Jones two-body potential for helium-helium interactions, with de Boer parameters $\sigma = 2.556$ Å and $\epsilon = 10.22$ K [60]. The region in which $|\boldsymbol{r}-\boldsymbol{r}'| < h_{OP}$ can be thought of as a self-interaction correction; as the distance between the two points $\boldsymbol{r}$ and $\boldsymbol{r}$' becomes vanishingly small, both sample the density profile at points within the radius of a single helium atom, meaning the potential should converge to zero.

The "c" term retains a form similar to that of the Stringari-Treiner functional, but now depends on a coarse-grained density term $\bar{\rho}(\boldsymbol{r})$:

$$\bar{\rho}(\boldsymbol{r}) = \int \rho(\boldsymbol{r}')\Xi(|\boldsymbol{r} - \boldsymbol{r}'|)d\boldsymbol{r}' \tag{2.4}$$

The $\Xi(|\boldsymbol{r} - \boldsymbol{r}'|)$ function is the coarse-graining function responsible for setting the limits for which $\bar{\rho}(\boldsymbol{r})$ is defined:

$$\Xi(|\boldsymbol{r} - \boldsymbol{r}'|) = \begin{cases} \frac{3}{4\pi h_{OP}^3} & |\boldsymbol{r} - \boldsymbol{r}'| \leq h_{OP} \\ 0 & |\boldsymbol{r} - \boldsymbol{r}'| > h_{OP} \end{cases} \tag{2.5}$$

Note that the same cutoff parameter h_{OP} responsible for separating the repulsive and attractive parts of the "b" term also dictates the bounds of $\Xi(|\boldsymbol{r} - \boldsymbol{r}'|)$.

Differentiation of the general energy functional (Eq. 2.2) with respect to the density function $\rho(\boldsymbol{r})$ and subsequent integration over the $\vec{x}$ and $\vec{y}$ coordinates leads to a form applicable to the 1-D planar systems studied in later sections [2, 61]:

$$U_{He-He}(z) = 4\pi\epsilon\sigma^2\Big(\int_{-\infty}^{z-h_{OP}} + \int_{z+h_{OP}}^{\infty}\Big)\rho(z')\Big(\frac{\sigma}{z-z'}\Big)^4\Big[\frac{1}{5}\Big(\frac{\sigma}{z-z'}\Big)^6 - \frac{1}{2}\Big]dz'$$
$$+\ 4\pi\epsilon\sigma^2\int_{z-h_{OP}}^{z+h_{OP}}\rho(z')\Big(\frac{\sigma}{h_{OP}}\Big)^4\Big\{\Big[\frac{8}{15}\Big(\frac{\sigma}{h_{OP}}\Big)^6 - \frac{5}{6}\Big] - \frac{1}{3}\Big(\frac{z-z'}{h_{OP}}\Big)^6\Big[\Big(\frac{\sigma}{h_{OP}}\Big)^6 - 1\Big]\Big\}dz'$$
$$+\ \frac{c}{2}[\bar{\rho}(z)]^{\gamma+1} + \frac{3c}{8h_{OP}}(\gamma+1)\int_{z-h_{OP}}^{z+h_{OP}}\rho(z')[\bar{\rho}(z')]^{\gamma}\Big[1 - \Big(\frac{z-z'}{h_{OP}}\Big)^2\Big]dz' \quad (2.6)$$

The remaining integrals along the z-coordinate cannot be computed analytically, as the integrand in each case contains the density function $\rho(z')$, whose form is not explicitly known. We thus resort to using numerical trapezoid rule integration for these terms, mitigating the error induced from the approximation by keeping the number of sampling points N large.

2.2 Self-Consistent Methods

In order to calculate the desired ground state properties of the helium systems explored in this work, an iterative process to update the trial helium density self-consistently is necessary. We evaluate two such processes for this purpose: a 'mixing' method in which the density of a given iteration is updated fractionally with the next calculated density until it is self-consistent, and an imaginary time propagation (ITP) method in which the trial density is propagated through imaginary time until the excited components are damped out and only the ground state remains. In each case, the initial density has the one-dimensional form of Eq. 2.7, established by Stringari and Treiner:

$$\rho_{init}(z) = \frac{\rho_0}{[1 + exp(z/a)]^{\nu}} \quad (2.7)$$

In this equation, ρ_0 represents the theoretical bulk density (2.1836×10^{-2} Å^{-3}) of He-4, while a and ν are parameters determined phenomenologically by the surface

thickness, defined as the distance necessary to reduce the helium density from 90% to 10% along the $\vec{z}$ direction. These values are found to be 1.96 Å and 2.5, respectively. Note that Eq. 2.7 only describes one-half of the free helium slab; its mirror image must also be constructed to complete the system.

Both self-consistent algorithms are initialized with this starting density profile, after which the iterative procedure differs for each method. In the 'mixing' method, the mean field potential is computed according to the choice of functional, described in Section 2.1, and the most up-to-date helium density profile. The kinetic energy is subsequently calculated utilizing the second-order central difference theorem to approximate the second derivative of the helium 'wavefunction' with respect to distance coordinate (Eq. 2.8).

$$\left(\frac{d^2\psi}{dz^2}\right) \approx \frac{\sqrt{\rho(z+\Delta z)} - 2\sqrt{\rho(z)} + \sqrt{\rho(z-\Delta z)}}{\Delta z^2} + O(\Delta z)^2 \tag{2.8}$$

The kinetic and potential energy terms are then assembled into a tridiagonal matrix - representative of the Hamiltonian of the system - in which the diagonal elements consist of the sum of the potential energy and the central term of Eq. 2.8, and the super- and sub- diagonal elements are simply the first and third terms of Eq. 2.8, respectively. By using an external Fortran package to calculate the eigenvalues and eigenvectors of the tridiagonal Hamiltonian matrix (we use LAPACK, the Linear Algebra PACKage), the chemical potential and new helium 'wavefunction' are obtained. By normalizing and squaring the new 'wavefunction', a new helium density profile is calculated and subsequently blended with the density profile from the beginning of the iteration via simple point-to-point addition. The degree of mixing is governed by an adjustable mixing parameter $\xi \in [0, 1]$, where $\xi = 0$ is a complete replacement of the old density profile with the new one. This final blended density profile then is used as a starting point for the next iteration, and the calculation proceeds until a self-consistent profile is obtained. We check for self-consistency by

performing a simple point-to-point differencing of the initial and final density profiles within each iteration. A diagram outlining this method is seen in Fig. 2.2.

As an aside, it can be noted that the second-order approximation to the kinetic energy can be replaced by any higher order finite difference approximation; we found no appreciable difference in the results obtained after upgrading to the fourth-order central difference approximation using the above mixing method.

Before describing the particular ITP algorithm used in this work [62], it is first instructive to introduce the ITP method conceptually. The core of this approach takes advantage of the fact that the two generic operators $\hat{A}$ and $e^{(\hat{A})}$ share a set of eigenfunctions ψ_i. Furthermore, the eigenvalues a_i of each operator are related to one another (Eqs. 2.9 and 2.10).

$$\hat{A}\psi_i = a_i\psi_i \tag{2.9}$$

$$e^{(\hat{A})}\psi_i = e^{(a_i)}\psi_i \tag{2.10}$$

By taking the operator $\hat{A}$ to be the Hamiltonian $\hat{\mathcal{H}}$, Eqs. 2.9 and 2.10 can be rewritten for the helium system:

$$\hat{\mathcal{H}}\sqrt{\rho(z)} = \mu\sqrt{\rho(z)} \tag{2.11}$$

$$e^{(\hat{\mathcal{H}})}\sqrt{\rho(z)} = e^{(\mu)}\sqrt{\rho(z)} \tag{2.12}$$

The other component necessary to the ITP approach is the establishment of the mathematical imaginary time propagator itself. Put simply, it is the operator that, when given an initial wavefunction $\psi(\boldsymbol{r}, \tau = 0)$, returns the wavefunction after advancement in imaginary time, $\psi(\boldsymbol{r}, \tau)$. Its form can be guessed at from analysis of the time-dependent Schrödinger Equation:

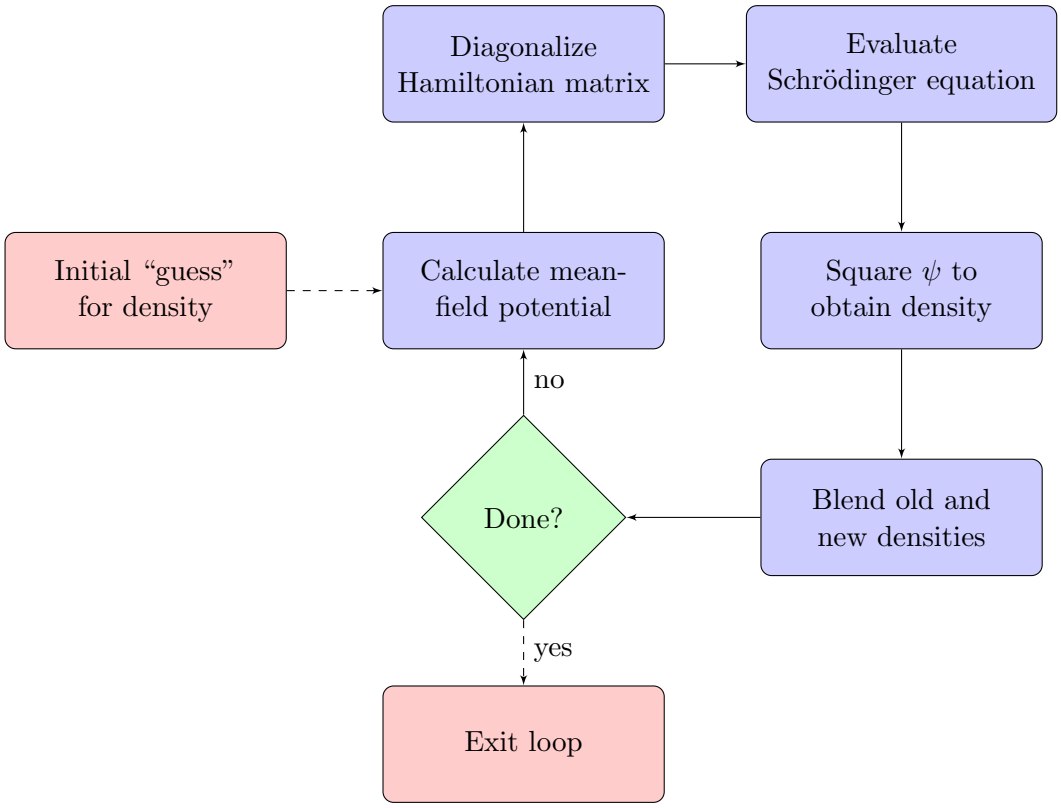


Figure 2.2: Pictorial representation of the 'mixing' algorithm used in our density functional calculations for planar slabs of helium.

$$\frac{\partial \Psi(\boldsymbol{r},t)}{\partial t} = i\hbar\hat{\mathcal{H}}\Psi(\boldsymbol{r},t) \tag{2.13}$$

By making the substitutions $\tau = it$ and $d\tau = idt$, Eq. 2.13 becomes:

$$\frac{\partial \Psi(\boldsymbol{r},\tau)}{\partial \tau} = \frac{-\hat{\mathcal{H}}}{\hbar}\Psi(\boldsymbol{r},\tau) \tag{2.14}$$

By inspection, the form of the propagator must be of exponential form with respect to τ, as $\Psi(\boldsymbol{r},\tau)$ is a term in its own derivative. Applying this to Eq. 2.14 with the definition of the propagator from above yields Eq. 2.15:

$$\Psi(\boldsymbol{r},\tau) = e^{(-\hat{\mathcal{H}}\tau/\hbar)}\Psi(\boldsymbol{r},\tau=0) = [e^{(-\hat{\mathcal{H}}\Delta\tau/\hbar)}]^n\Psi(\boldsymbol{r},\tau=0) \tag{2.15}$$

Note that in the final equality of Eq. 2.15, a discretization of the imaginary time variable is enacted ($\tau = n\Delta\tau$); the resultant term $\Delta\tau$ is referred to as the imaginary time step parameter, and represents a finite advancement of imaginary time. Thus, it is evident that repeatedly applying the imaginary time propagator to a wavefunction beginning at $\tau = 0$ yields the ground state wavefunction as $\tau \to \infty$. We then approximate $e^{(-\hat{\mathcal{H}})}$ as $e^{(-\frac{1}{2}\hat{V})}e^{(-\hat{T})}e^{(-\frac{1}{2}\hat{V})}$, a process known as the Trotter factorization, so that the ITP equation is applicable to our helium systems in a straightforward manner:

$$\sqrt{\rho(z,\tau)} = e^{(-\frac{\Delta\tau}{2\hbar}\hat{U}_{He})}e^{(-\frac{\Delta\tau}{\hbar}\hat{T}_{He})}e^{(-\frac{\Delta\tau}{2\hbar}\hat{U}_{He})}\sqrt{\rho(z,\tau=0)} \tag{2.16}$$

The degree of error introduced by the Trotter factorization is of the order $O(\Delta\tau^3)$, as evidenced by an analysis of the Taylor expansions of each of the expressions. More accurate (i.e. $O(\Delta\tau^5)$) factorizations of the Hamiltonian into a product of its kinetic and potential components exist [62], however, these also require an evaluation of the nested commutator [V,[T,V]], a task that is numerically and computationally complex. To ensure the second-order splitting is sufficiently accurate, we calculate the energy of the n=2 state of the electron-in-a-box system from a composite n=2 +

n=4 wavefunction - the resultant energy after 5000 iterations (19.7392080 Hartrees) agrees with the theoretical value (19.7392081 Hartrees) to six decimal places, and is self-consistent to arbitrary precision.

By considering Eqs. 2.15 and 2.16, the imaginary time algorithm used for our helium systems can be generated. First, the mean field helium potential for a given iteration is calculated and enters the right-most term of Eq. 2.16 as $\hat{U}_{He}$. Next, the kinetic energy operator $\hat{T}_{He}$ is evaluated - we take advantage of the fact that this component is easily computed in momentum space, where the kinetic energy operator $\hat{T}_{He}$ is simply $\frac{\hat{p}^2}{2m}$. The conversion from coordinate to momentum spaces (and back again) is performed via the fast Fourier transform, or FFT (inverse fast Fourier transform, iFFT) [63], a computationally efficient version of the more familiar Fourier transform (inverse Fourier transform). The general Fourier transform and its inverse are seen in Eq. 2.17:

$$\begin{aligned} \mathcal{F}[g(\boldsymbol{r})] = G(\boldsymbol{p}) &= \int_{-\infty}^{\infty} g(\boldsymbol{r}) e^{-2\pi i \boldsymbol{p}\boldsymbol{r}} d\boldsymbol{r} \\ \mathcal{F}^{-1}[G(\boldsymbol{p})] = g(\boldsymbol{r}) &= \int_{-\infty}^{\infty} G(\boldsymbol{p}) e^{2\pi i \boldsymbol{p}\boldsymbol{r}} d\boldsymbol{p} \end{aligned} \tag{2.17}$$

The generic functions $g(\boldsymbol{r})$ and $G(\boldsymbol{p})$ are a Fourier pair, and in our calculations correspond to the position and momentum space representations of the helium wavefunction, respectively. The numerical equivalent to Eq. 2.17 is the Fourier series, seen in Eq. 2.18:

$$\begin{aligned} \mathcal{F}[g_r] = G_p &= \sum_{r=0}^{N-1} g_r e^{-2\pi i p r/N} \\ \mathcal{F}^{-1}[G_p] = g_r &= \frac{1}{N} \sum_{p=0}^{N-1} G_p e^{2\pi i p r/N} \end{aligned} \tag{2.18}$$

The sets of points g_r and G_p are the discretized versions of the functions $g(\boldsymbol{r})$ and $G(\boldsymbol{p})$ from Eq. 2.17, respectively. This algorithm scales with the square of the number of sampling points N^2, while the aforementioned FFT/iFFT method used in our calculations (which is really the same calculation with a more efficient arrangement of data) exhibits an improved $Nlog(N)$ scaling.

After evaluation of the kinetic energy operator in momentum space, a transform back into coordinate space is performed, whereupon the second half of the potential energy $\hat{U}_{He}$ is calculated. Normalization of the resultant wavefunction after the three operations yields a new wavefunction more representative of the true ground state for the beginning of the next iteration. For clarity, a diagram detailing the ITP algorithm can be found in Fig. 2.3.

2.3 Attractive Planar Surfaces

In order to accurately model realistic surfaces interacting with adsorbed helium films, we find it necessary to first evaluate the mechanics of our underlying code. Beginning with the 'mixing' procedure, the first of these diagnostic tests involves probing the effect of mixing parameter on the convergence and accuracy of the resultant helium density profiles. It is straightforward to envision the qualitative impacts of using a mixing parameter outside the 'optimal' range - the convergence will either be very poor (ξ too small) or occur on an undesireably long timescale (ξ too large). By manually adjusting this value over a series of calculations, we find an optimal range to be $\xi \in [0.994, 0.998]$, meaning that 99.4-99.8% of the old density is retained after an iteration. The perhaps obvious second check of the mixing algorithm lies in its ability to successfully reproduce the ground state density profile of a simple free slab system. Stringari and Treiner report a range of density values with respect to bulk for a free slab of He-4; our code is capable of reproducing these data employing the same functional (Fig. 2.4).

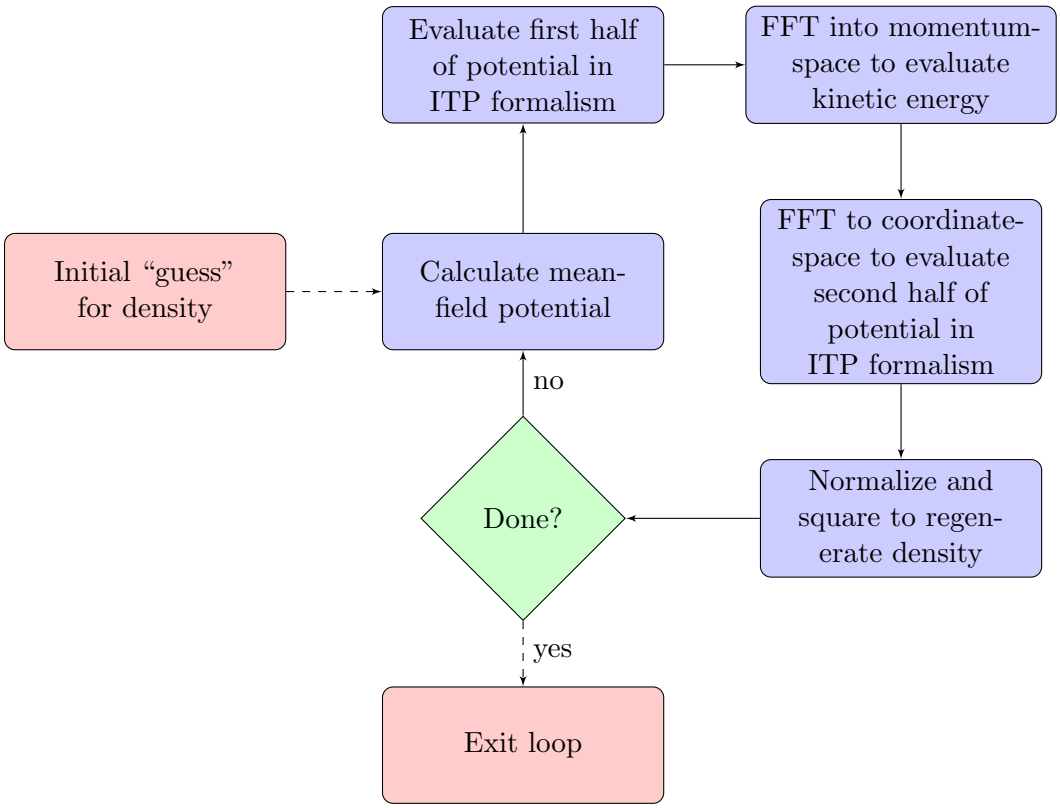


Figure 2.3: Pictorial representation of the ITP algorithm used in our density functional calculations for planar slabs of helium. FFT = Fast Fourier Transform

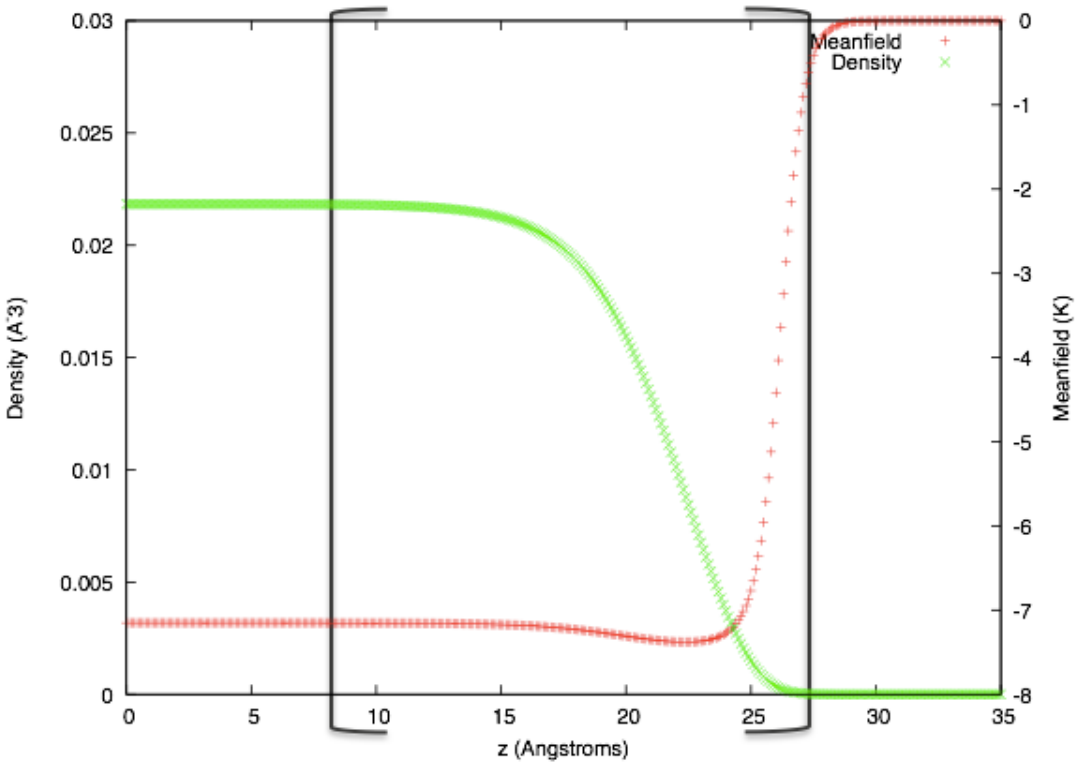


Figure 2.4: Density profile of a free planar slab of He-4 calculated using the Stringari-Treiner functional (green, left axis). Also shown is the corresponding mean field potential (red, right axis). The region bound by the black brackets contains the points that were checked to reproduce available literature data.

Upon verifying the accuracy of the free slab results using the 'mixing' scheme, a simple Lennard-Jones type surface was introduced at the center ($z = 0$) of the density profile (Eq. 2.19). Note that although this halves the number of sampling points available for calculation - the left half of the helium slab is discarded - the sampling resolution remains the same, meaning that the accuracy of the computation is not inhibited provided the number of points is sufficient to represent the full change in density.

$$V_{ext,LJ}(z) = 4\epsilon\left[\left(\frac{\sigma}{z} + l\right)^{12} - \left(\frac{\sigma}{z} + l\right)^{6}\right] \tag{2.19}$$

The 'attractiveness' of the artificial Lennard-Jones surface is governed by both ϵ and σ, as the former controls the depth of the attractive well, while the latter governs its width. Because sigma also controls the coordinate at which the repulsive wall vanishes, the parameter l is introduced to fix this location within the helium slab at $z = 2$ Å.

Two sets of calculations were then performed using the Stringari-Treiner and Orsay-Paris functionals, in which the value of ϵ varied from 5 K to 55 K in the former, and from 15 K to 45 K in the latter (10 K increments). The resultant density profiles can be found in Figs. 2.5 and 2.6, respectively. The effect of changing the value of the σ parameter was also tested using $\sigma = 3, 5.5$, and 7 Å; as this enacted only minor changes in the density profile, only the results using $\sigma = 7$ Å are reported. By contrasting Fig. 2.5 with Fig. 2.6, it is easily seen that the zero-range nature of the Stringari-Treiner functional oversimplifies the He-He interaction, and can only qualitatively capture the buildup of helium density indicative of a first solvation layer. Furthermore, it does not predict any layering effect beyond the initial density peak, but instead shows a gradual decay back to the bulk value. The Orsay-Paris functional, on the other hand, reflects the expected behavior of multiple helium solvation layers in the density profiles for each attractive surface. The failure of the Stringari-Treiner

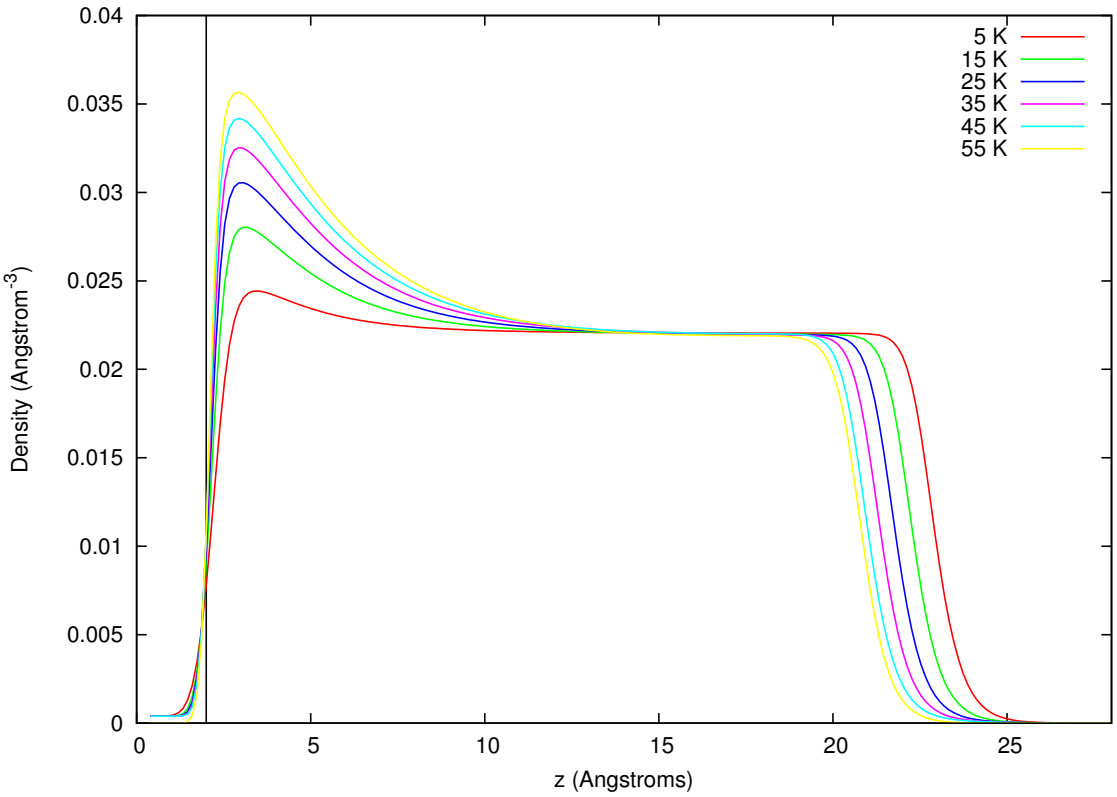


Figure 2.5: Helium density profiles for systems involving an artificial Lennard-Jones surface located at z=0 using the Stringari-Treiner functional. The vertical black line represents the LJ sigma parameter, where the He-surface interaction switches from repulsive to attractive.

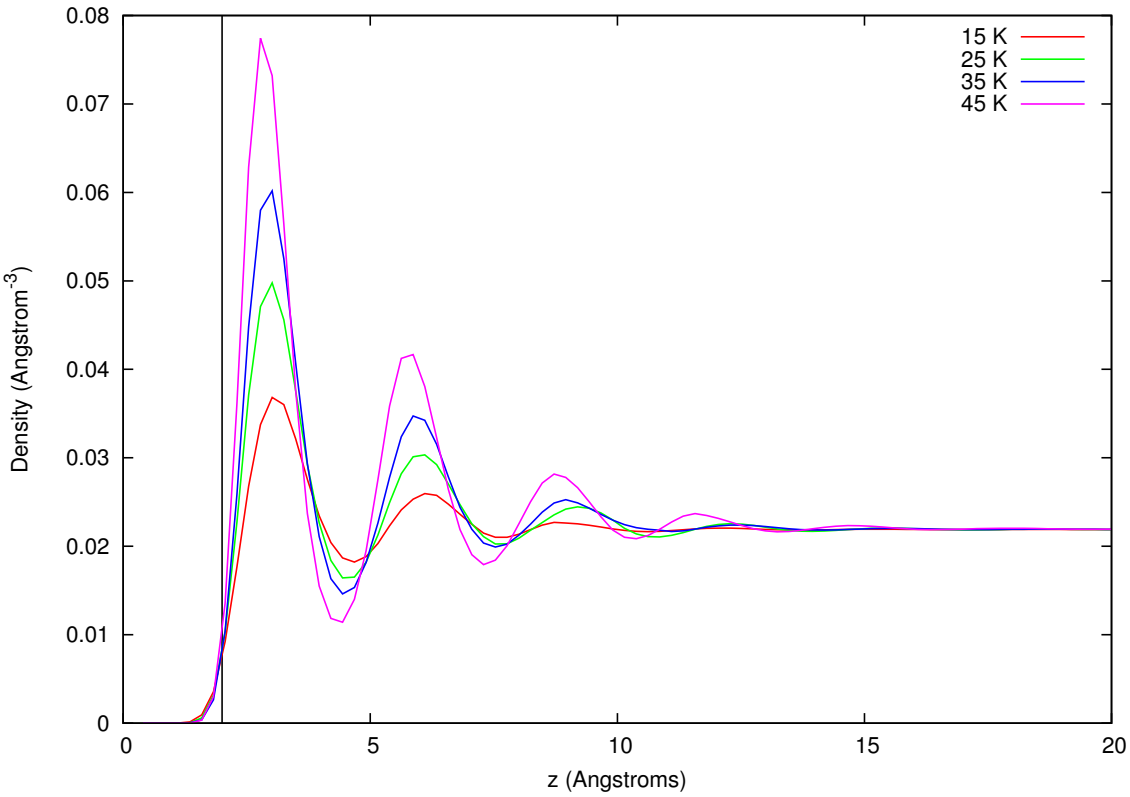


Figure 2.6: Helium density profiles for systems involving an artificial Lennard-Jones surface located at z=0 using the Orsay-Paris functional. The vertical black line represents the LJ sigma parameter, where the He-surface interaction switches from repulsive to attractive.

functional to capture the physics necessary to describe proper solvation effects thus leads us to abandon it in favor of the more able non-local functionals.

In order to be more quantitative with our analysis of the Orsay-Paris functional, it is necessary to introduce a surface that is more representative of a realistic system; however, because we reduce the dimensionality of the helium surface to a 1-D problem, this surface should ideally exist as a planar entity. To this end, graphite is an ideal choice of a simple realistic system - it lacks any bridging atoms and, because its composition is purely carbon, any appreciable surface corrugation due to variations in atomic size. The He-graphite interaction is of the form postulated by Carlos and Cole [64], in which the total potential is a sum over many laterally averaged carbon layers. The description of a single carbon layer is then simply the integral of the atomic He-C Lennard-Jones interaction over the carbon surface, and a subsequent summation of these layers to a desired thickness generates the full graphite slab. The resultant potential is expressed in Eq. 2.20 below, where the subscript 'g' refers to 'graphite', a_s indicates the surface area of the graphite unit cell (22.25 Å^2), and d is the interplane distance (3.37 Å):

$$V_g(z) = \frac{4\pi\epsilon\sigma^6}{a_s d^4}\left[\frac{2}{5}\left(\frac{\sigma}{d}\right)^6 \zeta(10, z/d) - \zeta(4, z/d)\right] \tag{2.20}$$

$$\zeta(n, z/d) = \sum_{j=0}^{\infty} (j + z/d)^{-n} \tag{2.21}$$

The σ and ϵ parameters again control the location and depth of the attractive Lennard-Jones well, respectively, and have values of 2.78 Å and 1624.64 K. The Riemann zeta function as it is shown in Eq. 2.21 accounts for the behavior of an infinitely thick slab; in practice the summation variable j is truncated at some large value, indicating that j layers of the graphite slab are to be accounted for.

Simulations of helium films adsorbed onto the graphitic surface described by Eqs. 2.20 and 2.21 were performed with N=1600 sampling points and 6000 iterations, using the 'mixing' algorithm to obtain the converged density profiles. These

resultant profiles can then be used to calculate physical properties, such as surface-to-solvation-layer distances and individual helium-layer densities, which in turn are directly compared to data from neutron scattering experiments determining the same properties. As seen in Fig. 2.7, the Orsay-Paris functional predicts a first-layer density of 0.0981 $Å^{-2}$ and a combined first- and second-layer density of 0.174 $Å^{-2}$ - evaluated by integrating the area under the first and first+second peaks, respectively - corresponding to a 17.6 and 18.8 percent difference, respectively, from the neutron studies of Carneiro et al. [13] The surface-to-first-solvation-layer distance is also predicted to be 3.01 Å, corresponding to a 3.7 percent difference from the experimental value. As an attempt to improve these values by correcting for the low resolution of the helium density in the first peaked region, linear extrapolations were performed along the left and right sides of the peak, with their intersection becoming the new corrected maximum value. Incorporating this extra point in the surface area calculations yields new values of 0.124 $Å^{-2}$ and 0.200 $Å^{-2}$ for the first- and first-plus-second layer densities, respectively, thus improving agreement with experiment to 5.81 and 4.88 percent. Taking the z-coordinate of the new corrected point to be the improved graphite-helium distance also shows better agreement with the experimental value, with a discrepancy of only 0.05 Å.

Although these results validate our decision to use the Orsay-Paris functional for non-homogeneous helium systems, they also reveal a weakness in the ability of the 'mixing' method to converge to a self-consistent density, as seen in Fig. 2.8. We refer to this as the problem of helium "sloshing", and it occurs in our simulations as the helium density builds near the graphite surface. As the density increases in this region, so does the He-He repulsive portion of the meanfield potential, causing the next iteration's density profile to be pushed away from the graphite surface. This then causes the region near the graphite surface to be especially attactive in the iteration after that, as the He-He repulsion term is particulary small due to the density having "sloshed" in the previous iteration. The error incurred by this artificial phenomenon can be mitigated by increasing the mixing parameter ξ, so that the percentage of the

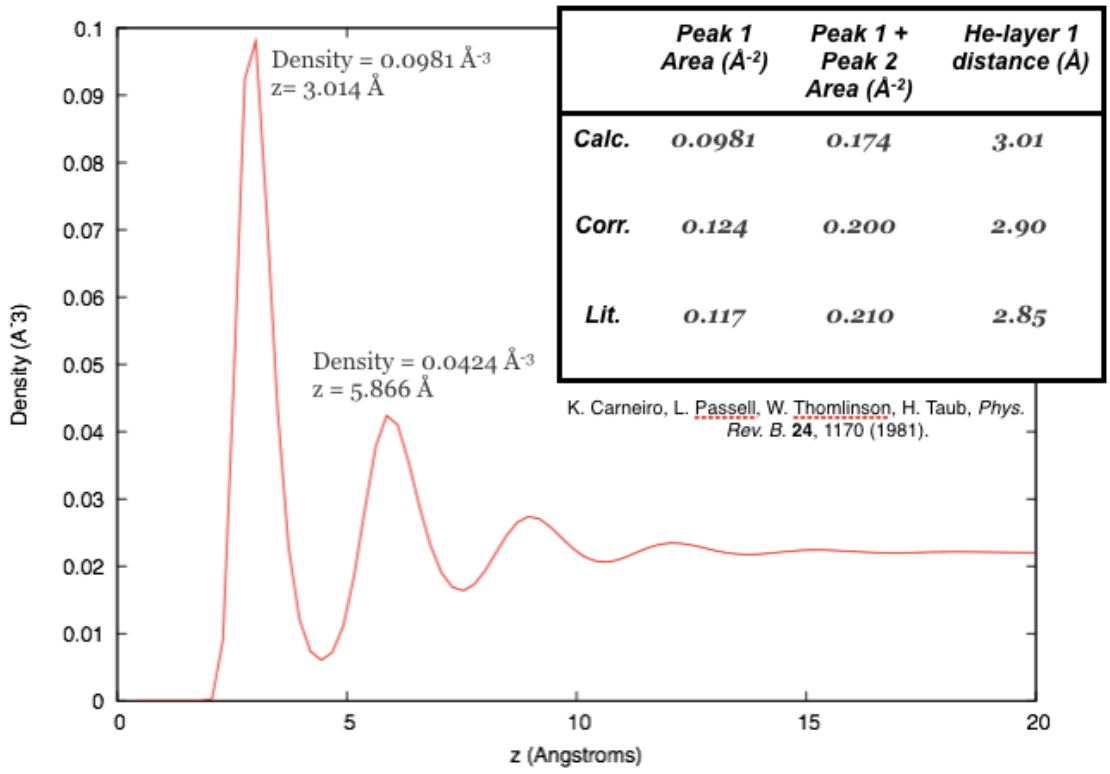


Figure 2.7: Solvation layer region of a He-4 density profile adsorbed to a planar graphite surface, calculated using the Orsay-Paris functional and the 'mixing' method. The graphite surface is located at z=0. Inset: Computed values for the first and first+second solvation layer densities, as well as for the He-graphite distance. The literature values represent neutron studies performed by Carneiro et al. [13]

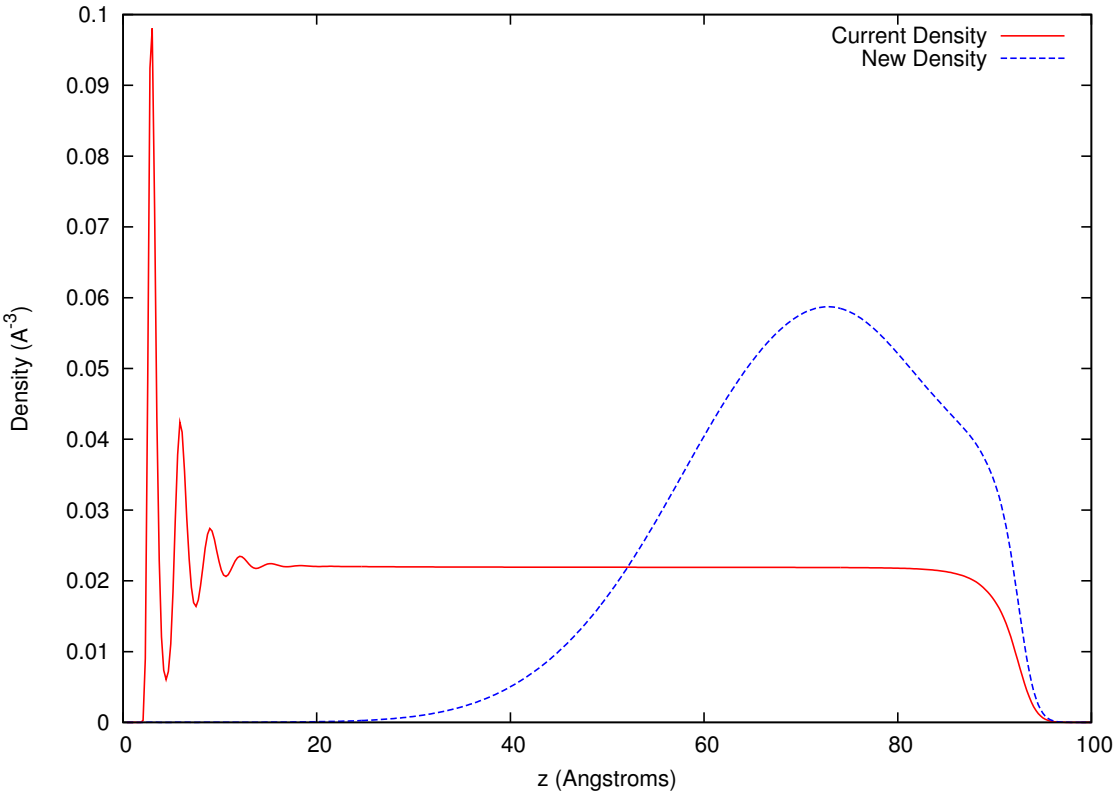


Figure 2.8: Consecutive density profiles of helium adsorbed onto a graphite surface, calculated using the 'mixing' method. The red line represents the nearly-converged density of an arbitrary (m-1)th iteration, while the dashed blue line shows the newly calculated density profile of iteration m. Note that this density has been pushed away from the surface entirely; we refer to this as the 'sloshing' of sequentially calculated density profiles, and it is indicative of a convergence issue with the 'mixing' method as a whole.

new "sloshed" profile introduced to the old one is very small; however, it is still the cause of the greatest uncertainty in the converged chemical potential value μ.

As an attempt to circumvent these convergence issues, we look to replace the 'mixing' procedure with the ITP algorithm described in Section 2.2. Prior to simulating the graphite system, however, preliminary calculations were run on bulk free surfaces to determine both an acceptable sampling mesh size and an appropriate value for the imaginary-time step, $\Delta\tau$. An immediate realization of the superiority of the ITP method is seen in the ability to choose a finer grid spacing (a larger value of sampling points N) than in the prior 'mixing' procedure, as the ITP method doesn't require the diagonalization of an $N \times N$ matrix to evaluate the kinetic energy operator every iteration. We observe the sensitivity of the calculated chemical potential to the number of points sampled when considering an appropriate sampling mesh; it can be seen in the inset of Fig. 2.9 that this quantity is converged to ± 0.001 K for $N \geq 3200$ when simulating free helium slabs after 5000 iterations. Furthermore, qualitative analysis of the helium surface regions of the density profiles seen in Fig. 2.9 shows agreement to within 1×10^{-4} Å^{-3} for all values of N tested. The imaginary time step parameter τ is system-dependent, and therefore must be chosen by a trial-and-error process; an appropriate range for free helium slabs is found to be $\Delta\tau \in [5000, 8000]$ a.u.

Subsequent simulations implementing the He-graphite system described above are performed with a number of gridpoints N=4000 and an imaginary time step $\Delta\tau$=1000 a.u. (again determined by trial-and-error). The converged density profile obtained after 5000 iterations can be seen in Fig. 2.10, overlaid with the 'mixing' method result for clarity. It can be seen in the inset table of Fig. 2.10 that the increased resolution of the ITP method yields a helium-graphite distance in very good agreement with the experimentally observed value; however, it predicts a helium coverage that is practically unchanged from the 'mixing' method for both the first and second peaks. This indicates that the 'mixing' method is surprisingly robust, as the predicted density profiles are in agreement with the results of the ITP algorithm despite incurring the

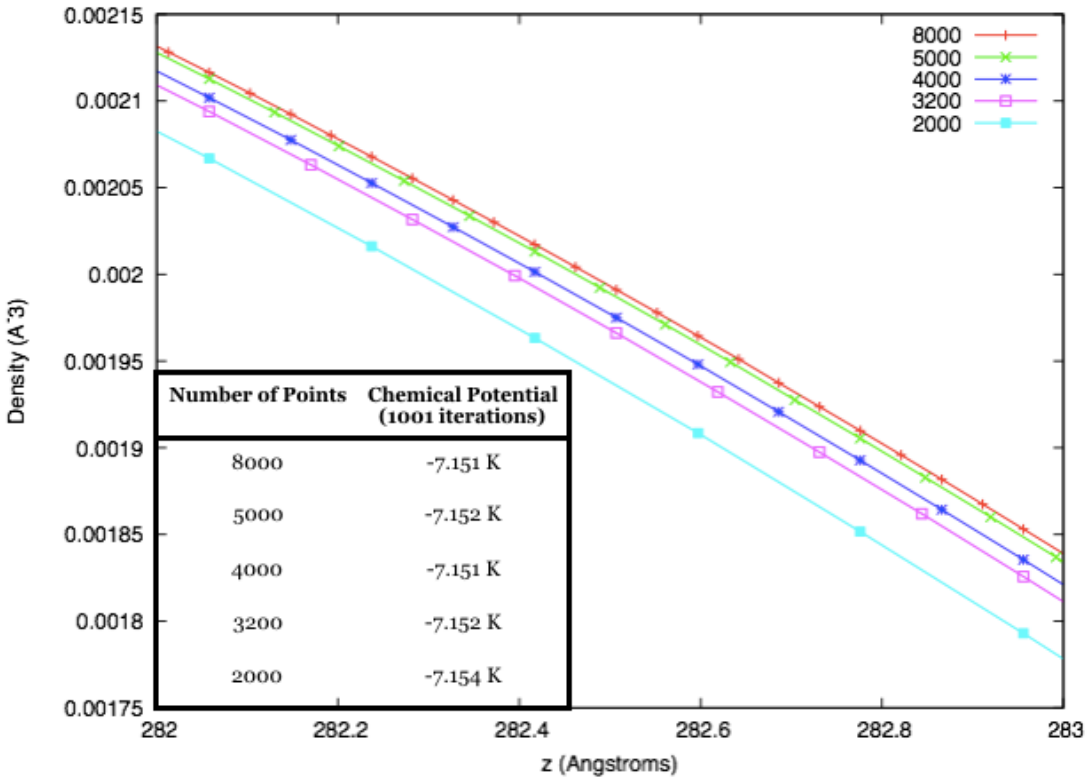


Figure 2.9: Surface regions of multiple density profiles of free planar slabs of He-4 calculated using the Orsay-Paris functional with the ITP method. Inset: Calculated chemical potential values corresponding to the various sampling meshes tested.

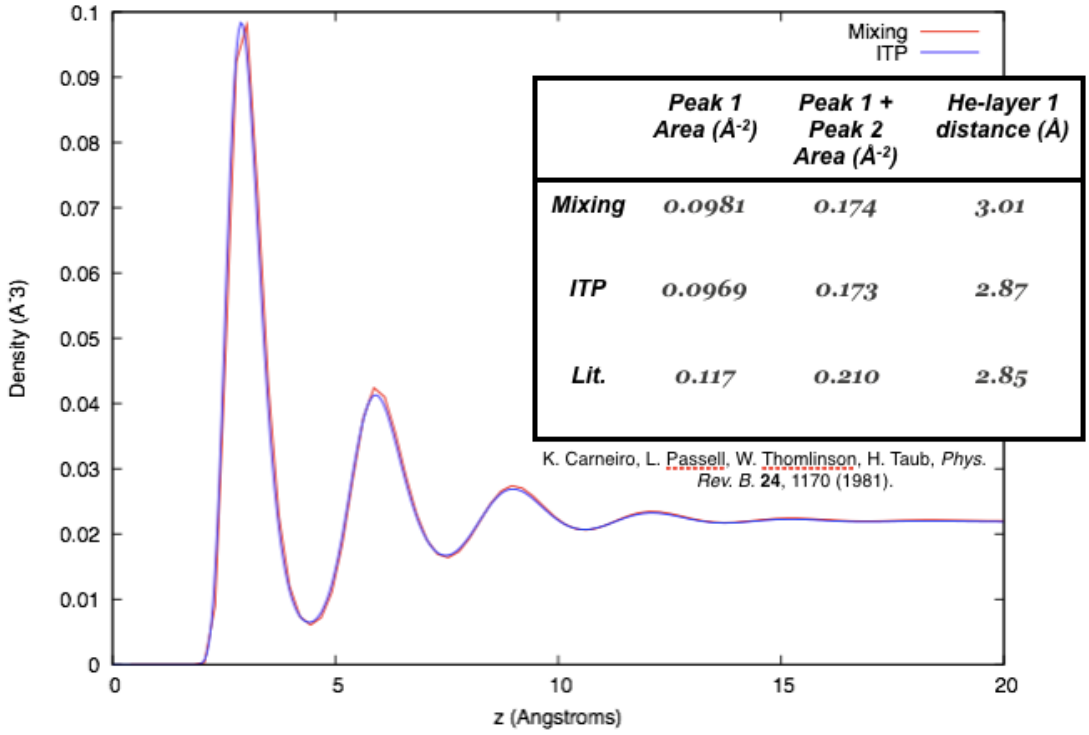


Figure 2.10: Solvation layer regions of He-4 density profiles adsorbed to a planar graphite surface, calculated using the Orsay-Paris functional. The graphite surface is located at z=0. Inset: Computed values for the first and first+second solvation layer densities, as well as for the He-graphite distance using both the 'mixing' and ITP profiles. The literature values represent neutron studies performed by Carneiro et al.

'sloshing' effects described above; however, we opt to use the ITP method for the remainder of our computations, as a high degree of self-consistency in the helium profiles is necessary to determine convergence between iterations. We also note that the chemical potential can be computed to arbitrary precision in the ITP method, something that is prohibited by the sloshing seen in the 'mixing' calculations.

2.4 Framework for Spherical Systems

Because the applicability of one-dimensional finite-sized helium slabs to real-world systems is hindered by the limited number of perfectly smooth surfaces, we also look to expand our model to be able to incorporate spherically symmetric helium droplets. Since the systems under study have been infinitely flat ($\vec{x}$, $\vec{y}$) helium slabs with finite thickness ($\vec{z}$), they have been represented conveniently using only one coordinate along the z-direction. Droplet systems, however, cannot be described in such a manner due to their finite size. One possible way to create a droplet of finite size computationally is to construct a box with side lengths appropriate to that of the desired droplet size, then place the droplet in the center of the box so that the helium density decays to zero before reaching the box faces.

The spatial discretization of the droplet must then be performed along each of the $\vec{x}$, $\vec{y}$, and $\vec{z}$ directions. In this work, the droplets are assumed to be spherically symmetric, and are thus constructed by uniformly sampling each side with N points. It should be noted, however, that each direction can in principle be sampled with differing frequencies, depending on the symmetry of the system under study and the gradient of the helium density expected along each direction. Such a discretization scheme would require thorough testing of the response of the helium density to changes in the sampling frequencies, for reasons that are discussed in more detail later. The starting helium density profile $\rho(\vec{\mathbf{r}})$ represented in 3-D Cartesian coordinates is then:

$$\rho_{init}(x,y,z) = \frac{\rho_0}{[1+exp(\sqrt{(x-i)^2+(y-j)^2+(z-k)^2}/a)]^\nu} \tag{2.22}$$

The additional parameters i, j, and k represent the x, y, and z coordinates for the center of the box, while the ρ_0, a, and ν parameters represent the same quantities as in Eq. 2.7. It is a trivial task to convert the Stringari-Treiner functional to this coordinate system due to its zero-range nature; the form of the Orsay-Paris functional in this coordinate scheme is mathematically equivalent to Eq. 2.2, and has the explicit form shown below:

$$\begin{aligned} U_{He-He}(x,y,z) = \frac{1}{2}\iiint \rho(x',y',z')V_{OP}(\sqrt{(x-x')^2+(y-y')^2+(z-z')^2}) \\ dx'dy'dz' + \frac{c}{2}[\bar{\rho}(x',y',z')]^{\gamma+1} + \frac{c}{2}(1+\gamma)\iiint \rho(x,y,z)\bar{\rho}(x',y',z')^\gamma \\ \times \Xi(\sqrt{(x-x')^2+(y-y')^2+(z-z')^2})dx'dy'dz' \end{aligned} \tag{2.23}$$

The form of the V_{OP} term is again dependent upon the distance between points (x,y,z) and (x',y',z') (Eq. 2.3), a property that becomes problematic in the three-dimensional coordinate scheme. To better understand this problem, it is first necessary to realize that the numerical evaluation of an integral in this space requires three applications of the trapezoid rule - one along each orthogonal axis. This creates small cube-like regions of integration with volume $\Delta x \Delta y \Delta z$, whose integral depends on the values at each corner of the region. The problem incurred with the calculation of the V_{OP} component of Eq. 2.23 is then how to treat the integration regions at the boundary of the cutoff distance h_{OP}, where the form of the integral changes within a single integration region. A depiction of this problem is seen in Fig. 2.11 for the 2D case, where the third dimension is not shown for clarity.

The most exact method of integrating the boundary region is by employing a Monte Carlo integration scheme for each of the offending boxes; however, we find that this approach renders computation of even small helium droplets unfeasible, as

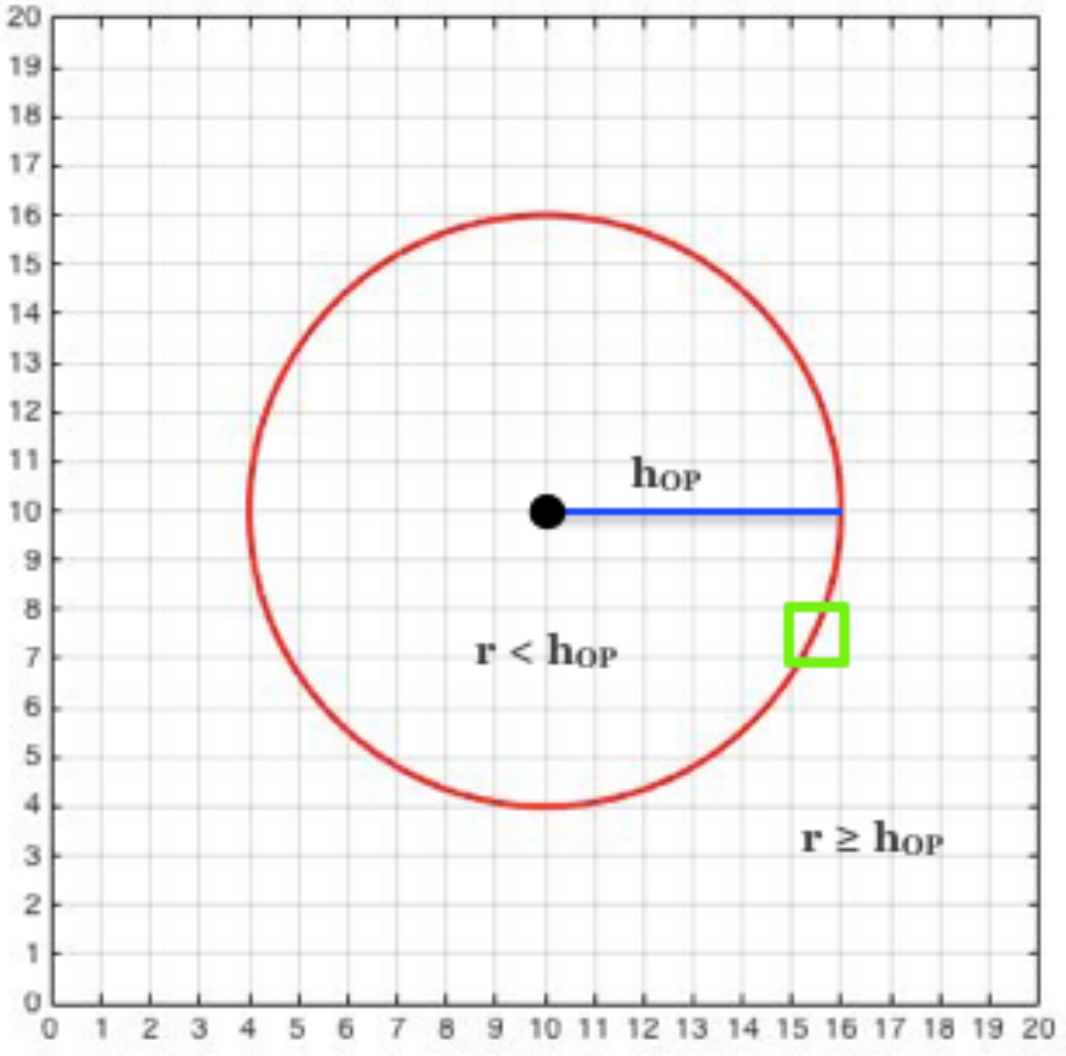


Figure 2.11: Two-dimensional depiction of the gridding problem incurred by attempting to describe the radial distance cutoff h_{OP} using Cartesian coordinates. The point of origin is at (10,10), and the length of the radius h_{OP} is not to scale. The green box highlights a problematic zone where the form of the interaction potential changes within a single integrable region.

there are many such instances in a single iteration, each requiring a costly Monte Carlo calculation. We instead opt to evaluate these problematic regions by applying the appropriate integrand to each corner, and summing the values of the corners to get the full integral for the region. Referencing Fig. 2.11 , applying this procedure to the region within the highlighted green square results in the top-left and bottom-left corners being assigned the function corresponding to $r < h_{OP}$, while the top- and bottom-right corners receive the function for which $r \geq h_{OP}$. The resultant two-dimensional integral is then the summation of these four values normalized by the factor $\frac{\Delta x \Delta y}{4}$.

The expressions for $\bar{\rho}(x, y, z)$ and one of the "c" terms in Eq. 2.23 also contain integrals whose form is dependent upon the distance from the point (x,y,z); we compute these in the same fashion as described above for V_{OP}, with the substitution of zeroes whenever an integration region's corner falls outside the distance cutoff h_{OP}.

A second problem in this three-dimensional representation scheme is the computational cost of the simulations. Consider a sampling mesh of $N = 2000$ points - corresponding to the smallest system of Fig. 2.9 or slightly larger than the 'mixing' method graphite simulation - which in three dimensions scales as $N^3 = 8 \times 10^9$, or eight billion points. Use of the Orsay-Paris functional, which requires integration over all points (x',y',z') for every point (x,y,z), requires a staggering $N^6 = 6.4 \times 10^{19}$ evaluations of the appropriate potential components for just the first term of Eq. 2.23. A subset of these points must also be evaluated (with the same N^6 scaling) when calculating $\bar{\rho}(x, y, z)$ and the second "c" term of the same equation, thus completing the mean field calculation portion of a single iteration.

We employ a number of computational tricks in the mean field subroutine to rein in the cost of these calculations, the first of which is an exploitation of the droplet symmetry. By realizing that the region of a spherically symmetric droplet in any particular octant of space (defined by the $\vec{x}$, $\vec{y}$, and $\vec{z}$ axes) is identical to the regions in the other seven octants, we can restrict our calculations to a single octant and reflect the results across the appropriate planes to generate the values for the

remaining octants. This reduces the number of points to be evaluated to an eighth of the full system $\left[\left(\frac{N}{2}\right)\left(\frac{N}{2}\right)\left(\frac{N}{2}\right)\right]$, although it should be noted that at this juncture the integrals over (x',y',z') must still be calculated using the entire N^3 set of points.

The number of operations per iteration can be further reduced by truncating the long-range tail of the helium-helium interaction, as the contribution to the total potential becomes vanishingly small in the limit of large separation. Inspection of the He-He potential energy curve leads us to select a cutoff of $r_{cut} = 20$ bohr, which is both far beyond the attractive region of the interaction and well within the size of the simulation box. This approximation is expected to have a negligible effect on the overall accuracy of the mean field potential, but allows for the triple integrals of Eq. 2.23 to be calculated with a subset of the full N^3 points corresponding to the cutoff parameter r_{cut}.

Finally, the speed of the simulations can be greatly increased by distributing the work to be done across multiple processors, a computational technique known as parallelization. In our code, the mean field calculations comprise more than 90% of the simulation time due to the aforementioned scaling issues; we thus parallelize this segment by assigning one processor per plane of points in the calculated octant, resulting in a theoretical speedup of $N/(2 * ceiling(\frac{N}{2n_{proc}}))$, where $N/2$ is the number of points along one edge of an octant, and n_{proc} is the number of processors used. As an example, using the earlier mentioned $N = 2000$ points on a side, the expected maximum speedup across 64 processors is $\frac{1000}{ceiling(1000/128)} = 125$x. The real speedup is necessarily less than this, as there is an overhead cost associated with processor initialization and communication.

Having tackled the modifications to the density initialization and mean field calculations, the last component to discuss is the 3-D imaginary time propagation routine. Because this component of the self-consistent algorithm is much less demanding than the mean field portion, it is done sufficiently quickly on a single processor. The only change from the ITP routine of the 1-D planar systems is that of the FFT used to convert between position and momentum spaces - in the 3-D

coordinate system, an FFT (iFFT) must be performed along each axis to fully convert into momentum (position) space. The FFT and iFFT algorithms used are exactly the same as in Section 2.2 [63]; they are simply applied three times sequentially in each iteration, thus increasing the computational duration of this section of code by an approximate factor of three.

Despite the improvements discussed above, the computational requirements of fully 3-D systems still exceeds the available resources for all but the smallest droplet sizes. Of the feasible simulations, we find that the calculated chemical potential values for small, undoped droplets are in agreement with the available literature values only to within ≈ 0.5K (Table 2.1). Even worse, we see that there is no clear convergence with an increase in sampling points for the currently attainable mesh densities, as seen in Figs. 2.12 and 2.13. Retrospective analysis of these values shows better agreement with our most recent calculations for the same systems - described in detail in Chapter 3 - although they still oscillate about the true value. We attribute this behavior to the discrete integration problem described above, and displayed in Fig. 2.11. As the number of sampling points increases, the potential (and therefore the density) should converge to the ground state; however, because the system sizes that are tractable to us are still very small, this convergence is not yet particularly good. In other words, the integration regions describing the surface of the droplet are still too large to accurately capture the energetics of the surface curvature, particularly since the droplets capable of being studied in this manner are very small. The improved agreement with our later chemical potential calculations is encouraging, but the overall inability to correctly describe regions of rapidly varying helium density - a problem that will only be exacerbated with the introduction of dopants into the droplets - motivates us to seek a new coordinate scheme to model these helium systems, as detailed in the next chapter.

Table 2.1: Chemical potentials calculated for N = 40 and N = 112 atom droplets using the 3-D coordinate system. The numerical subscripts on the chemical potentials indicate the number of sampling points used in each direction for that particular calculation. The last two columns are chemical potential values calculated by Barranco and Hernandez [1] and by us using a 1-D sampling scheme, respectively. Voided entries indicate simulations that did not complete in the available computing time, and so have no converged chemical potential data.

N (atoms)	μ_{64} (K)	μ_{78} (K)	μ_{94} (K)	μ_{110} (K)	μ_{124} (K)	μ_{140} (K)	μ_{BH} [1] (K)	μ_{Cat} (K)
40	-3.686	-3.565	-3.468	-3.639	-3.591	-3.555	-3.54	-3.548
112	-4.810	-4.631	-4.471	-4.728	—	—	-4.40	-4.614

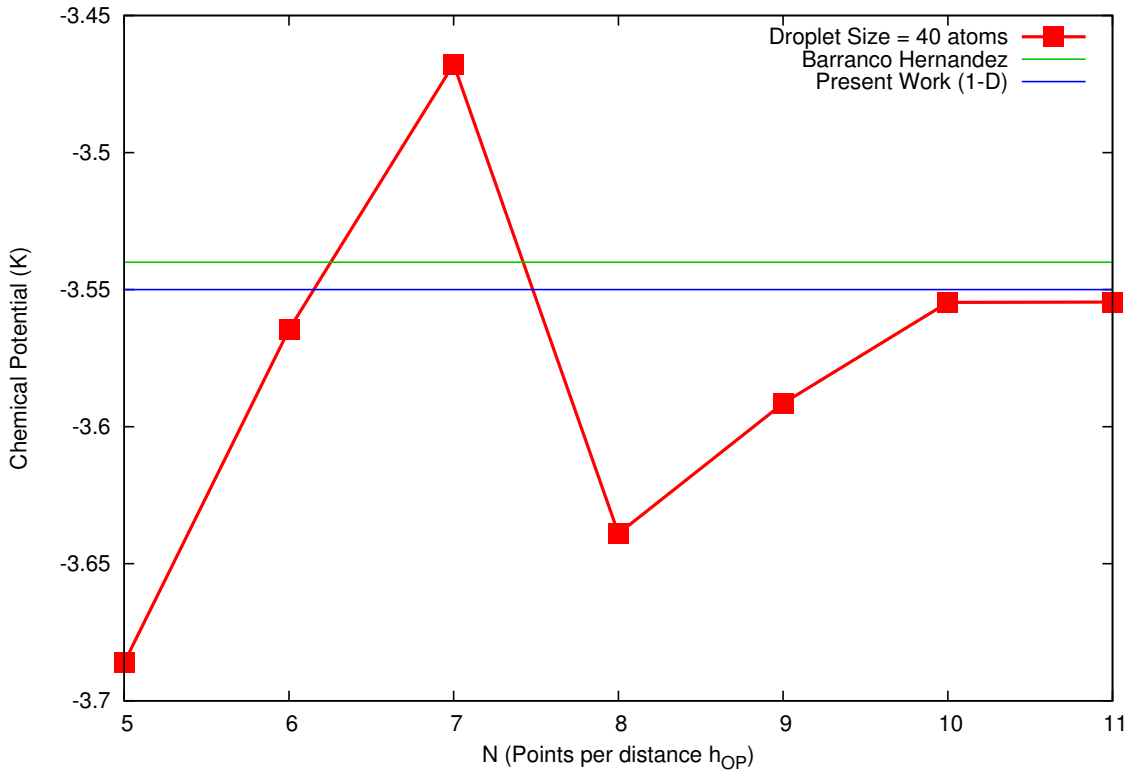


Figure 2.12: Calculated chemical potentials as a function of finer sampling mesh sizes for a 40-atom droplet using the Catalonia density functional. The horizontal green and blue lines represent the chemical potential for a 40-atom droplet calculated by Barranco and Hernandez[1] and by us using a 1-D droplet scheme, respectively.

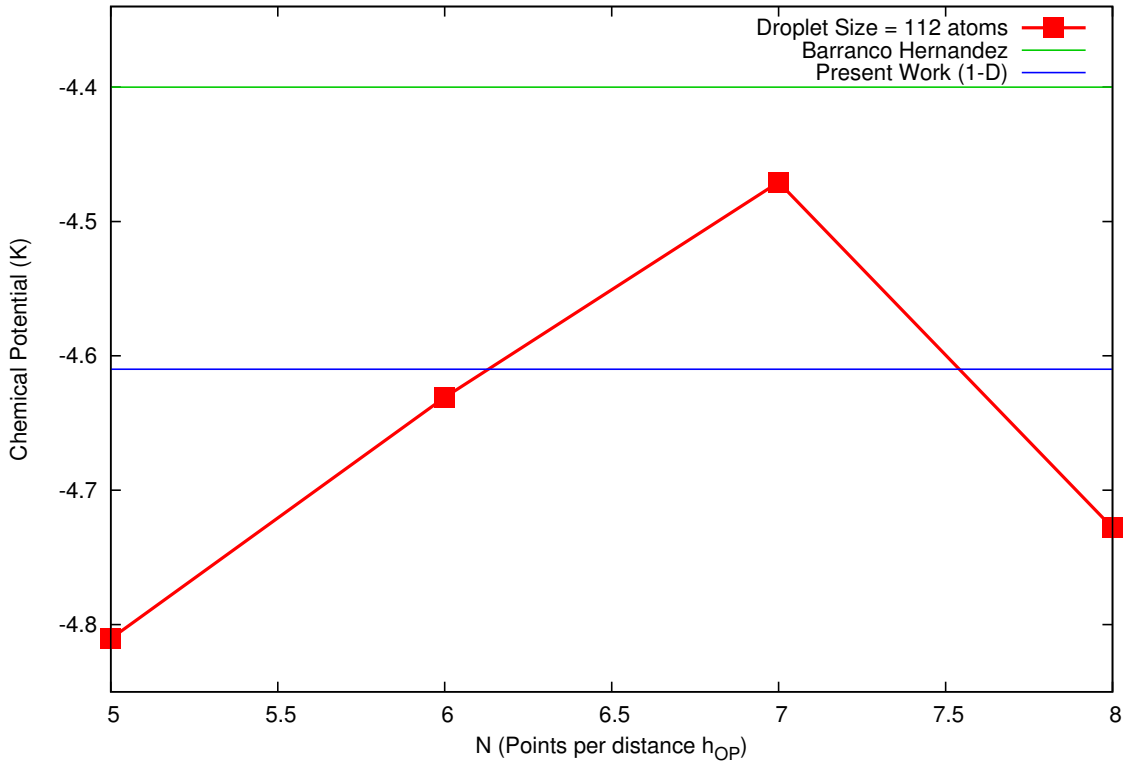


Figure 2.13: Calculated chemical potentials as a function of finer sampling mesh sizes for a 112-atom droplet using the Catalonia density functional. The horizontal green and blue lines represent the chemical potential for a 112-atom droplet calculated by Barranco and Hernandez[1] and by us using a 1-D droplet scheme, respectively.

Chapter 3

Doped He-4 Droplets

Recent decades have seen an increased interest in nanoscale superfluid helium droplets, particularly as cold, weakly-perturbing containers for spectroscopic studies of atoms, molecules and clusters and for understanding finite-size effects on the properties of helium as a quantum fluid ([33, 44, 45]). There has thus been a concurrent increase in theoretical work done in describing these systems. As described in Chapter 1, the atomistic prescription for modelling He droplets is a Monte Carlo evaluation of the droplet's ground state wavefunction; however, such a treatment quickly becomes intractable with an increasing number of particles. This is particularly problematic when considering droplets with many thousands of atoms, which are typical of experimental studies. An alternative continuum approach utilizes helium density functional theory (He-DFT) to calculate the total energy of the system by self-consistently evaluating the ground state one-body helium density according to the theorems of Hohenberg and Kohn [57]. The density functionals used in this method are intrinsically phenomenological in nature, with parameters that originate from fitting to the properties of bulk helium.

This type of theoretical modelling has been used in tandem with helium depletion experiments to predict the energetics of pick-up events, in which atomic and molecular dopants are captured by superfluid helium droplets. Such experiments observe the

distributions of helium droplet sizes before and after collisions with the desired dopant and, by considering the chemical potential of bulk helium ($\mu \approx$ -7.15 K/atom), can predict the energy deposited in the droplet by the captured species. However, He-DFT calculations presented both in this work and in the literature indicate that the bulk chemical potential is a poor approximation to the true chemical potential of even very large helium droplets, due to surface curvature effects.

The objectives of this chapter are twofold: we first look to evaluate rigorously the behavior of the Orsay-Paris [2], Catalonia [1], and Orsay-Trento [3] density functionals in the context of helium droplets with noble gas and transition metal dopants by examining the kinetic and potential energy components of each system (Sections 3.5 and 3.6). We then use the chemical potential values computed from He-DFT to improve the accuracy of energetic predictions derived from pick-up experiments (Section 3.7). A summary of our conclusions can be found in Section 3.8.

3.1 Orsay-Paris, Catalonia, and Orsay-Trento Functionals

The systems modeled in this chapter are spherically symmetric representations of droplets, meaning that they are most conveniently represented using spherical coordinates r, θ, and ϕ. The generic forms of the three non-local functionals we use to model these systems can be found in Eqs. 3.1 and 3.2 (Orsay-Paris functional), Eqs. 3.1 and 3.3 (Catalonia functional), or 3.6 and 3.7 (Orsay-Trento functional), depending on the functional of choice (note that the Orsay-Paris functional is the same as in Chapter 2). The bold-face vectors $\boldsymbol{r}$ and $\boldsymbol{r'}$ in each case correspond to the set of coordinates (r, θ, ϕ) and (r', θ', ϕ') in the system, respectively, although $\boldsymbol{r}$ can be chosen in such a way that θ and ϕ are both equal to zero. We enact this choice for $\boldsymbol{r}$, which changes our notation for the coordinates corresponding to $\boldsymbol{r'}$ to be (r', θ, ϕ) - since the only angular terms in the system now belong to $\boldsymbol{r'}$, we drop their 'prime'

designations for convenience. The behavior of these functionals can essentially be described as a summation of long-range van der Waals attractions and short-range repulsive interactions. To be more exact, the repulsive terms describe a screened-core interaction with decreasing distance that takes effect within a certain cutoff radius h_{cut}. The value of this cutoff is governed by parameters given in Table 3.1.

$$E_{OP,\,Cat} = \frac{1}{2}\int \rho(\boldsymbol{r})\rho(\boldsymbol{r}')V_{OP,\,Cat}(|\boldsymbol{r}-\boldsymbol{r}'|)d\boldsymbol{r}' + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} \tag{3.1}$$

$$V_{OP}(|\boldsymbol{r}-\boldsymbol{r}'|) = \begin{cases} 4\epsilon\left[(\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|})^{12} - (\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|})^{6}\right] & |\boldsymbol{r}-\boldsymbol{r}'| \geq h_{cut} \\ V_{OP}(h_{cut})\left(\frac{|\boldsymbol{r}-\boldsymbol{r}'|}{h_{cut}}\right)^{4} & |\boldsymbol{r}-\boldsymbol{r}'| < h_{cut} \end{cases} \tag{3.2}$$

$$V_{Cat}(|\boldsymbol{r}-\boldsymbol{r}'|) = \begin{cases} 4\epsilon\left[(\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|})^{12} - (\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|})^{6}\right] & |\boldsymbol{r}-\boldsymbol{r}'| \geq \sigma \\ b\left[1 - (\frac{|\boldsymbol{r}-\boldsymbol{r}'|}{\sigma})^{8}\right] & |\boldsymbol{r}-\boldsymbol{r}'| < \sigma \end{cases} \tag{3.3}$$

The constants σ and ϵ represent the Lennard-Jones He-He interaction [60], while h_{cut} is the cutoff distance for screening effects. The term $\bar{\rho}(\boldsymbol{r})$ is a coarse-grained density with the form seen in Eq. 3.4 for both functionals [2] (note that this is the same coarse-graining function as seen in Chapter 2).

$$\bar{\rho}(\boldsymbol{r}) = \int \rho(\boldsymbol{r})\Xi(|\boldsymbol{r}-\boldsymbol{r}'|)d\boldsymbol{r} \tag{3.4}$$

The term $\Xi(|\boldsymbol{r}-\boldsymbol{r}'|)$ above is a piecewise function that is non-zero only when the distance $|\boldsymbol{r}-\boldsymbol{r}'|$ is less than or equal to that of the cutoff distance h_{cut}, and represents normalization over the spherical volume for which the coarse-graining is defined.

$$\Xi(|\boldsymbol{r}-\boldsymbol{r}'|) = \begin{cases} \frac{3}{4\pi h_{cut}^{3}} & |\boldsymbol{r}-\boldsymbol{r}'| \leq h_{cut} \\ 0 & |\boldsymbol{r}-\boldsymbol{r}'| > h_{cut} \end{cases} \tag{3.5}$$

Table 3.1: Functional parameters for the functionals given in [2], [1], and [3]. Voided entries indicate that a particular parameter is not applicable to the given functional.

Functional	b (K)	c ($\times 10^7$ KÅ$^{3(\gamma+1)}$)	c′ ($\times 10^4$ KÅ^6)	c″ ($\times 10^6$ KÅ^9)	γ	h_{cut} (Å)	α (Å^3)	ρ_0 (Å^{-3})
Orsay-Paris ([2])	-888.810	1.04554	—	—	2.8	2.376728	—	—
Catalonia ([1])	-888.810	1.04554	—	—	2.8	2.376728	—	—
Orsay-Trento ([3])	-718.990	—	-2.411857	1.858496	—	2.190323	54.31	0.04

The Orsay-Trento non-local functional is again similar to the other two, but truncates the screened-core potential at a different value of the cutoff distance h_{cut} and splits the coarse-grained density term into two components (Eqs. 3.6, 3.7):

$$E_{OT} = \frac{1}{2}\int \rho(\boldsymbol{r})\rho(\boldsymbol{r}')V_{OT}(|\boldsymbol{r}-\boldsymbol{r}'|)d\boldsymbol{r}' + \frac{c'}{2}[\bar{\rho}(\boldsymbol{r})]^2 + \frac{c''}{3}[\bar{\rho}(\boldsymbol{r})]^3 + e_{gg}[\nabla\rho(\boldsymbol{r})] \quad (3.6)$$

$$V_{OT}(|\boldsymbol{r}-\boldsymbol{r}'|) = \begin{cases} 4\epsilon\left[(\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|})^{12} - (\frac{\sigma}{|\boldsymbol{r}-\boldsymbol{r}'|})^{6}\right] & |\boldsymbol{r}-\boldsymbol{r}'| \geq h_{cut} \\ 0 & |\boldsymbol{r}-\boldsymbol{r}'| < h_{cut} \end{cases} \quad (3.7)$$

Finally, in the Orsay-Trento functional, an additional term $e_{gg}[\nabla\rho(\boldsymbol{r})]$ is added as a non-local correction to the kinetic energy, which is dependent on the density gradient (Eq. 3.8).

$$e_{gg}[\nabla\rho(\boldsymbol{r})] = -\frac{\hbar^2}{4m}\alpha\int \rho(\boldsymbol{r}')\left[\frac{1}{\pi^{3/2}l^3}e^{-|\boldsymbol{r}-\boldsymbol{r}'|^2/l^2}\right] \times \left[1-\frac{\tilde{\rho}(\boldsymbol{r})}{\rho_0}\right]\left(\frac{\nabla\rho(\boldsymbol{r})}{\rho(\boldsymbol{r})}\right)\cdot\left(\frac{\nabla\rho(\boldsymbol{r}')}{\rho(\boldsymbol{r}')}\right)\left[1-\frac{\tilde{\rho}(\boldsymbol{r}')}{\rho_0}\right]d\boldsymbol{r}' \quad (3.8)$$

The use of soft brackets above indicates the terms which are to be used in the dot product, while the use of hard brackets and the multiplication symbol are understood to mean scalar multiplication. The constants α and ρ_0 are phenomenological parameters derived in [3], while $l = 1$ Å is a constant to account for correct units in the integral.

The new function $\tilde{\rho}(\boldsymbol{r})$ is a gaussian-weighted version of the density $\rho(\boldsymbol{r})$, and has the form seen in Eq. 3.9.

$$\tilde{\rho}(\boldsymbol{r}) = \int \rho(\boldsymbol{r}')\frac{1}{\pi^{3/2}l^3}exp(-|\boldsymbol{r}-\boldsymbol{r}'|^2/l^2)d\boldsymbol{r}' \quad (3.9)$$

Note that the use of these energy functionals in Eq. 1.5 requires them to first be differentiated with respect to the helium density function $\rho(\boldsymbol{r})$, resulting in an expression corresponding to the U_{He-He} term. This is shown in Eqs. 3.10 and 3.11, where the subscript indicates the functional being operated on.

$$U_{OP,\,Cat} = \frac{1}{2}\int \rho(\boldsymbol{r}')V_{OP,\,Cat}(|\boldsymbol{r}-\boldsymbol{r}'|)d\boldsymbol{r}' + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} + \frac{c(1+\gamma)}{2}\int \Xi(|\boldsymbol{r}-\boldsymbol{r}'|)\rho(\boldsymbol{r}')[\bar{\rho}(\boldsymbol{r})]^{\gamma}d\boldsymbol{r} \quad (3.10)$$

$$U_{OT} = \frac{1}{2}\int \rho(\boldsymbol{r}')V_{OT}(|\boldsymbol{r}-\boldsymbol{r}'|)d\boldsymbol{r}' + \frac{c'}{2}[\bar{\rho}(\boldsymbol{r})]^2 + c'\int \Xi(|\boldsymbol{r}-\boldsymbol{r}'|)\rho(\boldsymbol{r}')[\bar{\rho}(\boldsymbol{r})]d\boldsymbol{r} + \frac{c''}{3}[\bar{\rho}(\boldsymbol{r})]^3 + (c'')^2\int \Xi(|\boldsymbol{r}-\boldsymbol{r}'|)\rho(\boldsymbol{r}')[\bar{\rho}(\boldsymbol{r})]^2 d\boldsymbol{r} + e_{gg}[\nabla\rho(\boldsymbol{r})] \quad (3.11)$$

As was seen at the end of Chapter 2 in the case of Cartesian coordinates, a full three-dimensional evaluation of the droplet systems is intractable for droplets containing hundreds of helium atoms or more. To reduce the computational expense of these calculations, we can analytically integrate over the angular terms θ and ϕ implicitly present in Eqs. 3.10 and 3.11, resulting in functionals that are dependent upon only the radial coordinate (note that a similar procedure was done at the beginning of Chapter 2 in the context of 1-D planar slabs). However, the presence of both the short- and long- range interaction terms in the functionals complicates this process, and leads to a number of distinct cases to consider.

The first case, as seen in Fig. 3.1, is that of $r = 0$, and results in the division of the droplet into two regions separated by the boundary h_{cut}, corresponding to the two cases of Eqs. 2.3, 3.3, or 3.7, depending on the choice of functional. This also denotes the radial integration limits - $0 \leq r' < h_{cut}$ for region one, and $h_{cut} \leq r < \infty$ for region two. The limits of integration on the angular coordinates in this case are

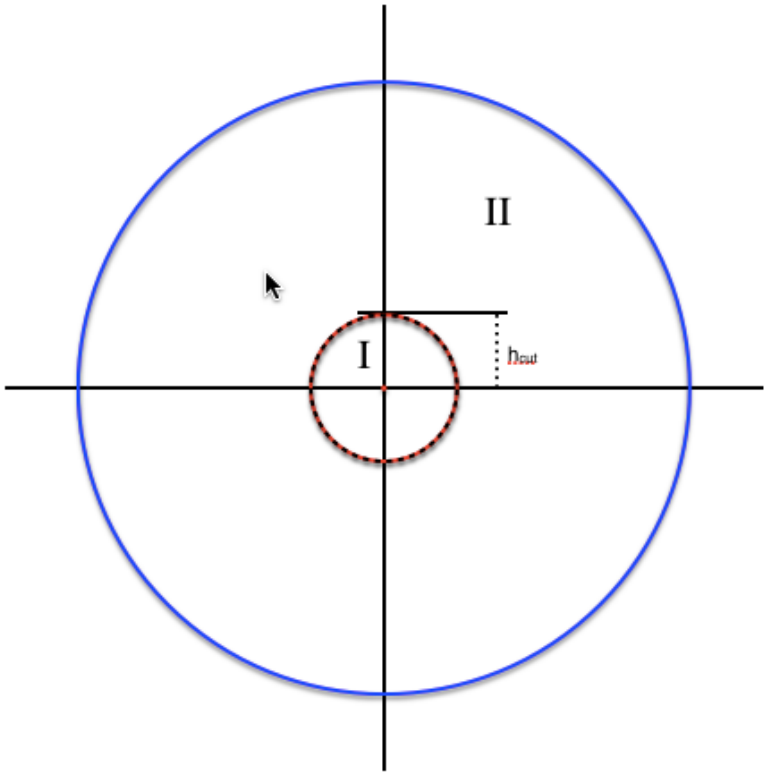


Figure 3.1: Cross-section of a helium droplet (blue circle) showing the distinct integration regions of Eqs. 3.12 - 3.14. The short-range interactions are contained within the red circle, while the dashed circles show the radial integral boundaries. In this instance, the only boundary aligns with the short-range cutoff distance, h_{cut}. Note that, in the Catalonia model, the parameter h_{cut} should be replaced with the Lennard-Jones sigma for He-He interactions.

$0 < \phi < 2\pi$ and $0 < \theta \leq \pi$ for both regions, resulting in Eqs. 3.12, 3.13, and 3.14 for the form of the 1-D integral of Eqs. 3.10 and 3.11 (the corresponding functional is noted as a subscript).

$$U_{OP,\,1}(r) = \frac{16\pi\epsilon}{h_{cut}^4}\Big[\Big(\frac{\sigma}{h_{cut}}\Big)^{12} - \Big(\frac{\sigma}{h_{cut}}\Big)^{6}\Big]\int_0^{h_{cut}} \rho(r')(r')^6 dr' + 16\pi\epsilon\int_{hcut}^{\infty}\rho(r')\Big[\Big(\frac{\sigma^{12}}{(r')^{10}}\Big) - \Big(\frac{\sigma^6}{(r')^4}\Big)\Big]dr' + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} + \frac{3c}{2h_{cut}^3}(1+\gamma)\int_0^{h_{cut}}(r')^2\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}dr' \quad (3.12)$$

$$U_{Cat,\,1}(r) = 4\pi b_{LJ}\int_0^{\sigma}\rho(r')\Big[1 - \Big(\frac{r'}{\sigma}\Big)^{8}\Big](r')^2 dr' + 16\pi\epsilon\int_{\sigma}^{\infty}\rho(r')\Big[\Big(\frac{\sigma^{12}}{(r')^{10}}\Big) - \Big(\frac{\sigma^6}{(r')^4}\Big)\Big]dr' + \frac{c}{2}[\bar{\rho}(r)]^{\gamma+1} + \frac{3c}{2h_{cut}^3}(1+\gamma)\int_0^{h_{cut}}(r')^2\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}dr' \quad (3.13)$$

$$U_{OT,\,1}(r) = 16\pi\epsilon\int_{h_{cut}}^{\infty}\rho(r')\Big[\Big(\frac{\sigma^{12}}{(r')^{10}}\Big) - \Big(\frac{\sigma^6}{(r')^4}\Big)\Big]dr' + \frac{c'}{2}[\bar{\rho}(r')]^2 + \frac{c''}{3}[\bar{\rho}(r')]^3 + \frac{3c'}{h_{cut}^3}\int_0^{h_{cut}}(r')^2\rho(r')dr'\bar{\rho}(r')dr' + \frac{3c''}{h_{cut}^3}\int_0^{h_{cut}}(r')^2\rho(r')dr'[\bar{\rho}(r')]^2 dr' \quad (3.14)$$

The second case, depicted in Fig. 3.2, occurs when $r < h_{cut}$, and divides the droplet into four different regions. The first and fourth regions are described by the short-range and long-range interactions, respectively, and have limits of integration $0 < \phi < 2\pi$ and $0 < \theta \leq \pi$ for the angular coordinates. The radial limits of integration in these cases are $0 \leq r' < h_{cut} - r$ and $r + h_{cut} \leq r' < \infty$, respectively. The second region is also described by short-range interactions, but has the limits of integration $h_{cut} - r \leq r' < r + h_{cut}$ for the radial component, and $0 < \phi < 2\pi$

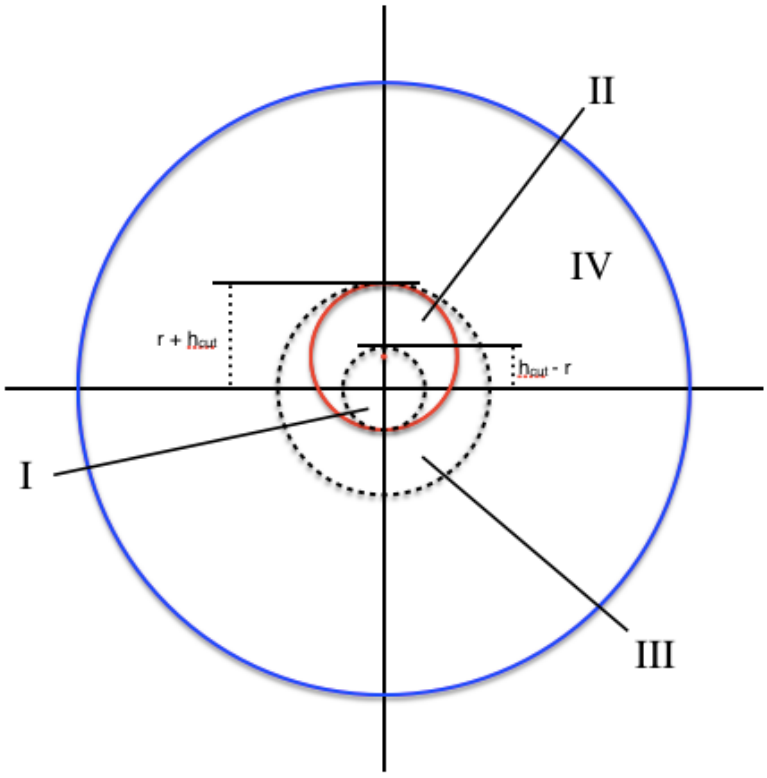


Figure 3.2: Cross-section of a helium droplet (blue circle) showing the distinct integration regions of Eqs. 3.15 - 3.17. The short-range interactions are contained within the red circle, while the dashed circles show the radial integral boundaries. The angular portion is omitted for clarity. Note that, in the Catalonia model, the parameter h_{cut} should be replaced with the Lennard-Jones sigma for He-He interactions.

and $1 \leq cos(\theta) \leq \frac{r^2+(r')^2-h_{cut}^2}{2rr'}$ for the angular components. The third region is the other long-range interaction region with radial limits of integration equal to those of region two, and the angular limits $0 < \phi < 2\pi$ and $\frac{r^2+(r')^2-h_{cut}^2}{2rr'} \leq cos(\theta) \leq -1$. The resultant 1-D integral corresponding to Eqs. 3.10 and 3.11 in this case is seen in Eqs. 3.15, 3.16, and 3.17.

$$
\begin{aligned}
U_{OP,\ 2}(r) = \frac{4\pi\epsilon}{3h_{cut}^4}\Big[\Big(\frac{\sigma}{h_{cut}}\Big)^{12} - \Big(\frac{\sigma}{h_{cut}}\Big)^{6}\Big]\Big(\int_0^{h_{cut}-r} \frac{r'\rho(r')}{r}[(r^2+(r')^2+2rr')^3 \\
- (r^2+(r')^2-2rr')^3]dr' + \int_{h_{cut}-r}^{r+h_{cut}} \frac{r'\rho(r')}{r}[h_{cut}^6 - (r^2+(r')^2-2rr')^3]dr'\Big) \\
+ 4\pi\epsilon\Big(\int_{h_{cut}-r}^{r+h_{cut}} \frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{h_{cut}^{10}} - \frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big) \\
\times \Big(\frac{1}{h_{cut}^4} - \frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr' + \int_{r+h_{cut}}^{\infty} \frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^5} \\
- \frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr'\Big) \\
+ \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} + \frac{3c}{2h_{cut}^3}(1+\gamma)\int_0^{h_{cut}-r} (r')^2\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}dr' + \frac{3c}{4h_{cut}^3}(1+\gamma) \\
\times \int_{h_{cut}-r}^{r+h_{cut}} (r')^2\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}\Big(1 - \frac{r^2+(r')^2-h_{cut}^2}{2rr'}\Big)dr' \quad (3.15)
\end{aligned}
$$

$$U_{Cat,\ 2}(r) = 2\pi b_{LJ}\Big(2\int_0^{\sigma-r}\rho(r')(r')^2dr' - \frac{1}{10\sigma^8}\int_0^{\sigma-r}\frac{r'\rho(r')}{r}[(r^2+(r')^2+2rr')^5 - (r^2+(r')^2-2rr')^5]dr' + \int_{\sigma-r}^{r+\sigma}\rho(r')(r')^2\Big[1-\frac{r^2+(r')^2-\sigma^2}{2rr'}\Big]dr' - \frac{1}{10\sigma^8}\int_{\sigma-r}^{r+\sigma}\frac{r'\rho(r')}{r}[\sigma^10 - (r^2+(r')^2-2rr')^5]dr'\Big) + 4\pi\epsilon\Big(\int_{\sigma-r}^{r+\sigma}\frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\times\Big(\frac{1}{\sigma^{10}}-\frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{\sigma^4}-\frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr' + \int_{r+\sigma}^{\infty}\frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^5}-\frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^2}-\frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr'\Big) + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} + \frac{3c}{2h_{cut}^3}(1+\gamma)\int_0^{h_{cut}-r}(r')^2\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}dr' + \frac{3c}{4h_{cut}^3}(1+\gamma)\int_{h_{cut}-r}^{r+h_{cut}}(r')^2\times\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}\Big(1-\frac{r^2+(r')^2-h_{cut}^2}{2rr'}\Big)dr' \quad (3.16)$$

$$U_{OT,\ 2}(r) = 4\pi\epsilon\Big(\int_{h_{cut}-r}^{r+h_{cut}}\frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{h_{cut}^{10}}-\frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big)\times\Big(\frac{1}{h_{cut}^4}-\frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr' + \int_{r+h_{cut}}^{\infty}\frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^2}-\frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr'\Big) + \frac{c'}{2}[\bar{\rho}(r')]^2 + \frac{c''}{3}[\bar{\rho}(r')]^3 + \frac{3c'}{h_{cut}^3}\int_0^{h_{cut}-r}(r')^2\rho(r')dr'\bar{\rho}(r')dr' + \frac{3c''}{h_{cut}^3}\int_0^{h_{cut}-r}(r')^2\times\rho(r')dr'[\bar{\rho}(r')]^2dr' + \frac{3c'}{2h_{cut}^3}\int_{h_{cut}-r}^{r+h_{cut}}(r')^2\rho(r')dr'\bar{\rho}(r')\Big(1-\frac{r^2+(r')^2-h_{cut}^2}{2rr'}\Big)dr' + \frac{3c''}{2h_{cut}^3}\int_{h_{cut}-r}^{r+h_{cut}}(r')^2\rho(r')dr'[\bar{\rho}(r')]^2\Big(1-\frac{r^2+(r')^2-h_{cut}^2}{2rr'}\Big)dr' \quad (3.17)$$

The third and final case can be seen in Fig. 3.3, and occurs when $r > h_{cut}$. This scenario again divides the droplet into four distinct regions that must be accounted for separately in the new integral. The first and fourth cases are both representative

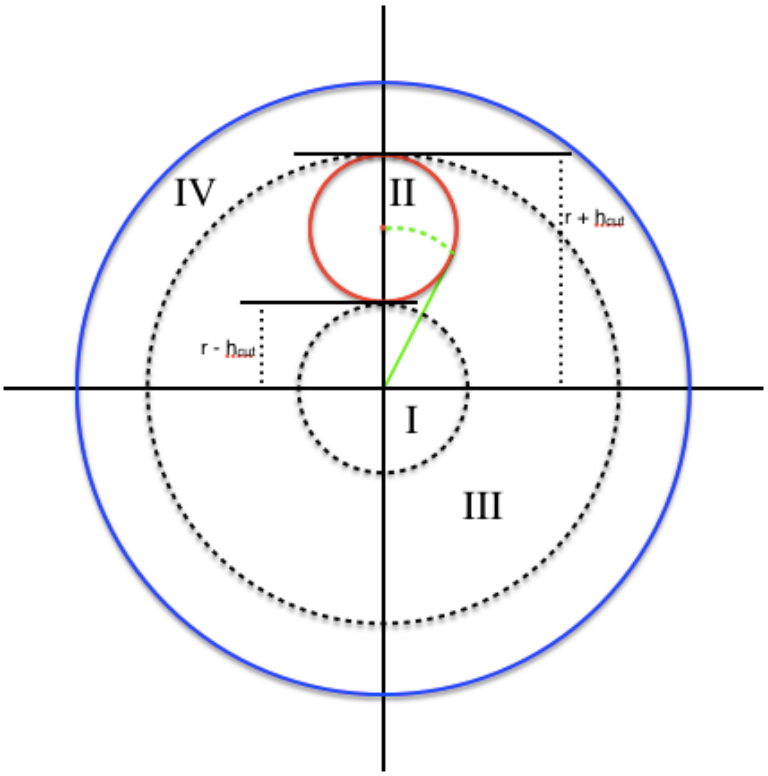


Figure 3.3: Cross-section of a helium droplet (blue circle) showing the distinct integration regions of Eqs. 3.18 - 3.20. The short-range interactions are contained within the red circle, while the dashed circles show the radial integral boundaries. The angular portion is represented by a green arc. Note that, in the Catalonia model, the parameter h_{cut} should be replaced with the Lennard-Jones sigma for He-He interactions.

of long-range interactions, and have limits of integration $0 < \phi < 2\pi$ and $0 < \theta \leq \pi$ for the angular terms. The radial integration limits are $0 \leq r' \leq r - h_{cut}$ for the first region, and $r + h_{cut} \leq r' < \infty$ for the fourth. The third region is also a region of long-range interactions, but has the limits of integration $r - h_{cut} < r' < r + h_{cut}$, $\frac{r^2+(r')^2-h_{cut}^2}{2rr'} \leq cos(\theta) \leq -1$, and $0 < \phi < 2\pi$. The second region is the only short-range interaction region in this case, with limits of integration $r - h_{cut} < r' < r + h_{cut}$, $1 \leq cos(\theta) \leq \frac{r^2+(r')^2-h_{cut}^2}{2rr'}$, and $0 < \phi < 2\pi$. The 1-D integral of Eqs. 3.10 and 3.11 in this final case is seen in Eqs. 3.18, 3.19, and 3.20.

$$
\begin{aligned}
U_{OP,\,3}(r) = 4\pi\epsilon\Bigg(& \int_0^{r-h_{cut}} \frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5}\Big) \\
& - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr' + \int_{r-h_{cut}}^{r+h_{cut}} \frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big) \\
& \times \Big(\frac{1}{h_{cut}^{10}} - \frac{1}{(r^2+(r')^2+2rr')^5}\Big) - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{h_{cut}^4} - \frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr' \\
& + \int_{r+h_{cut}}^{\infty} \frac{r'\rho(r')}{r}\Big[\Big(\frac{\sigma^{12}}{5}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5}\Big) \\
& - \Big(\frac{\sigma^6}{2}\Big)\Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2}\Big)\Big]dr'\Bigg) + \frac{4\pi\epsilon}{3h_{cut}^4}\Big[\Big(\frac{\sigma}{h_{cut}}\Big)^{12} \\
& - \Big(\frac{\sigma}{h_{cut}}\Big)^6\Big]\int_{r-h_{cut}}^{r+h_{cut}} \frac{r'\rho(r')}{r}[h_{cut}^6 - (r^2+(r')^2-2rr')^3]dr' + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} + \frac{3c}{4h_{cut}^3} \\
& \times (1+\gamma)\int_{r-h_{cut}}^{r+h_{cut}} (r')^2\rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma}\Big(1 - \frac{r^2+(r')^2-h_{cut}^2}{2rr'}\Big)dr' \quad (3.18)
\end{aligned}
$$

$$U_{Cat,\ 3}(r) = 4\pi\epsilon\Big(\int_0^{r-\sigma} \frac{r'\rho(r')}{r} \Big[\Big(\frac{\sigma^{12}}{5}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5} \Big)$$
$$- \Big(\frac{\sigma^6}{2}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2} \Big) \Big] dr' + \int_{r-\sigma}^{r+\sigma} \frac{r'\rho(r')}{r} \Big[\Big(\frac{\sigma^{12}}{5}\Big)$$
$$\times \Big(\frac{1}{\sigma^{10}} - \frac{1}{(r^2+(r')^2+2rr')^5} \Big) - \Big(\frac{\sigma^6}{2}\Big) \Big(\frac{1}{\sigma^4} - \frac{1}{(r^2+(r')^2+2rr')^2} \Big) \Big] dr'$$
$$+ \int_{r+\sigma}^{\infty} \frac{r'\rho(r')}{r} \Big[\Big(\frac{\sigma^{12}}{5}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5} \Big)$$
$$- \Big(\frac{\sigma^6}{2}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2} \Big) \Big] dr' \Big)$$
$$+ 2\pi b_{LJ} \Big(\int_{\sigma-r}^{r+\sigma} \rho(r')(r')^2 \Big[1 - \frac{r^2+(r')^2-\sigma^2}{2rr'} \Big] dr' - \frac{1}{10\sigma^8} \int_{\sigma-r}^{r+\sigma} \frac{r'\rho(r')}{r}$$
$$\times [\sigma^{10} - (r^2+(r')^2-2rr')^5] dr' \Big) + \frac{c}{2}[\bar{\rho}(\boldsymbol{r})]^{\gamma+1} + \frac{3c}{4h_{cut}^3}(1+\gamma) \int_{r-h_{cut}}^{r+h_{cut}} (r')^2$$
$$\times \rho(r')[\bar{\rho}(\boldsymbol{r})]^{\gamma} \Big(1 - \frac{r^2+(r')^2-h_{cut}^2}{2rr'} \Big) dr' \quad (3.19)$$

$$U_{OT,\ 3}(r) = 4\pi\epsilon\Big(\int_0^{r-h_{cut}} \frac{r'\rho(r')}{r} \Big[\Big(\frac{\sigma^{12}}{5}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5} \Big)$$
$$- \Big(\frac{\sigma^6}{2}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2} \Big) \Big] dr' + \int_{r-h_{cut}}^{r+h_{cut}} \frac{r'\rho(r')}{r} \Big[\Big(\frac{\sigma^{12}}{5}\Big)$$
$$\times \Big(\frac{1}{h_{cut}^{10}} - \frac{1}{(r^2+(r')^2+2rr')^5} \Big) - \Big(\frac{\sigma^6}{2}\Big) \Big(\frac{1}{h_{cut}^4} - \frac{1}{(r^2+(r')^2+2rr')^2} \Big) \Big] dr'$$
$$+ \int_{r+h_{cut}}^{\infty} \frac{r'\rho(r')}{r} \Big[\Big(\frac{\sigma^{12}}{5}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^5} - \frac{1}{(r^2+(r')^2+2rr')^5} \Big)$$
$$- \Big(\frac{\sigma^6}{2}\Big) \Big(\frac{1}{(r^2+(r')^2-2rr')^2} - \frac{1}{(r^2+(r')^2+2rr')^2} \Big) \Big] dr' \Big) + \frac{c'}{2}[\bar{\rho}(r')]^2 + \frac{c''}{3}[\bar{\rho}(r')]^3$$
$$+ \frac{3c'}{2h_{cut}^3} \int_{r-h_{cut}}^{r+h_{cut}} (r')^2 \rho(r') dr' \bar{\rho}(r') \Big(1 - \frac{r^2+(r')^2-h_{cut}^2}{2rr'} \Big) dr' + \frac{3c''}{2h_{cut}^3}$$
$$\times \int_{r-h_{cut}}^{r+h_{cut}} (r')^2 \rho(r') dr' [\bar{\rho}(r')]^2 \Big(1 - \frac{r^2+(r')^2-h_{cut}^2}{2rr'} \Big) dr' \quad (3.20)$$

Note that, in each of these cases, the averaging sphere represented by the Ξ function (Eq. 2.5) has also been integrated over the angular coordinates; these limits of integration are the same as those given for the short-range interactions of each particular case for the Orsay-Paris and Orsay-Trento functionals, and differ only in the cutoff parameter for the Catalonia functional. The constant b_{LJ} in the Catalonia equations (Eqs. 3.13, 3.16, and 3.19) is a combination of other constants, seen in Eq. 3.21.

$$b_{LJ} = \frac{33}{8}\left(\frac{8}{9}\epsilon + \frac{b}{4\pi\sigma^3}\right) \tag{3.21}$$

The constant b on the right-hand side of Eq. 3.21 is the same as in Table 3.1 for the Catalonia functional. The $e_{gg}[\nabla\rho(\boldsymbol{r})]$ term in the Orsay-Trento functional can also be integrated over the angular terms, and assumes the 1-D form of Eq. 3.22 (note that this component is case-independent):

$$\begin{aligned} e_{gg}[\nabla\rho(r)] = \frac{\hbar^2\alpha}{m\sqrt{\pi}}\left(1-\frac{\rho(r)}{\rho_0}\right)\int \nabla\rho(r')\left[1-\frac{\rho(r')}{\rho_0}\right] \\ \times r'\left[\left(1+\frac{1+2rr'}{2r^2}\right)e^{-(r^2+(r')^2+2rr')} - \left(1+\frac{1-2rr'}{2r^2}\right)e^{-(r^2+(r')^2-2rr')}\right]dr' \end{aligned} \tag{3.22}$$

3.2 Self-Consistent Approach

As was seen in Chapter 2 for planar systems, the ground state helium density profile is computed by iteratively solving (Eq. 1.5), where U_{He-He} is a mean-field potential energy term defined by the choice of helium-helium density functional (see Section 3.1). We adopt an imaginary-time propagation (ITP) formalism based on a second-order split operator method, as described in detail in [62]. Briefly, this method approximates an exponentiated Hamiltonian for the system according to Eq. 3.23 below:

$$e^{-(T+V)\Delta\tau/\hbar} = e^{-\frac{1}{2}V\Delta\tau/\hbar} \times e^{-T\Delta\tau/\hbar} \times e^{-\frac{1}{2}V\Delta\tau/\hbar} \tag{3.23}$$

In this instance, the potential V refers to the sum of U_{He-He} and $V_{external}$ (Eq. 1.5). By choosing an appropriate imaginary-time step, $\Delta\tau$, successive applications of Eq. 3.23 to a trial wavefunction converge to the true ground state. After testing a range of values for this parameter by checking for convergence of both the chemical potential and density profile, we find an appropriate imaginary-time step for our systems to be $\Delta\tau = 2500$ a.u. (Figs. 3.4 and 3.5, and Table 3.2).

An advantage to the ITP method employed here is the decoupling of the kinetic and potential energy terms in the exponentiated operator. This allows each to be evaluated efficiently in a convenient vector space, namely, coordinate space for the potential term and momentum space for the kinetic term. We choose this approach because, despite the extra effort needed for the conversion between vector spaces, it is overall more efficient than evaluating the kinetic energy term in Cartesian form. This is because, in Cartesian coordinates, the calculation of the kinetic energy term requires the use of a finite difference approximation for the Laplacian, which necessitates multiple function (density) values for a single computation. Conversely, in momentum coordinates, the kinetic energy calculation is simply a multiplication by the term $p^2/2m$. This conversion to momentum space is achieved via the Hankel transform, which can be thought of as an analogue of the Fourier transform for systems with spherical symmetry. To better understand the role of this operation in our self-consistent algorithm, we begin by firstpresenting the radial component of the Schrödinger equation in spherical polar coordinates operating on the helium wavefunction $\sqrt{\rho(r)}$ (Eq. 3.24), as this represents central the self-consistent equation in this coordinate scheme.

$$\frac{-\hbar^2}{2m}\left(\frac{d}{dr^2} + \frac{2}{r}\frac{d}{dr}\right)\sqrt{\rho(r)} + V(r)\sqrt{\rho(r)} = E\sqrt{\rho(r)} \tag{3.24}$$

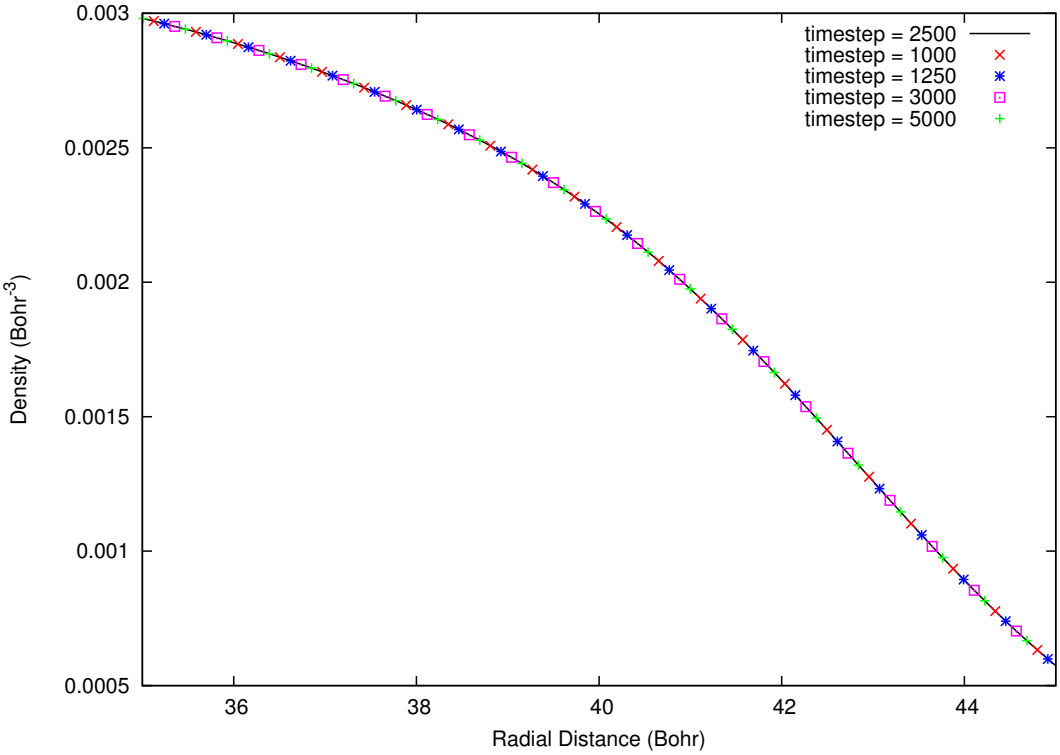


Figure 3.4: A portion of the surface region of resultant density profiles for an N=1000 atom droplet after 5000 iterations with varying imaginary time step parameters using the Orsay-Paris functional. Note that the values for $\Delta\tau$ = 1000-5000 a.u. produce density profiles that are all aligned with one another.

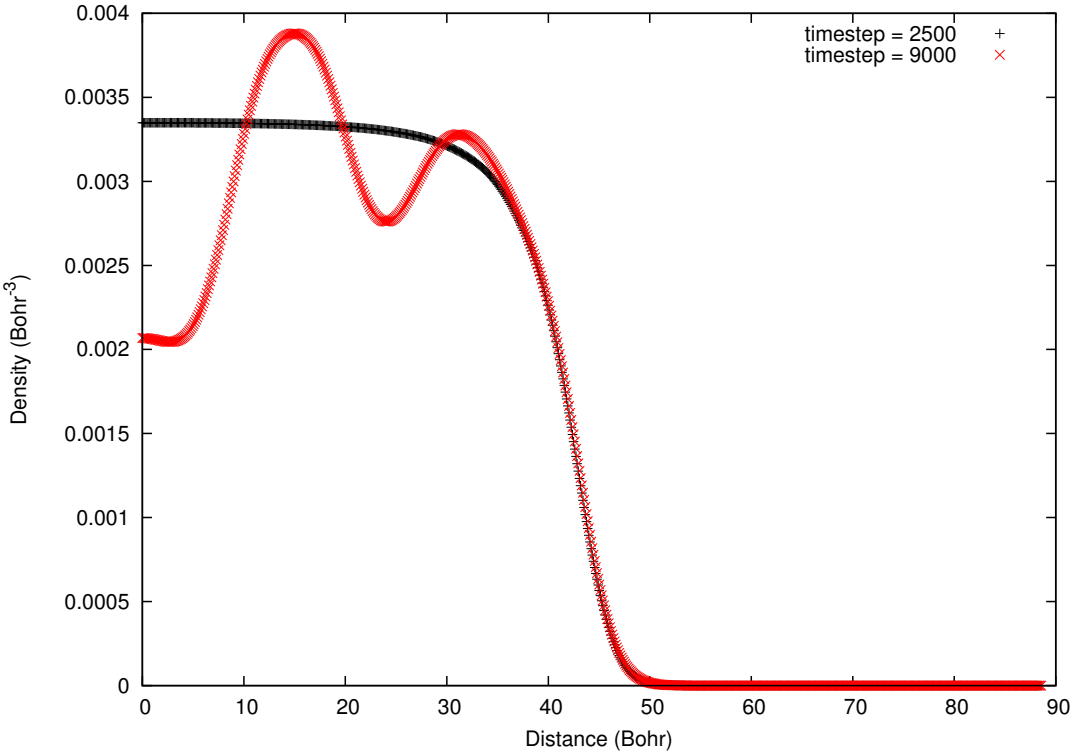


Figure 3.5: Resultant density profiles for an N=1000 atom droplet calculated with the Orsay-Paris functional after 5000 iterations with two time step parameters, $\Delta\tau$=2500 a.u. and $\Delta\tau$=9000 a.u. Note that the latter of these values produces an unstable and unconverged density profile.

Table 3.2: Energetic quantities predicted for an N=1000 atom droplet with various time step parameters. The green entry indicates the final time step chosen for our simulations, while the red entry represents an example of a particularly 'bad' time step based on the lack of convergence seen in the E/N and μ values.

$\Delta\tau$ (a.u.)	E/N (K/atom)	μ (K)
1000	-5.3969481984	-5.98068
1250	-5.3969481984	-5.98069
2500	-5.3969481977	-5.98071
3000	-5.3969481973	-5.98071
5000	-5.3969481926	-5.98072
9000	-5.3772763147	-5.79718

By making the substitution $f(r) = r\sqrt{\rho(r)}$ and multiplying by r on both sides, Eq. 3.24 is transformed into Eq. 3.25.

$$\frac{-\hbar^2}{2m}\left(\frac{d^2 f(r)}{dr^2}\right) + V(r)f(r) = Ef(r) \tag{3.25}$$

Note that the form of the kinetic energy in this expression is convenient for use in the Hankel transform into reciprocal space; we thus use $f(r) = r\sqrt{\rho(r)}$ in place of the helium wavefunction, and divide through by r at the end of our calculations before recovering the helium density. The Hankel transform itself can be considered to be the Fourier transform in spherical coordinates, and is expressed for both the forward (Eq. 3.26) and reverse (Eq. 3.27) cases using the implementation of Bisseling and Kosloff [65]:

$$G(k) = \sqrt{\frac{2}{\pi}} \int_0^\infty r J_\nu(kr) F(r) dr \tag{3.26}$$

$$F(r) = \sqrt{\frac{2}{\pi}} \int_0^\infty k J_\nu(kr) G(k) dk \tag{3.27}$$

The generic functions $F(r)$ and $G(k)$ are a Fourier pair, and correspond to the coordinate-space and reciprocal-space representations of the system wavefunction, respectively, while the function $J_\nu(kr)$ is a Bessel function of order ν. Because we consider exclusively spherically symmetric systems, the Bessel function used in this work is $J_{\nu=0.5}(kr) = sin(kr)/\sqrt{kr}$; this choice is again guided by the work of Bisseling and Kosloff, who use this function on calculations of the 3-D harmonic oscillator. This function can then be related to the spherical Bessel function j_0 by the relation seen in Eq. 3.28.

$$j_0(kr) = \sqrt{\frac{\pi}{2kr}} J_{0.5}(kr) \tag{3.28}$$

The spherical Bessel function derived from $J_{\nu=0.5}(kr)$ is then $j_0(kr) = sin(kr)/kr$. Incorporating both the spherical Bessel function and the one-dimensional function

$f(r) = r\sqrt{\rho(r)}$ into the spherical Bessel transform given by Serov [66] results in Eq. 3.29 to convert between coordinate and reciprocal space:

$$g(k) = \int_0^\infty kr \frac{sin(kr)}{kr} f(r) dr \tag{3.29}$$

The inverse operation is then displayed in Eq. 3.30.

$$f(r) = \int_0^\infty kr \frac{sin(kr)}{kr} g(k) dk \tag{3.30}$$

Note that, after cancelling the kr terms in the numerator and denominator, Eqs. 3.29 and 3.30 are in actuality simply sine transforms of the appropriate helium wavefunction. By combining the zeroth-order spherical Bessel transform with the imaginary-time propagation algorithm, a complete cycle in our iterative approach then consists of the calculation seen in Eq. 3.31.

$$g(k) = \sqrt{\frac{2}{\pi}} \int_0^\infty sin(kr) e^{\frac{-V(r)\Delta\tau}{2\hbar}} \sqrt{\rho_{old}(r)} dr \tag{3.31}$$

$$f_{new}(r) = N e^{\frac{-V(r)\Delta\tau}{2\hbar}} \sqrt{\frac{2}{\pi}} \int_0^\infty sin(kr) e^{\frac{-\gamma(k)^2\Delta\tau}{2m}} g(k) dk \tag{3.32}$$

The functions $V(r)$ and $g(k)$ are the coordinate-space mean-field potential and reciprocal-space wavefunction, respectively, while N is a normalization constant chosen to preserve the total number of atoms between iterations. It is implied that Eqs. 3.29 and 3.30 are used for the forward and reverse transforms necessary in Eq. 3.31.

To summarize, a trial ground-state density enters the ITP equation, where it is sequentially transformed, first by half of the potential energy term (dictated by the choice of helium functional and any external potential terms), then by the momentum-space representation of the kinetic energy term, and finally by the second half of the potential energy term. The resultant function is then normalized to maintain constant

particle number N, and the cycle begins again. This process is repeated for 5000 steps - an ample amount of time to achieve self-consistency, as evaluated by point-to-point differencing of the old and new helium wavefunctions. The convergence of the density profile is determined by the maximum value obtained from point-to-point differencing of the initial and final profiles of each cycle. A cartoon depicting this iterative process can be seen in Fig. 3.6.

3.3 Analysis of System Discretization

Due to the numerical nature of the calculations performed, the helium density profile and corresponding mean-field potential is discretized into a finite number of linearly spaced points. We enforce spherical symmetry upon the droplet systems, and so only sampling of the radial direction is necessary. Preliminary calculations on multiple undoped droplet systems were performed to observe the effects of sampling discretization on the helium density profiles. These test calculations use the Orsay-Paris functional and a system size of N=330 atoms, determined by trapezoid-rule integration of the density profile. Because of the limits of integration on the $\bar{\rho}(r)$ term in the functional, we consider sampling point meshes that guarantee an integer number of points on the interval $(r - h_{cut},\, r + h_{cut})$, including points located exactly on both the lower and upper bounds. This convention, denoted $n_{h_{cut}}$ (number of points per interval h_{cut}), prevents skewing of the coarse-grained helium density used in the repulsive terms of the functional.

As evidenced by Table 3.3, the per-atom energy and chemical potential vary by only 0.002 K for sampling densities as small as $n_{h_{cut}} = 30$. We choose a sampling density of $n_{h_{cut}} = 40$ for all calculations, as this falls within the range for which these quantities have converged to within 0.001 K, and makes for a convenient spacing between consecutive points when using the Catalonia functional. This second point is important to consider, as both h_{cut} and the Lennard-Jones σ_{He-He} are distance cutoff parameters in the Catalonia functional.

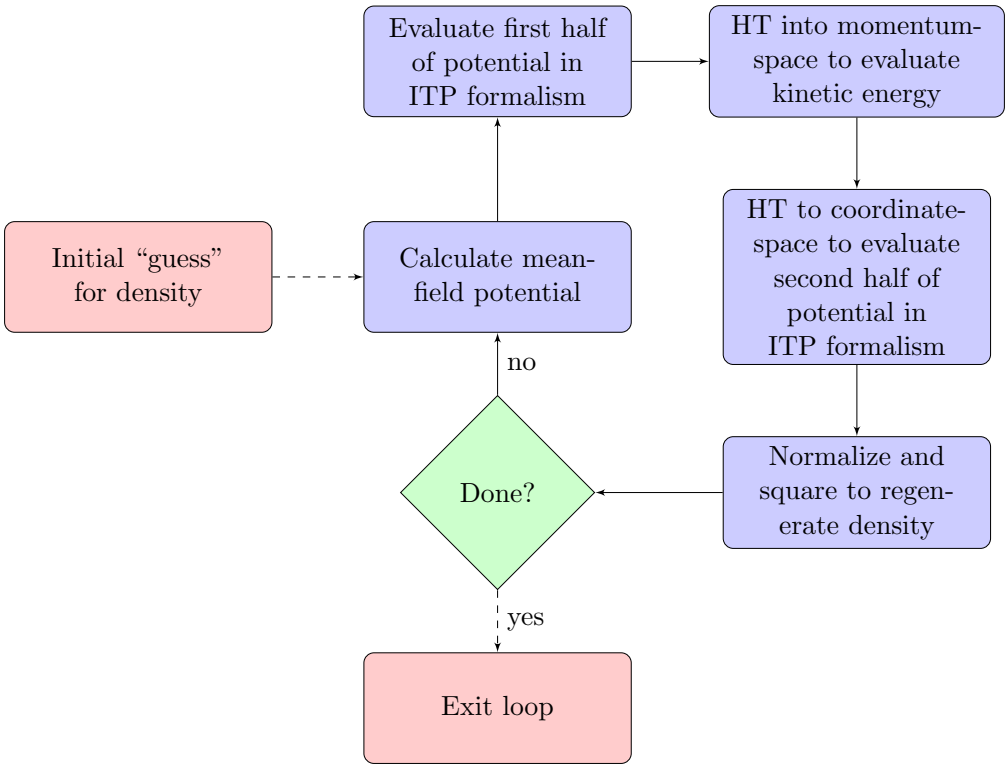


Figure 3.6: Diagram of the iterative procedure used to calculate self-consistent helium density profiles. ITP = Imaginary Time Propagation, HT = Hankel Transform

Table 3.3: Effect of system discretization on the energetic properties of pure droplets.

$n_{h_{cut}}$	dr (bohr)	E_{tot} (K)	E/N (K/atom)	μ (K)
10	0.499041	-1531.75	-4.64167	-5.46573
20	0.236388	-1527.01	-4.62731	-5.44708
30	0.154875	-1526.34	-4.62526	-5.44438
40	0.115163	-1526.12	-4.62460	-5.44348
50	0.091661	-1526.02	-4.62430	-5.44309
100	0.045367	-1525.89	-4.62392	-5.44256

Because the number of atoms associated with each droplet is enforced via numerical integration of the normalized density profiles (and is thus susceptible to roundoff error), it is also important to check the sensitivity of the chemical potential to small variations in the corresponding droplet size. We do this by enforcing density profiles that integrate to a non-integer number of atoms, either slightly below (N_{lower}) or slightly above (N_{upper}) an integer target value N, then approximating the chemical potential at N using the formula $\mu = (E_{upper} - E_{lower})/(N_{upper} - N_{lower})$. We repeat this test at many different droplet sizes. By comparing the chemical potential for a droplet with N atoms to the approximated chemical potentials calculated using the equation introduced above, we can verify that the energetic output is not affected by the precision to which the droplet represents an integer number of atoms. These results are shown in Table 3.4.

3.4 Pure Droplet Results

As in any DFT-related study, our first task is to verify the functionals used to describe the helium droplets. To this end, we investigate a set of clusters ranging in size from N=20 to N=728 helium atoms, according to previous studies in references [59] and [1]. Energetic data for these cluster sizes utilizing the Catalonia density functional can be found in Table 3.5, while data for the remaining two functionals relative to that of the Catalonia functional are seen in Table 3.6. We find that the per-particle energies of the present work are in excellent agreement with those calculated by Barranco for the same functional, and also in good agreement with those calculated by Stringari and Treiner for the larger droplet sizes. The chemical potential values calculated in this work are also in agreement with those of Barranco, although they suggest a slightly more bound droplet for the larger sizes tested. We convince ourselves that the present values predicted by the Catalonia functional are accurate for these larger droplets ($N \geq 112$) by noting that they are in good agreement with the Orsay-Paris

Table 3.4: Energetic data for droplets containing non-integer numbers of atoms. Note that the approximations (dE/dN) to the chemical potential at the m±1 and m±2 values (where the m indicates the ingeter-value droplet, and m±1, m±2 are that many 'steps' from the integer droplet) are in agreement with one another to 6+ decimal places, and in agreement to the chemical potential at N to within ≈0.01 K.

m		
N	Energy (K)	μ (K)
20.001	-27.70681846	-2.61288619
40.007	-90.87853675	-3.54765611
69.999	-207.9613661	-4.17169855
111.993	-393.5995354	-4.61414625
167.997	-662.2748677	-4.94319005
239.989	-1028.256620	-5.19707342
329.984	-1505.952520	-5.39878478
727.992	-3752.665341	-5.81352961

m±1				
N_{lower}	N_{upper}	E_{lower} (K)	E_{upper} (K)	dE/dN (K)
19.996	20.006	-27.69371715	-27.71992389	-2.62264676
39.998	40.015	-90.84982823	-90.90725123	-3.55449679
69.987	70.011	-207.9118734	-208.0108667	-4.17686464
111.976	112.009	-393.5241961	-393.6748843	-4.61836851
167.975	168.018	-662.1686740	-662.3810727	-4.94679193
239.962	240.017	-1028.114606	-1028.398646	-5.20019542
329.950	330.018	-1505.769731	-1506.135323	-5.40155205
727.935	728.050	-3752.330580	-3753.000121	-5.81558730

m±2				
N_{lower}	N_{upper}	E_{lower} (K)	E_{upper} (K)	dE/dN (K)
19.991	20.011	-27.68061996	-27.73303343	-2.62264680
39.990	40.023	-90.82112569	-90.93597169	-3.55449681
69.975	70.022	-207.8623884	-208.0603749	-4.17686465
111.960	112.025	-393.4488662	-393.7502427	-4.61836852
167.954	168.040	-662.0624915	-662.4872890	-4.94679194
239.935	240.044	-1027.972606	-1028.540685	-5.20019542
329.916	330.052	-1505.586957	-1506.318141	-5.40155205
727.877	728.108	-3751.995838	-3753.334922	-5.81558730

Table 3.5: Energetic data for select droplet sizes using the Catalonia density functional.

N	E/N (K/atom, Present Work)	E/N (K/atom, Ref. [1])	E/N (K/atom, Ref. [59])	μ (K, Present Work)	μ (K, Ref. [1])	μ (K, Ref. [59])
20	-1.385	-1.40	-1.27	-2.613	-2.62	-2.52
40	-2.272	-2.27	-2.18	-3.548	-3.54	-3.57
70	-2.971	-2.97	-2.93	-4.172	-4.16	-4.24
112	-3.515	-3.51	-3.51	-4.614	-4.40	-4.70
168	-3.942	-3.94	-3.96	-4.943	-4.74	-5.04
240	-4.285	-4.28	-4.31	-5.197	-5.00	-5.26
330	-4.564	-4.56	-4.60	-5.399	-5.20	-5.46
728	-5.155	-5.16	-5.19	-5.814	-5.61	-5.87

Table 3.6: Energetic data for select droplet sizes using the Orsay-Paris and Orsay-Trento density functionals. The entries are taken relative to the Catalonia values of Table 3.5, and color-coordinated to show stronger (blue) versus weaker (red) binding behavior relative to the Catalonia functional. The last column shows the ratio of densities of the centers of the droplets (ρ_c) to that of the bulk value (ρ_0).

	Orsay-Paris			Orsay-Trento		
N	E/N (K/atom)	μ (K)	ρ_c / ρ_0	E/N (K/atom)	μ (K)	ρ_c / ρ_0
20	-0.1084	-0.1270	0.8600	-0.0620	-0.0522	0.9210
40	-0.1092	-0.0980	0.9622	-0.0815	-0.0127	0.9727
70	-0.0998	-0.0791	1.0057	-0.0787	0.0047	0.9919
112	-0.0894	-0.0663	1.0249	-0.0711	0.0123	1.0538
168	-0.0800	-0.0571	1.0345	-0.0631	0.0159	1.0080
240	-0.0720	-0.0501	1.0382	-0.0558	0.0176	1.0675
330	-0.0653	-0.0447	1.0396	-0.0495	0.0183	1.0185
728	-0.0505	-0.0354	1.0373	-0.0354	0.0185	1.0492

and Orsay-Trento values of this work, as well as with the previously calculated values of Stringari and Treiner.

Upon verifying the density functionals' ability to reproduce the energetics of the small set of droplets, we evaluate a much larger set of droplet sizes ranging from $N = 50$ to $N = 9500$ helium atoms per droplet for all three functionals. This data is displayed graphically for convenience (Fig. 3.7). The calculated droplet energies can be fit to a generic formula (Eq. 3.33), allowing for extrapolation to large system sizes where calculation times are inconveniently long. The values of the coefficients a, b, c, and d for each functional are displayed in Table 3.7. Differentiating Eq. 3.33 with respect to N gives an expression for the size-dependent chemical potential $\mu(N)$ of the droplets, as seen in Eq. 3.34.

$$E(N) = aN + bN^{2/3} + cN^{1/3} + d \tag{3.33}$$

$$\mu(N) = a + \frac{2}{3}bN^{-1/3} + \frac{1}{3}cN^{-2/3} \tag{3.34}$$

The a term in the above fits represents the bulk per-particle energy, and in all cases agrees well with the known theoretical value (-7.15 K, [58]). The b term is related to the surface tension (σ_s) of the droplet through Eq. 3.35 [58], and yields surface tension values of 0.27953 K/Å^2, 0.28642 K/Å^2, and 0.28000 K/Å^2 for the Orsay-Paris, Catalonia, and Orsay-Trento functionals, respectively (note that the parameter ρ_0 in this equation is the value for bulk helium density, not the parameter listed in Table 3.1). These compare favorably to the experimental value of 0.256 K/Å^2 [67]. Furthermore, residual analysis of the fitting equation to the calculated energies yields a maximum difference of $\approx$ 1 K across the range of cluster sizes considered (N=50 to N=9500), further verifying the quality of the fits. To check that our fitting parameters still accurately describe the properties of droplets larger than N=9500, we calculate the energy and chemical potential of a 25,000 atom droplet using the

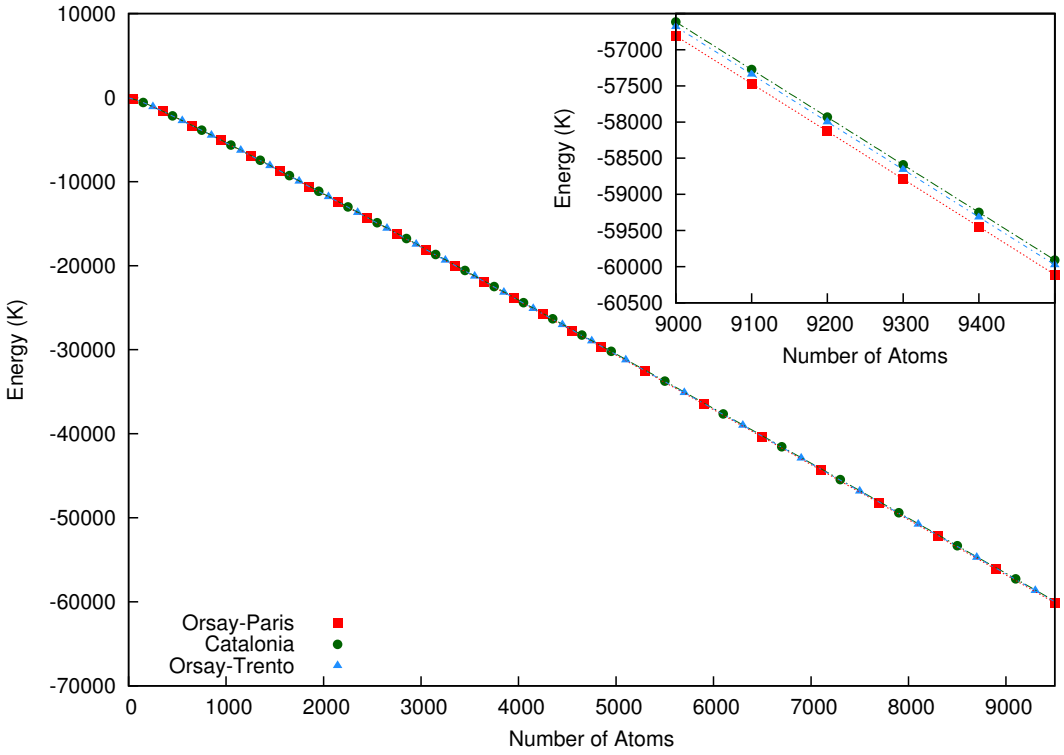


Figure 3.7: Energy as a function of droplet size for each functional. The colored lines correspond to a fit of Eq. 3.33 to the appropriate points. Some values are omitted for clarity.

Table 3.7: Fitting parameters for Eq. 3.33

Functional	a (K/atom)	b (K/atom$^{2/3}$)	c (K/atom$^{1/3}$)	d (K)
Orsay-Paris	-7.15398	17.30335	5.86748	-31.98443
Catalonia	-7.15389	17.72987	6.36351	-34.28385
Orsay-Trento	-7.14079	17.33265	5.65332	-31.05216

Orsay-Paris functional; these quantities are predicted by Eqs. 3.33 and 3.34 to within 4 K and 0.0005 K, respectively.

$$\sigma_s = \frac{b}{(4\pi)^{1/3}(3/\rho_0)^{2/3}} \tag{3.35}$$

3.5 Droplets with Noble Gas Dopants

Because the helium droplet systems are restricted to be spherically symmetric, we focus our investigation on dopants that share this symmetry (i.e. S-state atoms). The dopant-helium two-body interaction enters the functional as $V_{external}$, and in the case of noble gas atoms, has the form established in [68] for neon and argon, and in [69] for xenon (Eqs. 3.36 and 3.38, respectively). In the Ne/Ar potential, the repulsive region is characterized by an exponential decay, while the long-range attraction includes terms up to r^{16}:

$$V_{Ne,Ar-He}(r) = Ae^{-\alpha r+\beta r^2} - \sum_{n=3}^{8} f_{2n}(r,b)c_{2n}/r^{2n} \tag{3.36}$$

$$f_{2n}(r,b) = 1 - e^{-br} - \sum_{k=0}^{2n}(br)^k/k! \tag{3.37}$$

Meanwhile, the xenon potential is defined in a piecewise fashion, with the repulsive region and attractive well described separately from the long-range dispersion interactions:

$$V_{Xe-He}(r) = \begin{cases} \frac{C_6}{120}\left(\frac{b}{3}\right)^6\left(ae^{-br} - \chi e^{-2br/3} - e^{-br/3}\right) & br < 16.6 \\ \frac{C_6}{120}\left(\frac{b}{3}\right)^6\left(ae^{-br} - \frac{C_6}{r^6 - Q^2r^4}\right) & br \geq 16.6 \end{cases} \tag{3.38}$$

Note that the value for b in Eq. 3.38 is not necessarily the same as in Eqs. 3.36 and 3.37; the reader is directed to references [68] and [69] for the specific values of the parameters in these equations.

Mirroring the procedure outlined in Section 3.3 for undoped droplets, we calculate the structural and energetic properties of helium droplets of up to size N=5000 atoms with a single central noble gas (Ne, Ar, Xe) dopant atom. To better visualize the effect of these dopants on the droplet, a region of the helium density profile for each of these systems is seen in Fig. 3.8.

Of further interest is the binding energy of the dopant-droplet system, defined to be the difference in total energy of the doped system with respect to the undoped droplet, $E_{bind} = E_{N,pure} - E_{N,doped}$. We find that this quantity plateaus for all three dopants around N $\approx$ 500 atoms when calculated with the Orsay-Paris and Catalonia functionals; however, the Orsay-Trento functional produces binding energies that oscillate about the plateau value until N $\approx$ 3000 atoms (Fig. 3.9). The ordering of the functionals with respect to predicted binding energies conceptually aligns with the results of the pure droplet calculations - the Orsay-Paris functional predicts the most strongly bound droplets in both cases, while the Catalonia functional yields the weakest. Furthermore, these values are compared with the binding energies - referred to in other works as the chemical potential of the impurity - as calculated in [70] and [4], and are displayed in Table 3.8. Note that the literature values have been switched in sign from their original forms; this is because the binding energy defined here is the reverse of that in [70] and [4].

There are two interesting points to be made about the data in Table 3.8. First, the binding energy of the neon dopant calculated using the Orsay-Paris functional in reference [70] is approximately half that of the other binding energy values for neon calculated both in the present work and in [4]. The second point to be made is that the calculations performed in this work indicate that the Catalonia functional universally predicts a weaker binding energy than both the Orsay-Paris or Orsay-Trento functionals; however, this is not seen in the Ne- and Ar- doped droplets of [70] and [4]. We attribute the small differences in binding energy observed to the difference in choice of He-Ne and He-Ar pair potential; the version used in this work is a slightly more modern variant [68] of the Tang-Toennies potential used previously

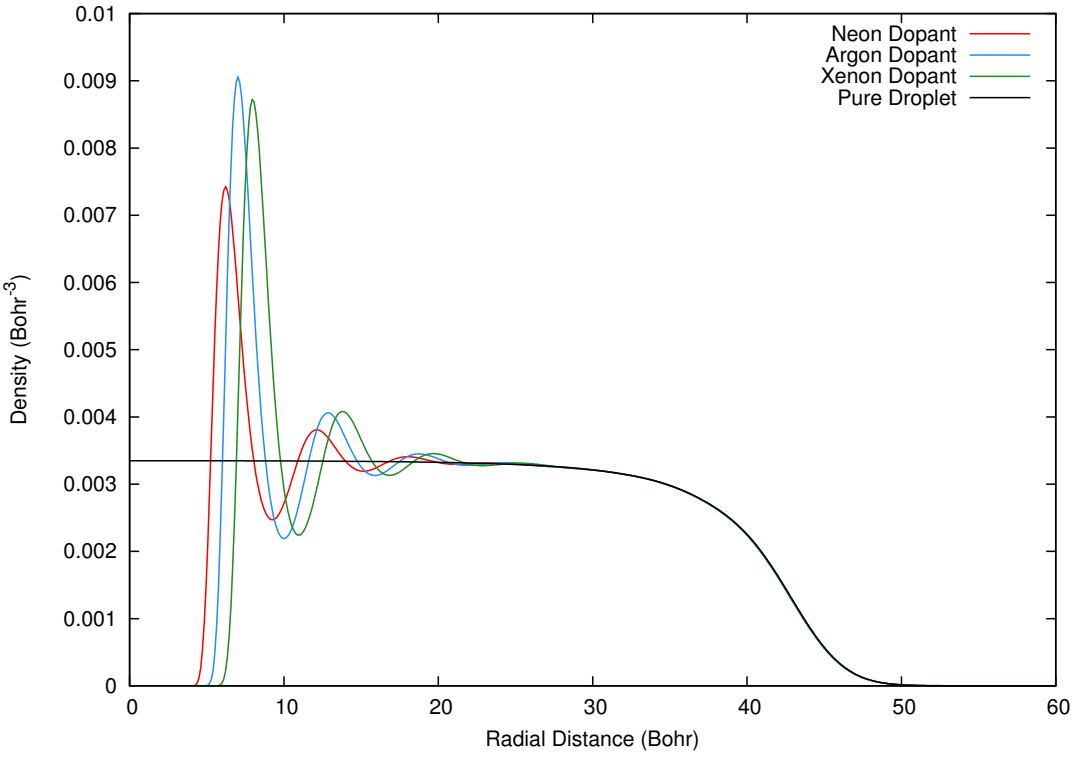


Figure 3.8: Density profiles of Ne-He_{1000}, Ar-He_{1000}, and Xe-He_{1000} droplets using the Orsay-Paris functional. The pure 1000 helium atom droplet profile is included for reference.

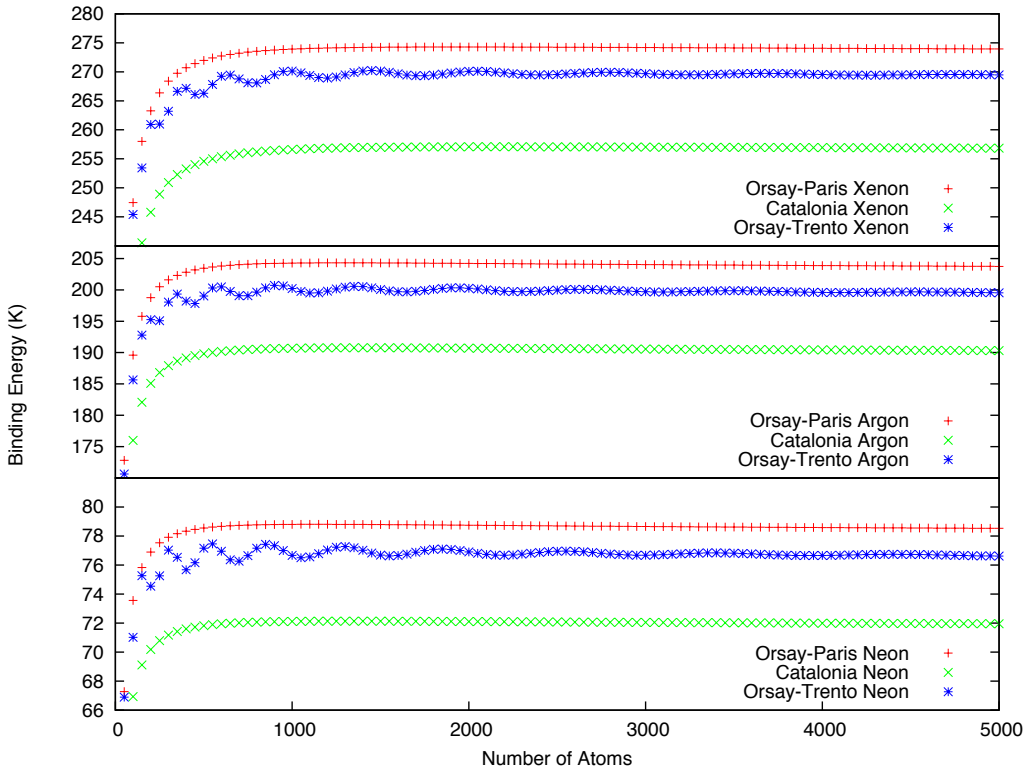


Figure 3.9: Binding energies for Ne, Ar, and Xe doped droplets.

Table 3.8: Asymptotic binding energies for noble gas dopants. The uncertainty in the values of [4] stems from visual estimation of their Fig. 5. The voided entries result from a lack of available literature data, while the entry with an asterisk corresponds to the value for an N=500 droplet.

Functional	Neon, K (Present Work)	Neon, K (Prev. Work)	Argon, K (Present Work)	Argon, K (Prev. Work)	Xenon, K (Present Work)	Xenon, K (Prev. Work)
Orsay-Paris	78	39 [70]	204	195 [70]	274	313 [70]
Catalonia	72	79 ± 3 [4]	191	200 ± 3 [4]	257	289 ± 3 [4]
Orsay-Trento	76	—	199	—	269	278* [71]

[72], although this does not account for the 50% increase in the Orsay-Paris values for neon. For comparison, we calculate a value of $E_{bind} = 70$ K for neon using the Tang-Toennies potential from [72] and the Orsay-Paris functional.

To further investigate the dopant binding energy, the converged helium density profiles are used to decompose the total energy into its principal components, namely, the helium kinetic energy, helium-helium repulsion energy, helium-helium attraction energy, and helium-dopant interaction energy. It should be noted that we separate the the He-He interaction components according to Eqs. 3.1 and 3.6 - the first term of each is taken to be the 'attractive' component, while the remaining part constitutes the 'repulsive' interactions for the appropriate functional. This procedure is done for both the pure and doped droplets, and the components in each case are compared in the same manner as the binding energy, $E_{component(pure)} - E_{component(doped)}$. These quantities converge asymptotically with increasing droplet size - we report the asymptotic values for each energetic component with regard to neon, argon, and xenon dopants in Table 3.9.

Lastly, the chemical potential $\mu = \left(\frac{dE}{dN}\right)$ computed from Eq. 1.5 is investigated as a function of droplet size for each functional. It can be seen in Fig. 3.10 that this quantity is insensitive to the identity of the dopant for sufficiently large (N≈400) droplets, corresponding to the building up of the bulk region beyond the helium solvation layers (Fig. 3.11). Extrapolation of the chemical potential to droplets larger than those explicitly calculated here can be done by considering a form for $\mu(N)$, shown earlier in Eq. 3.34.

Though the parameters in Eq. 3.34 are obtained from fitting the total energy $E(N)$ for pure droplets, they are equally applicable to the chemical potential for doped droplets, as the agreement in the large droplet limit is excellent (see the inset of Fig. 3.10). The residuals of Eq. 3.34 indicate a poor agreement with the chemical potentials of small doped droplets, as can be qualitatively seen in the corresponding region of Fig. 3.10, but rapidly improve to a difference of only $\approx 1 \times 10^{-5}$ K/atom as the droplet size increases beyond N≈1000.

Table 3.9: Composition of the total rare gas-doped droplet energies relative to the pure droplet for the N=5000 case. Destabilizing effects are indicated by a negative sign.

Functional	Δ Kinetic Energy (K)	Δ He-He Repulsion (K)	Δ He-He Attraction (K)	He-Dopant Interaction (K)
		Neon		
Orsay-Paris	-41.0	-25.1	15.0	129.6
Catalonia	-35.4	-20.6	4.5	123.4
Orsay-Trento	-53.3	-21.0	19.9	131.0
		Argon		
Orsay-Paris	-74.4	-64.1	36.9	305.4
Catalonia	-62.9	-55.5	17.9	290.8
Orsay-Trento	-93.6	-54.9	43.7	304.2
		Xenon		
Orsay-Paris	-84.9	-83.1	39.2	402.7
Catalonia	-71.2	-72.3	15.5	384.8
Orsay-Trento	-107.4	-71.4	46.9	401.4

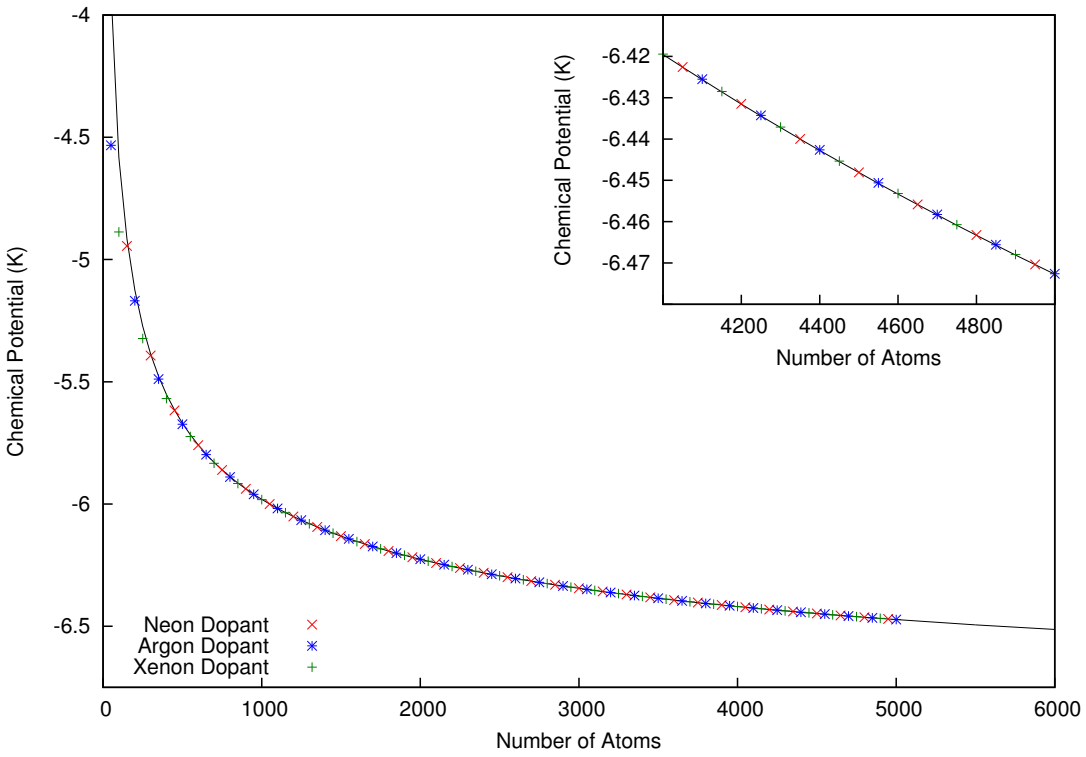


Figure 3.10: Chemical potential of Ne, Ar, and Xe doped droplets as a function of droplet size, calculated using the Orsay-Paris functional. The solid line connects the calculated chemical potential values of the pure droplet case, and is drawn to guide the eye. The inset focuses on the large droplet limit; the agreement with the undoped chemical potential values (black line) is excellent.

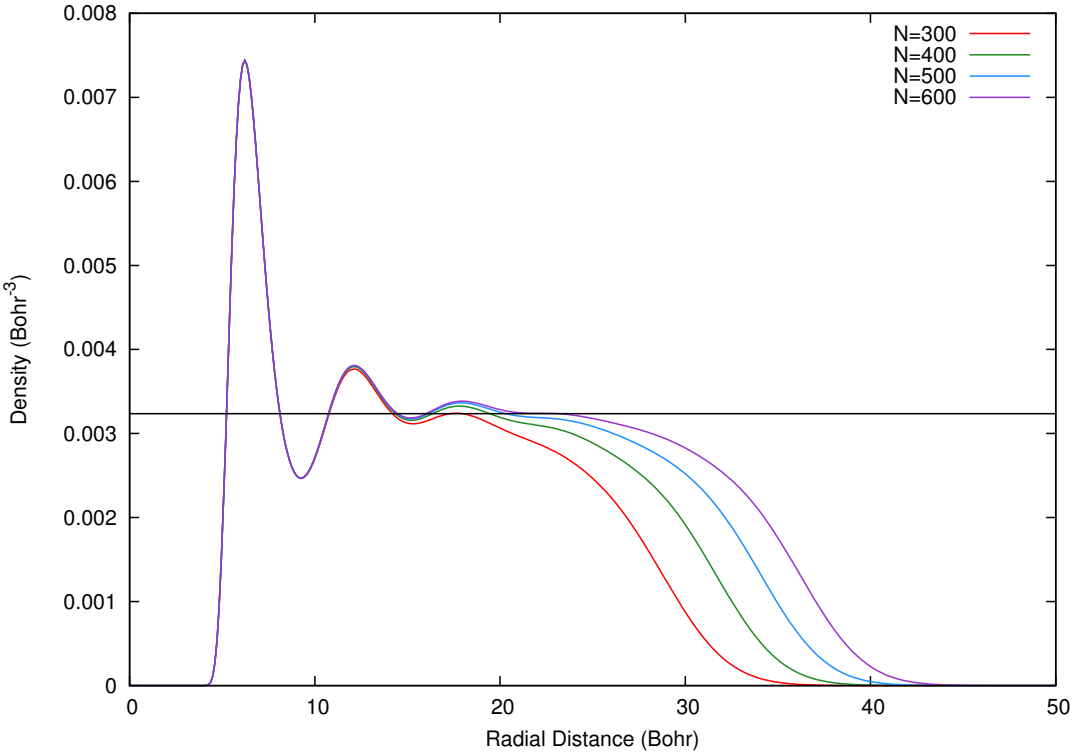


Figure 3.11: Density profiles of Ne-doped droplets with various particle numbers, calculated using the Orsay-Paris functional. The horizontal black line represents the theoretical He-4 bulk density. Note the buildup of the bulk region beyond the solvation layers as the droplet size increases.

3.6 Droplets with Transition Metal Dopants

We continue our evaluation of doped helium droplet systems by including atomic transition metals whose ground state is also spherically symmetric, namely, the coinage metals of silver, gold, and copper. Once again, the dopant-helium interaction enters the functional as $V_{external}$, but now has the form suggested in [73] (Eq. 3.39):

$$V_{Ag,Au,Cu-He}(r - r_{eq}) = -D_e\left(1 + \sum_{i=1}^{n} a_i(r - r_{eq})^i\right)e^{-a_1(r-r_{eq})} \tag{3.39}$$

As with the droplets containing noble gas dopants, calculations are performed on systems of up to N=5000 helium atoms with a single metal atom dopant. The helium density profiles look qualitatively similar to those with the noble gas dopants, indicating that the metal-helium interaction is of the same magnitude as the noble gas-helium interaction (Fig. 3.12). This is further verified by examining the binding energies of the metal-doped droplets over the full range of droplet sizes (Fig. 3.13 and Table 3.10), where binding energy is calculated using the formula described in Section 3.5.

A qualitative assessment of the metal-droplet binding energies shows a later onset of the asymptotic region (N $\approx$ 800) relative to those of the rare gas doped droplets (N $\approx$ 500), and closer agreement between the Orsay-Paris and Orsay-Trento functionals. The total energy of the system can again be decomposed into kinetic and potential parts, as in Section 3.5. The results of this procedure are displayed in Table 3.11.

The chemical potential of the metal-doped droplets is evaluated as described in Section 3.5 for all functionals. Once again, this quantity is insensitive to the identity of the dopant for droplets large enough to have a buildup of density beyond the initial solvation layers (Fig. 3.14, 3.15). The extrapolation of the chemical potential to larger droplet sizes is still satisfied by Eq. 3.34.

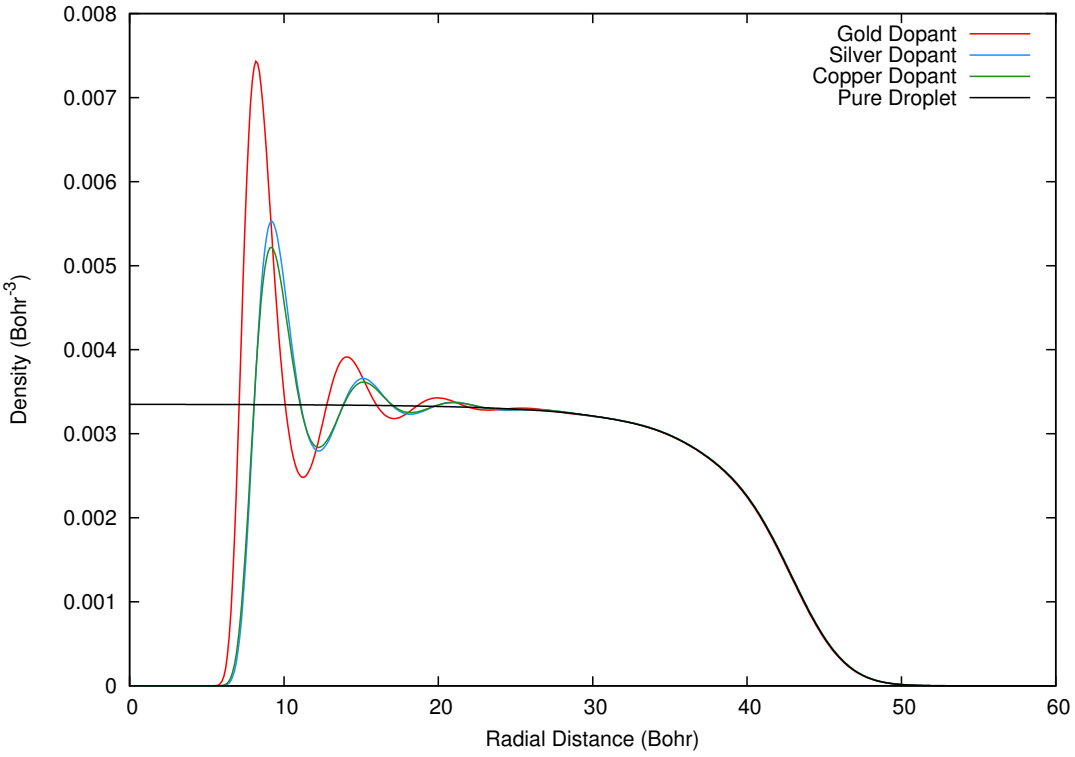


Figure 3.12: Density profiles of Au-He_{1000}, Ag-He_{1000}, and Cu-He_{1000} droplets using the Orsay-Paris functional. The pure 1000 helium atom droplet profile is included for reference.

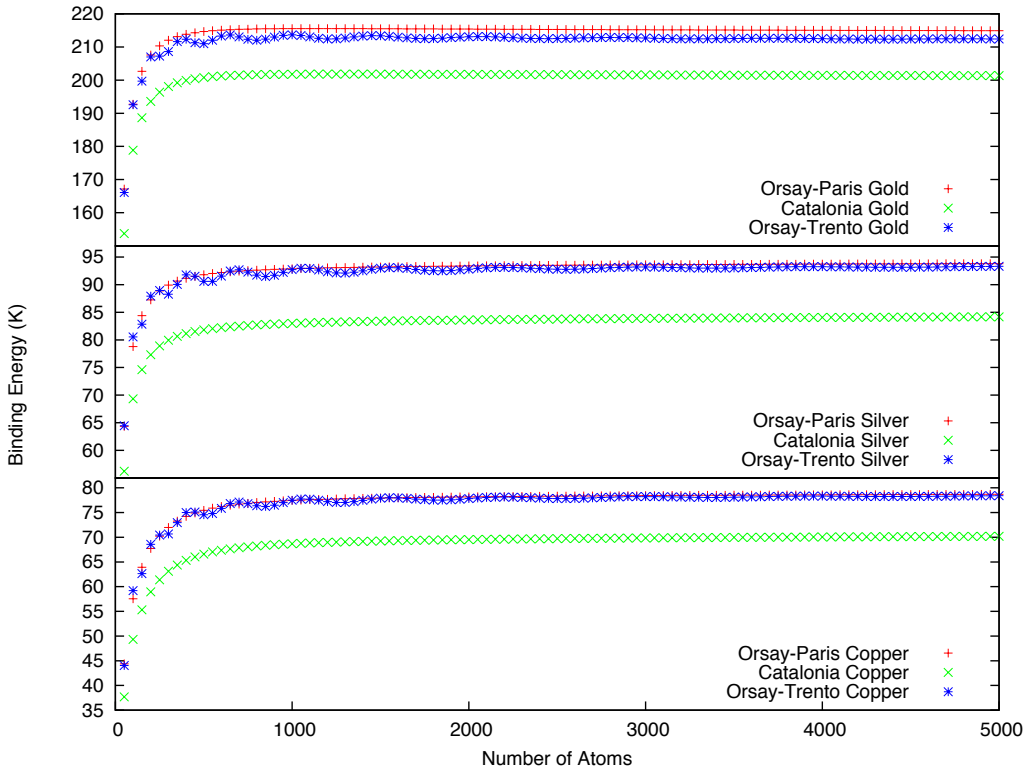


Figure 3.13: Binding energies for Au, Ag, and Cu doped droplets.

Table 3.10: Asymptotic binding energies of gold, silver, and copper dopants for all functionals.

Functional	Gold, K	Silver, K	Copper, K
Orsay-Paris	215	94	79
Catalonia	201	84	70
Orsay-Trento	212	94	79

Table 3.11: Composition of the total transition metal-doped droplet energy relative to the pure droplet for the N=5000 case. Destabilizing effects are indicated by a negative sign.

Functional	Δ Kinetic Energy (K)	Δ He-He Repulsion (K)	Δ He-He Attraction (K)	He-Dopant Interaction (K)
		Gold		
Orsay-Paris	-61.0	-62.6	21.7	316.8
Catalonia	-51.3	-54.0	2.5	304.0
Orsay-Trento	-80.5	-54.0	28.8	318.0
		Silver		
Orsay-Paris	-39.3	-17.3	-28.0	178.4
Catalonia	-33.9	-10.7	-43.1	172.0
Orsay-Trento	-56.7	-18.3	-14.6	183.0
		Copper		
Orsay-Paris	-34.2	-12.5	-30.8	156.2
Catalonia	-29.8	-6.7	-44.4	151.0
Orsay-Trento	-50.0	-14.6	-17.4	160.6

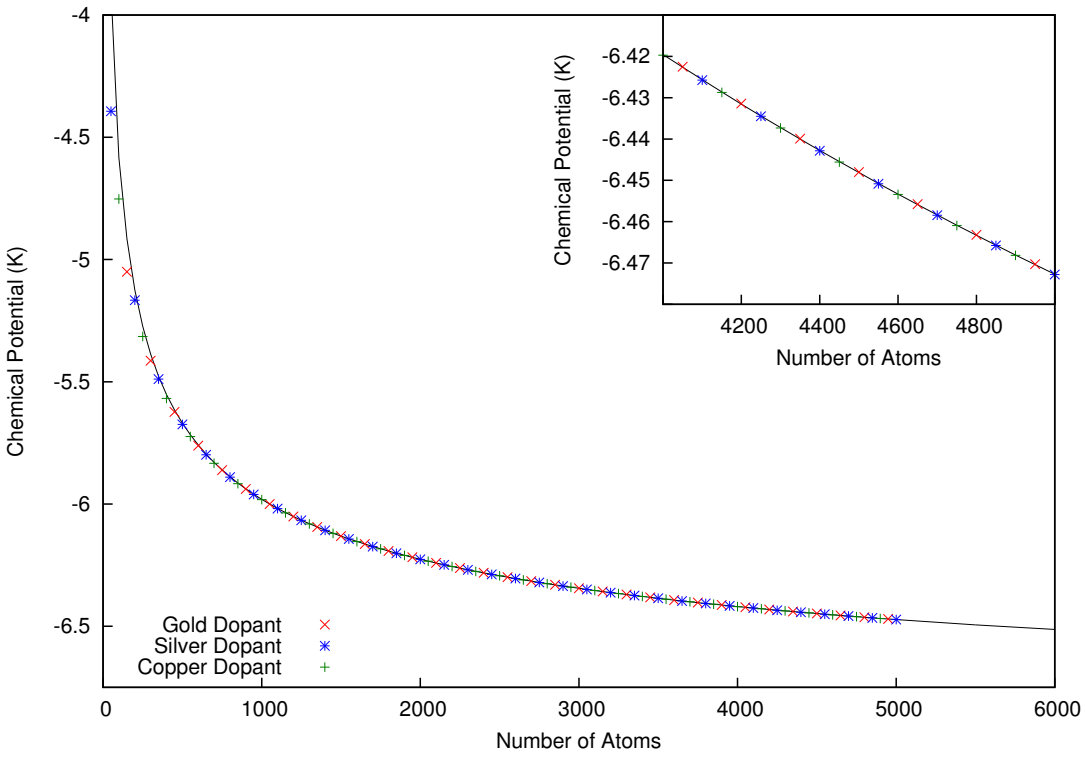


Figure 3.14: Chemical potential of Au, Ag, and Cu doped droplets as a function of droplet size, calculated using the Orsay-Paris functional. The solid line connects the chemical potential values of the pure droplet case, and is drawn to guide the eye. The inset again focuses on the large droplet limit; the agreement with the undoped chemical potential values (black line) is again excellent.

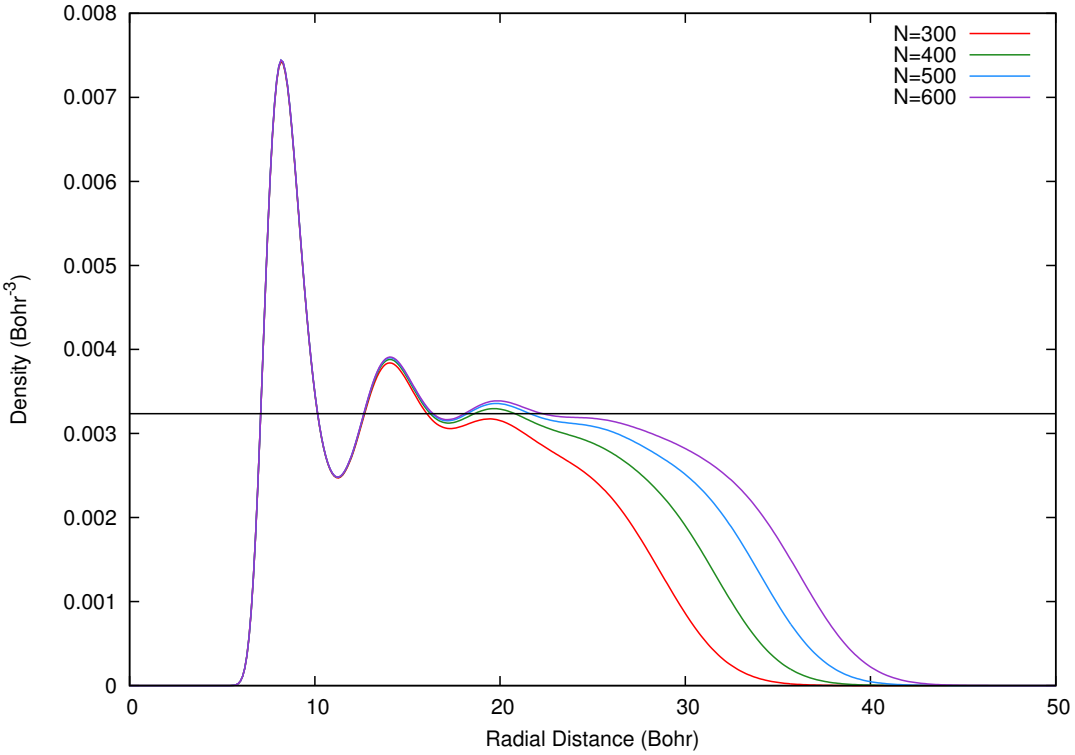


Figure 3.15: Density profiles of Au-doped droplets with various particle numbers, calculated using the Orsay-Paris functional. The horizontal black line represents the theoretical He-4 bulk density. Note the buildup of the bulk region beyond the solvation layers as the droplet size increases.

3.7 Application to He Droplet Calorimetry

We turn our attention now to the field of helium droplet calorimetry; specifically, the application of our results to improving theoretical predictions of the evaporation events that follow dopant pick-up. Consider a log-normal distribution of helium droplet sizes typical of an experimental droplet beam apparatus [34] with a width governed by σ and an average droplet size $\langle N_0 \rangle$ (Eq. 3.40). The capture of dopant atoms or molecules by droplets above a critical size then evaporates helium atoms according to the energy released by the capture event, leading to a new distribution of smaller droplet sizes with an average $\langle N_1 \rangle$. The number of helium atoms evaporated (ΔN_{evap}) is then a function of both the rate of energy transfer into the droplet and the chemical potential of the droplet itself. Theoretical calculations of ΔN_{evap} have in the past utilized the bulk chemical potential as an approximation to the droplet value; however, our chemical potential fitting equations indicate that this approach is prone to error, particularly for smaller droplets, where the predicted value of μ is very different from the bulk value.

$$f(N) = \frac{1}{\sigma\sqrt{2\pi}} exp\Big[- \frac{(\ln N - \alpha)^2}{2\sigma^2} \Big] \tag{3.40}$$

To correct for the size-dependence of the chemical potential in determining the extent of helium evaporation, we use Eq. 3.41 for a droplet of initial size N_0 receiving an energetic transfer of magnitude ΔE:

$$N_1 = N_0 + \frac{\Delta E}{\mu(N_0)} \tag{3.41}$$

The value of μ entering this equation is predicted from the fits of Eq. 3.34 and Table 3.7; we find no appreciable difference in the predicted size N_1 with respect to fitting equations derived from the different functionals, and thus use the Orsay-Paris values for the sake of brevity in the remainder of the section. A subtle assumption of Eq. 3.41 is that the energy imparted into the droplet evaporates all of the appropriate

helium atoms simultaneously, resulting in a direct transition from a size N_0 droplet to a size N_1 droplet; we describe this scenario as 'fast' evaporation. The opposite extreme can be thought of as an evaporation of helium atoms one-by-one from the droplet, where the decay from a size N_0 droplet to a size N_1 droplet experiences every intermediate size; we call this 'slow' evaporation. The impact of assuming 'fast' evaporation in the calculation of N_1 is addressed later in this work.

Although Eq. 3.41 is useful as a simple illustration, it does not reflect the desire of experiment to predict the energy transfer ΔE for an observed doping process. To make our results useful as a predictive tool, we emulate a typical spectroscopic experiment by creating a lognormal distribution of droplets according to an average initial value $\langle N_0 \rangle$ with an appropriate width parameter σ. We then impart a specific amount of energy to each droplet, and compute its final size using Eq. 3.41. If the droplet survives the energetic injection (i.e. doesn't completely evaporate), it is incorporated into a new distribution of droplet sizes with an average $\langle N_1 \rangle$. By replacing the quantities N_0 and N_1 in Eq. 3.41 with their averages calculated from the two distributions, an apparent ΔE can be calculated and compared to the inputted quantity. Finally, the difference $\langle N_0 \rangle - \langle N_1 \rangle$ yields an observed ΔN_{evap} for the simulated 'experiment'. By choosing values for $\langle N_0 \rangle$ and ΔE in such a way that they are aligned with those observed in actual helium depletion studies, we employ the above method to generate improved theoretical values for ΔE and ΔN_{evap} for a wide array of collision events (Tables 3.12 - 3.14).

The initial distributions of droplet sizes found in Tables 3.12 - 3.14 are comprised of 10,000,000 droplets, and randomly generated according to the log-normal parameters σ and α (see Eq. 3.40); to ensure these values are relevant to experimentally observed droplet distributions, we fit the values of α as a logarithmic function of average droplet size reported by Harms, Toennies, and Dalfovo [34]. It is then straightforward to convert a series of representative average droplet sizes to their respective α parameters. We also select three different quantities for σ to account for distributions of various widths. Because of the stochastic nature of this procedure, the computed

Table 3.12: Values of ΔN_{evap} predicted for lognormal (σ=0.468) distributions with various average droplet sizes (represented in the left-hand column by their α parameter) and injected energies. The values in parentheses correspond to the observed $\langle N_1 \rangle$ for the particular distribution generated. Voided entries indicate complete evaporation of the initial average droplet by the energy transferred.
[a]Energetic values taken from Table III of reference [5].
[b]Energetic values taken from Table I of reference [6].

α	$\langle N_0 \rangle$	1323 K[a]	2565 K[a]	4398 K[a]	69630 K[b]	150860 K[b]
7.75	2590 ± 1	211 (2379)	409 (2181)	687 (1903)	1054 (1536)	-783 (3373)
7.93	3101 ± 1	210 (2891)	406 (2695)	690 (2411)	1386 (1715)	1246 (1855)
8.30	4490 ± 2	206 (4284)	400 (4090)	685 (3805)	2307 (2183)	1502 (2788)
8.91	8263 ± 2	202 (8061)	392 (7871)	672 (7591)	4529 (3734)	4120 (4143)
9.28	11962 ± 3	200 (11762)	388 (11574)	664 (11298)	6291 (5671)	6431 (5531)
9.55	15670 ± 4	198 (15472)	385 (15285)	660 (15010)	7617 (8053)	8620 (7050)
9.78	19720 ± 4	197 (19523)	383 (19337)	656 (19064)	8629 (11091)	10793 (8927)
10.02	25073 ± 5	196 (24877)	381 (24692)	653 (24420)	9422 (15651)	13273 (11800)

Table 3.13: Values of ΔN_{evap} predicted for lognormal (σ=0.573) distributions with various average droplet sizes (represented in the left-hand column by their α parameter) and injected energies. The values in parentheses correspond to the observed $\langle N_1 \rangle$ for the particular distribution generated. Voided entries indicate complete evaporation of the initial average droplet by the energy transferred.
[a]Energetic values taken from Table III of reference [5].
[b]Energetic values taken from Table I of reference [6].

α	$\langle N_0 \rangle$	1323 K[a]	2565 K[a]	4398 K[a]	69630 K[b]	150860 K[b]
7.75	2736 ± 1	212 (2524)	405 (2331)	649 (2087)	447 (2289)	-1078 (3814)
7.93	3275 ± 1	210 (3065)	405 (2870)	667 (2608)	735 (2540)	-299 (3574)
8.30	4742 ± 2	207 (4535)	400 (4342)	680 (4062)	1528 (3214)	387 (4355)
8.91	8724 ± 3	202 (8522)	392 (8332)	672 (8052)	3513 (5211)	2568 (6156)
9.28	12635 ± 5	200 (12435)	388 (12247)	665 (11970)	5121 (7514)	4627 (8008)
9.55	16548 ± 9	199 (16529)	385 (16163)	660 (15888)	6404 (10144)	6583 (9965)
9.78	20832 ± 8	198 (20634)	383 (20449)	657 (20175)	7474 (13358)	8515 (12317)
10.02	26482 ± 6	196 (26286)	381 (26101)	653 (25829)	8458 (18024)	10811 (15671)

Table 3.14: Values of ΔN_{evap} predicted for lognormal (σ=0.612) distributions with various average droplet sizes (represented in the left-hand column by their α parameter) and injected energies. The values in parentheses correspond to the observed $\langle N_1 \rangle$ for the particular distribution generated. Voided entries indicate complete evaporation of the initial average droplet by the energy transferred.
[a]Energetic values taken from Table III of reference [5].
[b]Energetic values taken from Table I of reference [6].

α	$\langle N_0 \rangle$	1323 K[a]	2565 K[a]	4398 K[a]	69630 K[b]	150860 K[b]
7.75	2800 ± 1	211 (2589)	401 (2399)	628 (2172)	169 (2631)	-1157 (3957)
7.93	3352 ± 1	210 (3142)	402 (2950)	652 (2700)	456 (2896)	-807 (4159)
8.30	4852 ± 2	207 (4645)	400 (4452)	674 (4178)	1227 (3625)	-157 (5009)
8.91	8931 ± 3	202 (8729)	392 (8539)	672 (8259)	3102 (5829)	1898 (7033)
9.28	12929 ± 3	200 (12729)	388 (12541)	665 (12264)	4657 (8272)	3879 (9050)
9.55	16938 ± 4	199 (16739)	385 (16553)	660 (16278)	5917 (11021)	5745 (11193)
9.78	21320 ± 6	198 (21122)	383 (20937)	657 (20663)	6995 (14325)	7612 (13708)
10.02	27098 ± 10	197 (26901)	381 (26717)	653 (26445)	8026 (19072)	9815 (17283)

$\langle N_0 \rangle$ value differs slightly from the exact mean that would be computed from an infinitely large ensemble. This is reflected in Tables 3.12 - 3.14 by applying an uncertainty to $\langle N_0 \rangle$ calculated from the five independent ensembles generated for a specific pair of σ and α parameters over the course of populating each row.

Note that the values found in Tables 3.12 - 3.14 are calculated by assuming 'fast' energy transfer into the droplet. To verify the validity of this approximation, we take an initial distribution with an average droplet size $\langle N_0 \rangle \approx 2600$ (σ=0.468, α=7.75) and compute two new distributions after energy transfers of 1323 K, 2565 K, and 4398 K (i.e. the first row of Table 3.12). The first of these distributions is calculated according to the methodology described above, in which the energy is transferred to the droplet assuming the chemical potential corresponding to the initial droplet size. The second distribution is calculated assuming atom-by-atom helium evaporation, adjusting the chemical potential to align with the new droplet size after every evaporation step - this is the extreme case of 'slow' energy transfer. The resultant 'fast' and 'slow' distributions are seen in Figs. 3.16 - 3.18; the difference between the two is negligible in each of these cases.

The same 'fast' and 'slow' distributions are then calculated for the remaining large energies of Table 3.14 (69630 K and 150860 K) - in each of these cases a noticable difference between the distributions of the two rates of evaporation appears. This is due to the smaller chemical potentials of small droplets relative to the large droplets necessary to incorporate the injected energies - the 'fast' assumption uses the initial large droplet chemical potential, which doesn't accurately reflect the behavior of the smaller droplets present after evaporating helium. Thus, the 'slow' evaporation scenario, which fully incorporates each droplet's chemical potential, results in the loss of more helium from the droplet. This is recognized in Figs. 3.19 and 3.20 by the lower height of the 'slow' evaporation distribution curve relative to the 'fast' one.

Realizing that the actual rate of evaporation for any droplet-dopant system likely lies somewhere between our 'fast' and 'slow' extremes, the agreement seen in Figs. 3.16 - 3.18 suggests that we can use Eq. 3.41 in circumstances where the energy

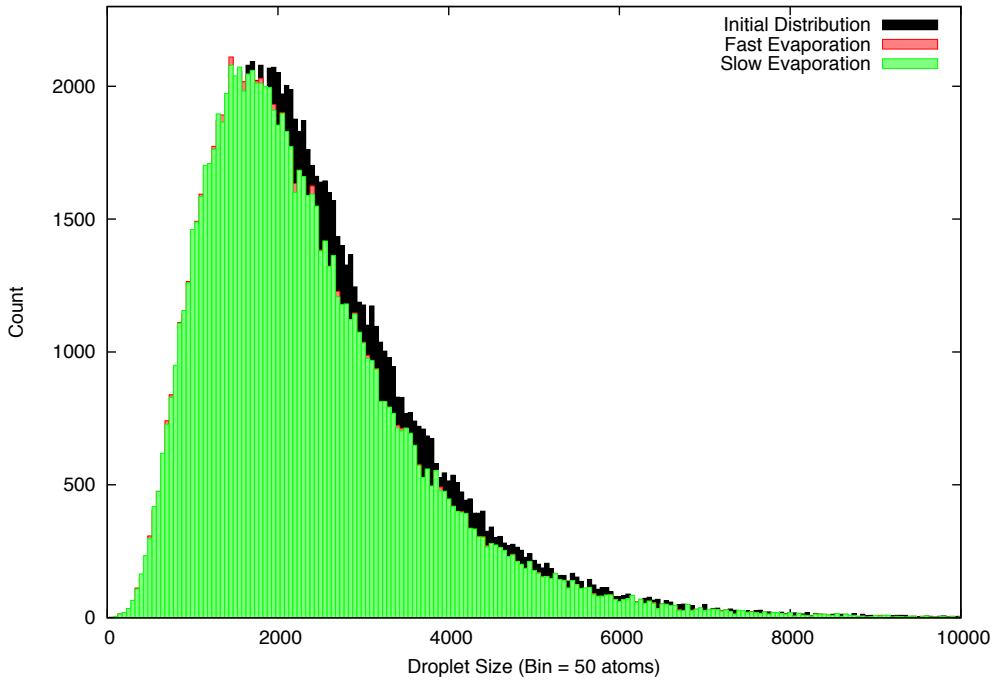


Figure 3.16: Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 2600$ injected with an energy of 1323 K. The difference in average values between the slow and fast distributions is 1 atom.

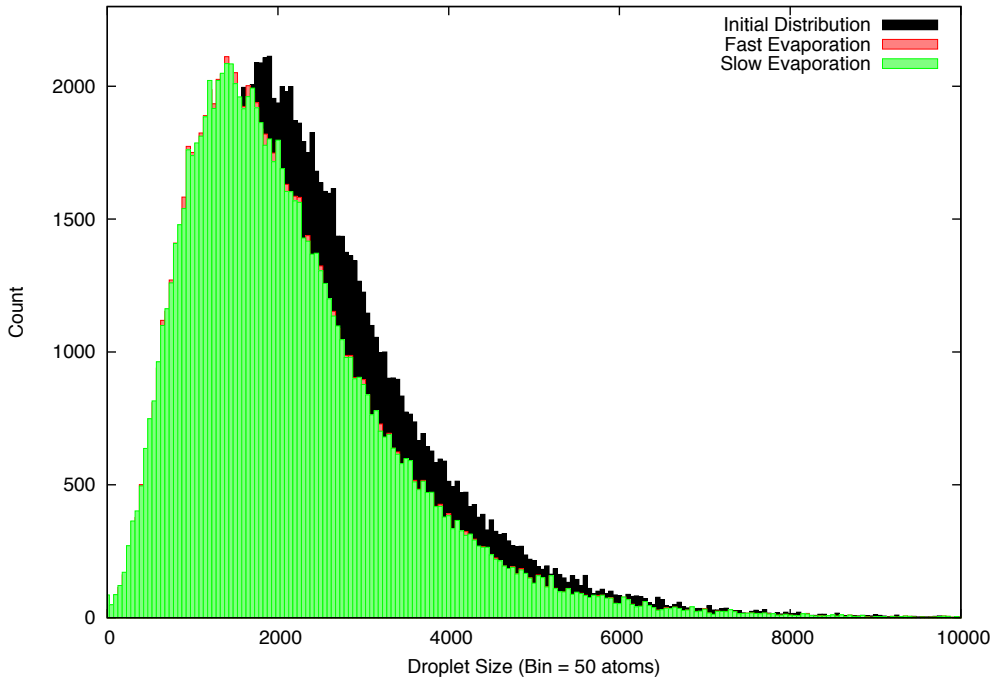


Figure 3.17: Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 2600$ injected with an energy of 2565 K. The difference in average values between the slow and fast distributions is 2.3 atoms.

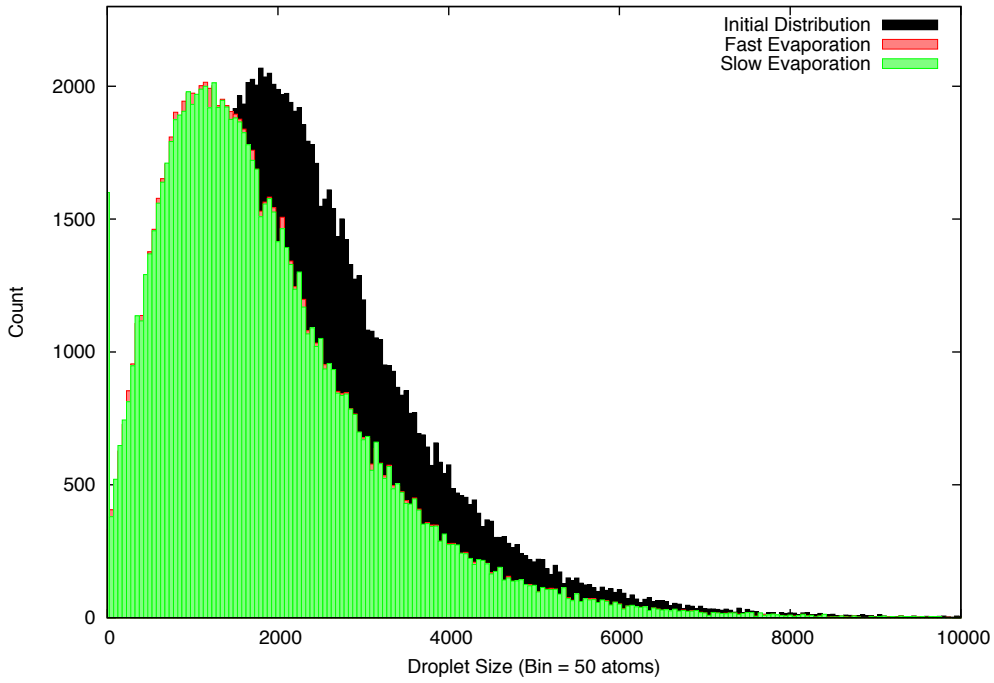


Figure 3.18: Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 2600$ injected with an energy of 4398 K. The difference in average values between the slow and fast distributions is 3 atoms.

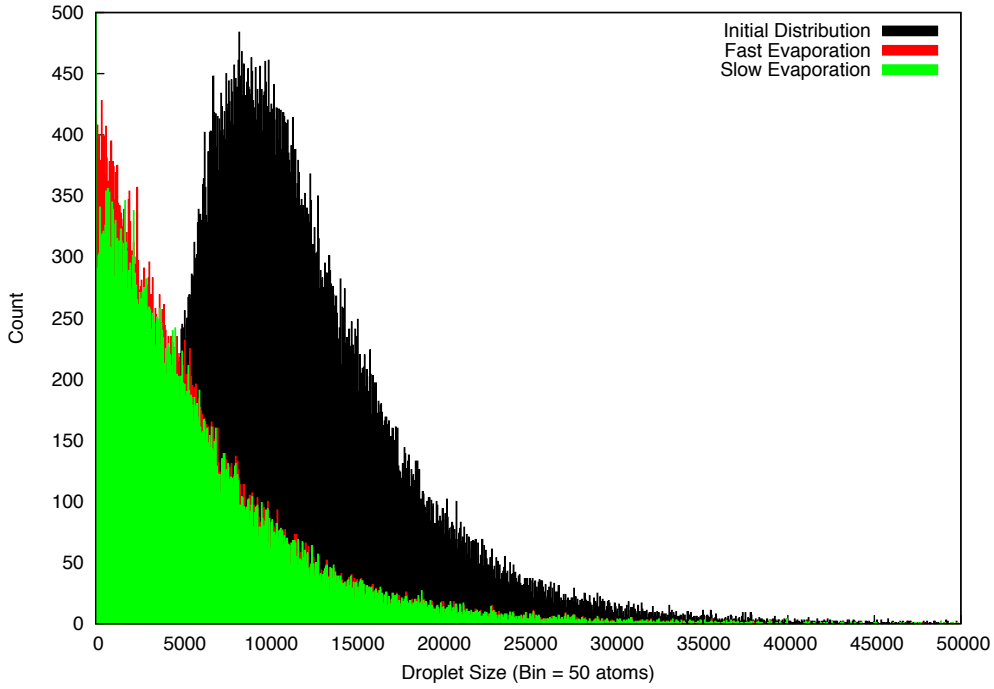


Figure 3.19: Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 12000$ injected with an energy of 69630 K. The difference in average values between the slow and fast distributions is 233 atoms.

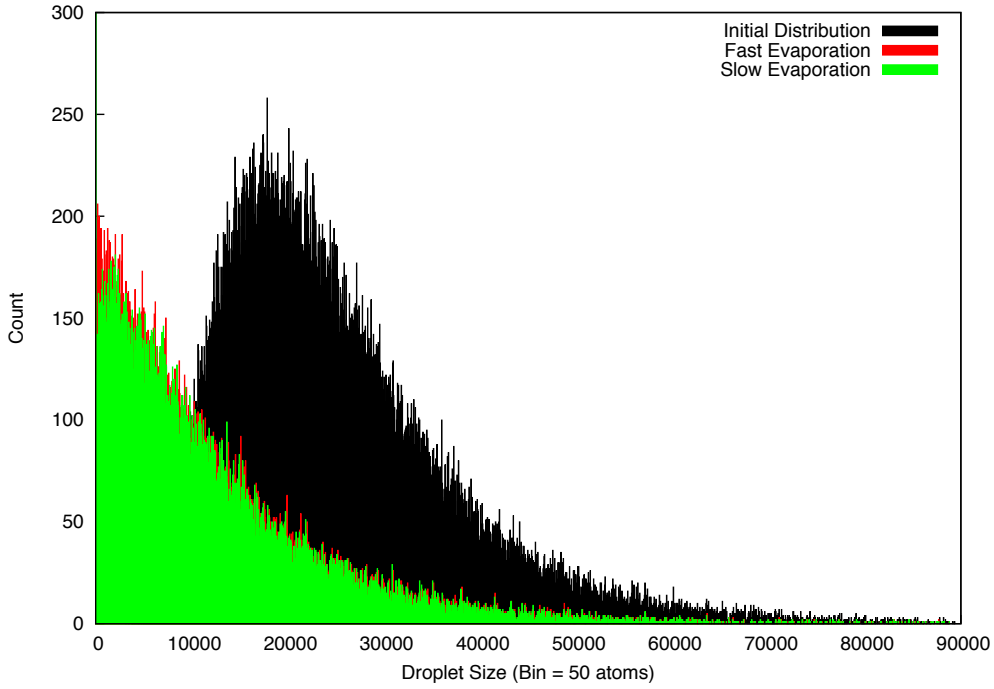


Figure 3.20: Fast versus slow evaporation of an initial droplet distribution with average size $\langle N_0 \rangle \approx 25000$ injected with an energy of 150860 K. The difference in average values between the slow and fast distributions is 365 atoms.

transferred is not great; however, the average droplet values predicted in instances of large energy transfers will likely be erroneously small, due to the extra evaporation of helium from small droplets not accounted for in the 'fast' evaporation approximation. The similarity in shape of the 'fast' and 'slow' curves of Figs. 3.19 and 3.20 suggests that sufficiently increasing the droplet sizes of the initial distribution (i.e. so that the final distribution still contains large droplets, rather than small ones) will better resolve the differences between the resultant 'fast' and 'slow' average sizes. It is also relevant to point out that, although the analyses shown are for the smallest applicable values of σ and α, an increase in either parameter will increase the average droplet size; however, the largest discrepancy between 'fast' and 'slow' evaporation is expected for small droplets, where the chemical potential varies most rapidly with droplet size. Thus, the scenarios shown represent the most likely instances to see disagreement between these two extremes.

As a final point, we remark that in Figs. 3.19 and 3.20, a non-negligible fraction of the droplets have been completely evaporated in each case, resulting in curves that are no longer described by a log-normal distribution. This explains the unusual behavior seen in the calculated $\langle N_1 \rangle$ values of the relevant columns in Tables 3.12 - 3.14, where the observed average value for the new distribution increases with increasing droplet size. Certain entries of these problematic columns even show an apparent gain in helium after the simulated pick-up event (indicated by a negative ΔN) - this is due to the complete evaporation of the entirety of the initial distribution with the exception of a few very large droplets in the tail region. This very small handful of surviving droplets then characterizes the average $\langle N_1 \rangle$ of the final size distribution. These problems are indicative of a larger problem present in helium calorimetry experiments as a whole, as a log-normal description of the helium droplets both before and after dopant pick-up is sometimes assumed to be appropriate. However, our results indicate that the log-normal character of the droplet size distribution is not necessarily preserved after pick-up events with large energy transfers, and so great care must be taken in choosing sufficiently large initial droplets. Calculations of the

type carried out in this section can aid in predicting an average minimum size for the initial droplets, which in turn can mitigate the problems associated with the complete evaporation of the majority of the initial droplet distribution throughout the course of an experiment.

3.8 Discussion and Conclusions

The energetic and structural properties of He-4 droplets have been thoroughly investigated using He-DFT methods with three commonly-used modern helium density functionals. The calculated chemical potential and per-particle energy values for the pure droplets are in excellent agreement with both prior density functional studies and asymptotic (bulk) results as verified via the energetic fitting of Eq. 3.33. Furthermore, this fitting predicts droplet surface tensions that are reasonably close to experimental values.

Subsequent calculations of doped (noble gas, transition metal) helium droplets are then utilized to both assess the behavior of the different helium functionals under external stimuli and explore the physical changes brought about by the presence of atomic impurities. Regarding the former, we find that the Orsay-Paris functional consistently predicts the strongest binding of the dopant atoms, on the order of 10-15 K higher than the weakest predicted values of the Catalonia functional. Neon is the only abnormality here, as the difference in binding energies predicted by the Orsay-Paris and Catalonia functional is only 6 K. Calculations using the Orsay-Trento functional yield binding energies that are consistently weaker than the Orsay-Paris results by at most 5 K. Recognizing that the pure droplet calculations yield energies that are very close in value to one another across the three functionals, we inspect the energetic components of the doped droplets to rationalize the apparent ordering of functionals with respect to binding energy prediction. In each scenario, the Orsay-Paris functional predicts a larger magnitude for every energetic component relative to the Catalonia one (i.e. a more positive He-He repulsion and a more negative

He-He attraction); the Orsay-Trento functional generates the largest kinetic energy values, the smallest (magnitude) He-He interactions, and He-impurity interactions that generally agree with the Orsay-Paris results to within a few Kelvin.

The decomposition of the total energy into its components reveals that the binding energy of the droplet is in every case less than the energy gained from favorable He-dopant interactions. This is due to increases in helium kinetic energy and He-He repulsive interactions associated with the high-density first solvation layer surrounding the dopant, both of which are destabilizing. It is also interesting to note that, in the case of Ag and Cu dopants, the magnitude of the attractive He-He interactions also decreases relative to the pure droplet.

Perhaps the most interesting facet of this work is the behavior of the chemical potential for large droplets; specifically, its deviation from the theoretical bulk limit (≈ -7.15 K). This phenomenon is expected to directly impact the quality of calculations frequently used in analysis of helium droplet calorimetry experiments, where determination of the distribution of droplet sizes before and after pick-up of dopants is of interest. We have shown a new manner of predicting the number of helium atoms evaporated from a droplet after energetic transfer from a dopant by way of the chemical potentials predicted from our DFT calculations. Using this method, initial droplet size distributions inspired by experimental data have been injected with energy representative of various pick-up events and analyzed in terms of the resultant size distribution and number of helium atoms evaporated, thereby creating a lookup table of sorts. Furthermore, we verify that this treatment is valid regardless of the rate of helium evaporation, provided that the initial and final distributions are both approximately log-normal.

The simulated pick-up events with the largest energies are shown to completely evaporate a substantial number of the helium droplets (even in the case of our largest $\langle N_1 \rangle$ values), resulting in distributions that are no longer log-normal in nature. This presents a challenge to the field of helium spectroscopy, which assumes log-normal descriptions of the helium droplet sizes are appropriate throughout the course of an

experiment. The results of this work help to reduce the impact of the above problem by suggesting a procedure to calculate the approximate minimum size needed to retain most of the helium droplets after capture events, although we look to establish a more accurate description of the post-capture event droplet size distribution in a future work.

Finally, we note that the binding energy used in reference [5] for argon is approximately 100 K larger than our calculated values using the Orsay-Paris, Catalonia, and Orsay-Trento density functionals. By substituting our calculated value (204 K, Orsay-Paris) into the droplet evaporation method described above, we calculate a new ΔN_{evap} of 192 atoms, a decrease from the predicted 210 atoms calculated using the same method with the previously reported value. Both of these quantities still represent an increase over the value of 184 evaporated atoms predicted using the bulk chemical potential.

Chapter 4

He-DFT in Unconventional Environments

In the previous chapter, the properties of large ($N > 500$) droplets pertinent to predicting and interpreting experimentally observed distributions were evaluated using various functionals in the He-DFT scheme. The behavior of these functionals is expected to be - and in fact is - quite good in this size regime, as they depend upon phenomenological parameters derived from bulk superfluid helium. However, in many circumstances, the systems under study do not resemble bulk helium, and so the ability of He-DFT to accurately predict properties in such conditions is not necessarily guaranteed. It is therefore the aim of this chapter to assess the capabilities of He-DFT in these 'extreme' cases.

4.1 Small Droplets

Perhaps the most obvious scenario that represents helium deviating from bulk behavior involves having very small ($N < 100$) droplets. In these systems, the helium density never reaches the bulk fluid value ($\sim$0.0218 Å^{-3}), even at the centers of the droplets. Because of the small number of atoms present, quantum Monte Carlo

methods are the typical theoretical approaches used in these circumstances, as they are capable of generating highly accurate energetic and structural data (see Section 1.4). This presents an excellent opportunity to benchmark the He-DFT approach against other widely accepted and highly accurate computational methods.

We first consider the case of undoped helium clusters (the term 'droplet' is replaced with 'cluster' here, as 'droplet' begins to lose meaning in the limit of very few atoms); the energetic data collected are displayed in Table 4.1. Our benchmarking standards are diffusion Monte Carlo (DMC) calculations performed by Bressanini et al.[74] and by Mella et al.[7]

The results of Table 4.1 show consistent underbinding of the clusters predicted by all three functionals when compared to the DMC values, although better agreement between the two methods is seen for the larger clusters. A portion of this discrepancy can certainly be ascribed to the differences between the Tang-Toennies-Yiu (TTY) He-He pair potential[75] used in the DMC simulations and the Lennard-Jones 12-6 pair potential incorporated into each of the three functionals. For instance, the predicted well depth (ϵ) and location (r_ϵ) are both larger in the TTY potential (10.985 K and 5.6165 bohr versus 10.22 K and 5.4216), which qualitatively corresponds to a more bound cluster. We naively attempt to account for the differences in potential functions used by introducing a multiplicative factor equal to the ratio of attractive well depths $\epsilon_{TTY}/\epsilon_{LJ}$; application of this ratio to the bottom row of Table 4.1 yields energetic values of -101.97661 K (-1.4%), -97.28293 K (3.3%), and -100.78701 K (0.19%) for the Orsay-Paris, Catalonia, and Orsay-Trento functionals, respectively. This procedure recaptures part of the binding character seen in the DMC simulations, although the agreement for the smaller cluster values is still less than excellent. As an aside, it could perhaps be fruitful to introduce a helium density functional that is not dependent on the Lennard-Jones He-He interaction, but instead upon one of the more recent curves fitted from experimental or *ab initio* data [48, 75, 76].

A second method of extracting small-cluster energetic information from He-DFT simulations is via extrapolation of the droplet energy fits (Eqs. 3.33 and Table 3.7);

Table 4.1: Energies for small He_N clusters calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals and an imaginary time step $\Delta\tau = 2500$ a.u. Benchmarking values are calculated using DMC methods by various authors (see text); the uncertainty in these reported values is maintained in the conversion from cm^{-1} to K via conservation of significant figures. Parenthetical quantities indicate percent error with respect to the DMC calculations.

N	Orsay-Paris (K)	Catalonia (K)	Orsay-Trento (K)	DMC (K)
5	0.13958 (110.8%)	0.48101 (137.1%)	0.63565 (149.0%)	-1.2971[74]
6	-0.93033 (59.8%)	-0.51428 (77.8%)	-0.26261 (88.6%)	-2.31319[74]
7	-2.08854 (41.2%)	-1.56888 (56.0%)	-1.42699 (60.0%)	-3.56901[74]
8	-3.47946 (32.2%)	-2.85974 (44.3%)	-2.71029 (47.2%)	-5.1339[7]
11	-8.48338 (19.1%)	-7.49730 (28.5%)	-7.63025 (27.2%)	-10.483[74]
12	-10.41364 (17.3%)	-9.29271 (26.2%)	-9.55505 (24.1%)	-12.585[7]
20	-29.66868 (10.5%)	-27.50126 (17.0%)	-28.74096 (13.3%)	-33.151[74]
25	-44.19480 (0.28%)	-41.42332 (6.0%)	-43.22518 (1.9%)	-44.07[7]
30	-60.07803 (0.73%)	-56.74302 (6.2%)	-59.06611 (2.4%)	-60.518[7]
40	-94.87492 (5.7%)	-90.50811 (10.0%)	-93.76816 (6.8%)	-100.6[7]

however, these values are much less accurate than the direct calculations of the same quantities. For example, the smallest cluster of Table 4.1, $N = 5$, is predicted to have an energy of -7.13 K using the fitting parameters derived from the Orsay-Paris calculations (-7.33 K and -6.41 K with the Catalonia and Orsay-Trento parameters, respectively).

Chemical potentials for the small clusters with consecutive N values can be calculated[59] according to Eq 4.1 using the available DMC data, allowing for a second metric of evaluation for the density functionals (recall that the chemical potential is the eigenvalue of the Schrödinger-like equation evaluated with the self-consistent density in the density functional scheme). These data are presented in Table 4.2.

$$\mu(N) = E(N) - E(N-1) \tag{4.1}$$

Despite the poor quality of results for the first entries of Table 4.1, it is encouraging that the He-DFT chemical potential values are in better agreement with the values derived from the DMC data. The agreement of the energies to within roughly 10% beginning around $N = 20$ in Table 4.1 (remember, the form of the He-He interaction is different between methods!) seems to indicate a lower bound for the validity of the density functional approach with respect to number of atoms present in the simulated cluster. This may be due to the presence of a second-shell type effect, where the introduction of additional atoms to the forming droplet causes an encapsulation of an inner 'core' by a second helium layer. It is still quite remarkable that the He-DFT method is capable of handling these systems, as the helium density never reaches bulk value for any of the clusters calculated in Table 4.1 (see Fig. 4.1). For reference, the first instance of an undoped droplet achieving bulk density in our simulations is around $N = 100$ atoms.

Continuing our analysis of He-DFT in the regime of small droplets, we look to test the performance of the various functionals on a system containing a dopant atom. We choose neutral silver for this purpose both because of the availability of diffusion

Table 4.2: Chemical potentials for small He_N clusters calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals with an imaginary time step $\Delta\tau = 2500$ a.u. Benchmarking values are derived from calculated DMC values by various authors (see text) according to Eq. 4.1, where precision is maintained to the lowest number of decimal places reported. Parenthetical quantities indicate percent error with respect to the DMC calculations.

N	Orsay-Paris (K)	Catalonia (K)	Orsay-Trento (K)	DMC (K)
6	-0.88416 (13.0%)	-0.75026 (26.2%)	-0.78769 (22.5%)	-1.0161[74]
7	-1.11271 (11.4%)	-0.97239 (22.6%)	-1.03905 (17.3%)	-1.25582[74]
8	-1.32290 (15.5%)	-1.17967 (24.6%)	-1.25381 (19.9%)	-1.5649[7]
12	-1.96531 (6.5%)	-1.81977 (13.4%)	-1.90516 (9.4%)	-2.102[7]

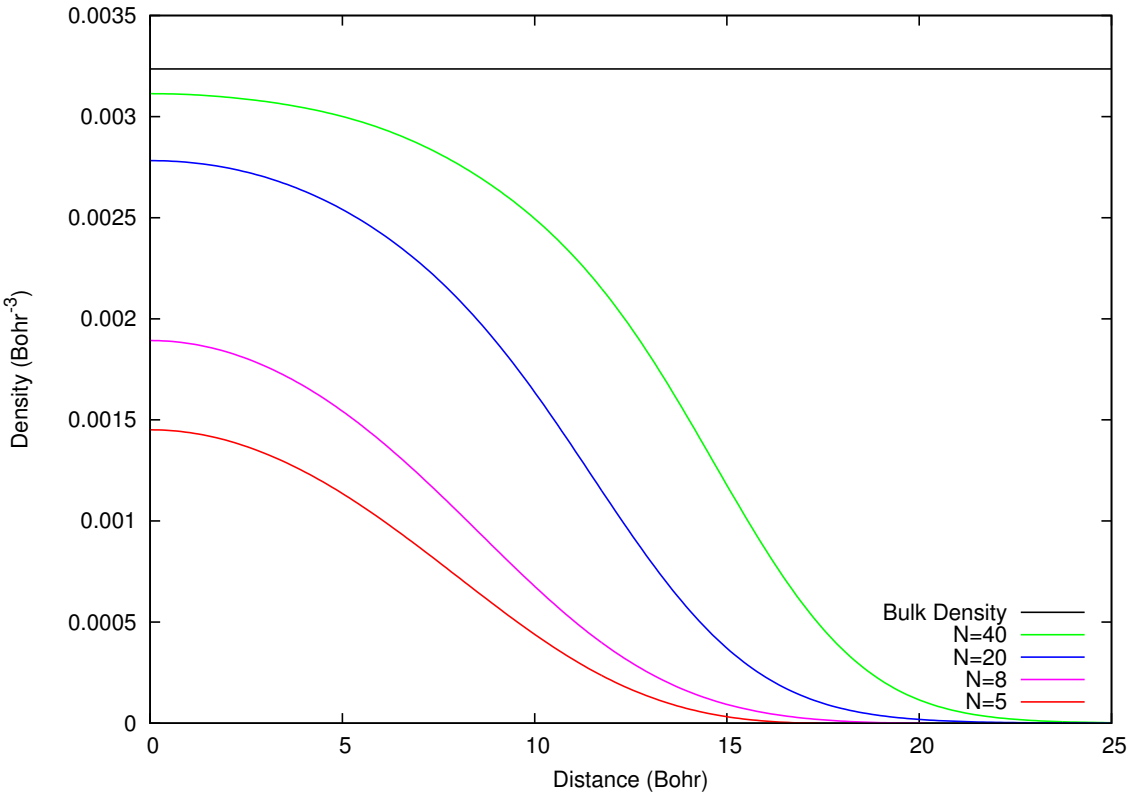


Figure 4.1: Density profiles for select small droplets ranging in size from N=5 to N=40 atoms calculated using the Orsay-Paris functional. Note that even the largest droplet shown here still never achieves bulk density (represented by the horizontal black line) at any point.

Monte Carlo data (Mella et al.[7]) and because we have previously modelled larger silver-containing droplets successfully (see Chapter 3). The Ag-He potential used by Mella et al. in their DMC studies is older than the one described in the previous chapter, and has the form shown in Eq. 4.2[7]. The constants A (3,870,264.2 cm^{-1}) and B (2.7637976 $Å^{-1}$) of Eq. 4.2 govern the close-range repulsive wall, while C (12,443,020 $Å^{10}cm^{-1}$), D (2,491,338 $Å^{8}cm^{-1}$), and E (67,213.355 $Å^{6}cm^{-1}$) dictate the long-range attraction, and originate from a fitting to the MP2 calculations of Jakubek and Takami[77].

$$V_{He-Ag} = Ae^{-Br} + \frac{C}{r^6} + \frac{D}{r^8} + \frac{E}{r^{10}} \tag{4.2}$$

Guided by the results of Table 4.1, we perform simulations of silver-doped helium clusters using the form of the potential described in Eq. 4.2 with N = 20, 25, 30, and 40. The binding energies of the doped clusters are then calculated as in Chapter 3 and displayed in Table 4.3, along with total cluster energies and the DMC values.

The agreement of the He-DFT results with the available DMC cluster energies is similar to that of the pure case, indicating the presence of the silver atom has little impact on the overall accuracy of our calculations. Again, the largest contributor to the observed discrepancies is likely the difference in He-He potential function used, although application of the corrective scaling procedure used to account for this in the pure clusters case is now less straightforward with a dopant present. This is because the rescaling procedure inherently scales both the kinetic and potential energies within the cluster; the error introduced by scaling the kinetic energy is small relative to the gain in accuracy of scaling the potential energy for pure clusters, and so the procedure is tolerable. In the doped clusters, however, the kinetic energy increases due to localization of the helium into solvation shells, meaning the error introduced by incorrectly scaling it is amplified. An extra term describing the He-dopant interaction energy is also present in the total energy of doped clusters, which would be incorrectly scaled if the scaling factor was applied to the cluster energies reported. We can instead

Table 4.3: Binding energies for small He_N clusters doped with a single central Ag atom, calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals and an imaginary time step $\Delta\tau = 2500$ a.u. Benchmarking values are DMC values by Mella et al[7]. Parenthetical quantities indicate percent error with respect to the DMC calculations.

N	E_{OP} (K)	E_{Cat} (K)	E_{OT} (K)	E_{DMC} (K)	$E_{bind,\ OP}$ (K)	$E_{bind,\ Cat}$ (K)	$E_{bind,\ OT}$ (K)	$E_{bind,\ DMC}$ (K)
20	-120.333 (10.2%)	-107.942 (19.5%)	-125.058 (6.7%)	-134.06	90.664 (10.2%)	80.441 (20.3%)	96.317 (4.6%)	100.91
25	-145.926 (8.1%)	-131.923 (16.9%)	-150.753 (5.0%)	-158.71	101.731 (11.3%)	90.500 (21.1%)	107.528 (6.3%)	114.7
30	-170.237 (6.2%)	-155.300 (14.4%)	-174.271 (4.0%)	-181.455	110.159 (8.9%)	98.557 (18.5%)	115.205 (4.7%)	120.94
40	-217.423 (4.8%)	-201.194 (11.9%)	-218.690 (4.2%)	-228.344	122.548 (4.1%)	110.686 (13.4%)	124.922 (2.2%)	127.8

decompose the reported He-DFT energies into their components, as described in Chapter 3, scale only the helium-helium terms, and retotal, thus accounting for the difference in He-He potentials in a more appropriate manner.

Using this approach, the new N=20 cluster energies are -121.210 K (OP), -108.437 K (Cat), and -125.916 K (OT), while the new N=40 energies are -222.714 K (OP), -205.748 K (Cat), and -224.482 K (OT). The greater change observed for the N=40 clusters is due to the increased helium density - the interactions between helium atoms constitute a greater component of the total energy when more are present. A more subtle implication evident in the N=20 data is that the Ag-He interactions are also not being completely captured by the density functional method, as the corrective procedure for the He-He components did not much alter the total energies. This is partially because the silver atom is assumed static in the center of the cluster in He-DFT, whereas in the DMC simulations - and in reality - it has a finite kinetic energy that goes unaccounted for in our calculations. Despite these shortcomings, the binding energies calculated using the Orsay-Paris and Orsay-Trento functionals are actually quite good, particularly when using the adjusted total energies.

To conclude our studies of small droplets, we investigate the differences in energies predicted by He-DFT when using the more modern Ag-He diatomic potential of Eq. 3.39 in place of Eq. 4.2. The shape of this potential can be seen in Fig. 4.2, along with the old Ag-He potential of Mella et al. and the He-He Lennard-Jones potential for reference. Because it has an attractive well much lower than that of the previous potential, the new Ag-He potential is expected to produce much lower binding energies than those seen in Table 4.3.

The results of our investigation into small silver-doped clusters using the modern Ag-He pair potential can be seen in Table 4.4. Because of the shallowness of the Ag-He potential well, the binding energies determined for small droplets are 2-3 times smaller than with the older Ag-He potential, although there is no available Monte Carlo data to explicity monitor the accuracy of these calculations. Regardless, it is evident that

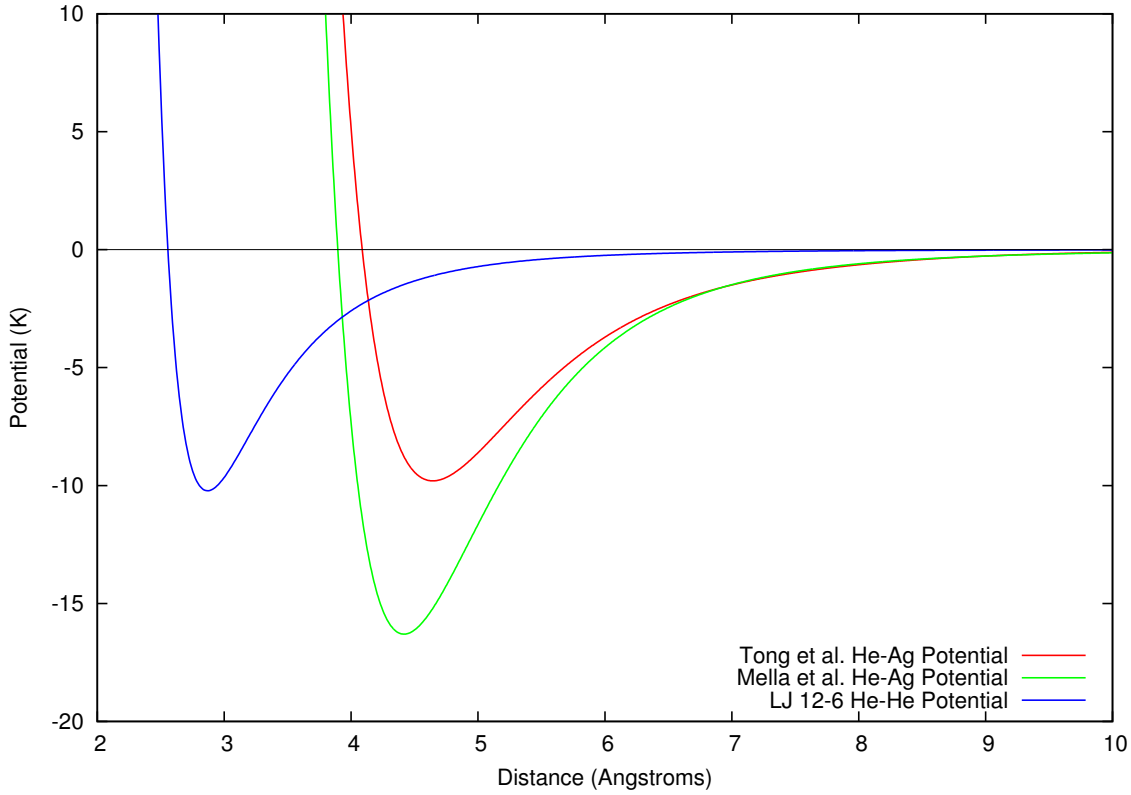


Figure 4.2: Comparison of Ag-He potentials used to model small silver-doped helium clusters. Note that the potential of Tong et al. predicts an attractive well similar in magnitude to that of the He-He Lennard-Jones well.

Table 4.4: Binding energies for small He_N clusters doped with a single central Ag atom described by a modern potential fitted to ab initio data, calculated using the Orsay-Paris, Catalonia, and Orsay-Trento functionals and an imaginary time step $\Delta\tau = 2500$ a.u.

N	E_{OP} (K)	E_{Cat} (K)	E_{OT} (K)	$E_{bind,\ OP}$ (K)	$E_{bind,\ Cat}$ (K)	$E_{bind,\ OT}$ (K)
20	-70.600	-63.603	-70.843	40.931	36.102	42.102
25	-90.851	-82.193	-92.447	46.656	40.770	49.222
30	-111.454	-101.507	-113.426	51.376	44.764	54.360
40	-153.629	-141.749	-154.447	58.754	51.241	60.679

the energetics of small doped droplets can be more sensitive to the choice of helium-dopant pair potential than the calculation method used, which in turn suggests that improving this potential (if applicable) could yield a greater increase in accuracy than abandoning He-DFT for a more computationally powerful tool, even in droplet size regimes that are below bulk helium density.

4.2 Super-bulk Densities

The second type of 'extreme' helium system we investigate is one with a region of very high density - the opposite of the cases seen in Section 4.1. Assigned the label of 'super-bulk' densities, these regions represent solvation layers that have accumulated enough atoms to have densities appropriate to a helium solid, although not necessarily with any particular crystal structure. An excellent example of these types of systems seen experimentally are the 'snowballs' present in helium droplets solvating ionic atoms. As alluded to in Section 1.3, these 'snowballs' are solid clusters of helium atoms within a droplet bound by a particularly attractive ionic dopant and disjointed from the rest of the superfluid component. Because 'snowball'-containing systems are intrinsically spherically symmetric in the cases of singularly doped atomic ions, they represent a convenient avenue of study with our density functional method.

The ionic dopants we choose to investigate can be separated into two categories: alkali cations (Li^+, Na^+, and K^+), and halide anions (F^- and Cl^-). The form of the diatomic potential for both sets of ions can be seen in Eq. 4.3, where Ak = Li, Na, K denotes the former group, and Hl = F, Cl indicates the latter. In both cases, the ion-He potential originates from work done by Ahlrichs et al.[78]

$$V_{Ak^+,Hl^- - He}(r) = Ae^{-br} - \sum_{n=2}^{5} \frac{f_{2n}(r) D_{2n}}{r^{2n}} \tag{4.3}$$

Eq. 4.3 looks very similar to previous He-dopant interactions, although the terms D_{2n} include helium polarizabilities through hexadecapole as well as the He-ion

dispersion coefficients C_{2n}. The function $f_{2n}(r)$ is identical to the function found in Eq. 3.37, but with a different value for the parameter b. As a result of the electrostatic interactions of the cation-helium pair, the well depths of Eq. 4.3 for the alkali cations are all more than ten times deeper than the well for the deepest neutral dopant, xenon. By comparison, the anion-helium well depths are approximately eight (fluoride) and two (chloride) times deeper than the xenon-helium well.

We perform simulations of ion-doped droplets ranging in size from N=50-5000 atoms using the Orsay-Paris and Catalonia functionals and an imaginary time step $\Delta\tau$= 100 a.u. The value of this parameter is much smaller than in the neutral dopant studies, as the corresponding density gradients in the first solvation shells are much steeper, and thus more sensitive to larger steps between iterations. Example density profiles for the converged 1000-atom ion-droplet systems can be seen in Figs. 4.3 and 4.4 for the cationic and anionic dopants, respectively. Table 4.5 shows our calculated results for the location (R_1) and number of atoms (N_1) associated with the first solvation peak within the ion-doped droplets, as well as the asymptotic binding energies for the dopant.

The density functional method with the Orsay-Paris description of He-He interactions predicts a much larger binding energy for the cationic dopants than is seen in the variational Monte Carlo studies [8]. This in turn explains the larger number of helium atoms associated with the snowball layers as predicted by He-DFT, although it is interesting to note that the calculated size of the snowballs (i.e. the distance from the ion to the first helium solvation layer) are generally in agreement with the PIGS [11] and PIMC [10] results. The doped Li^+ scenario is particularly interesting, as the VMC simulations do not predict a well-defined snowball around the ion due to frequent atomic exchanges of helium atoms in the first solvation layer with the rest of the system. This is in contradiction with the later PIGS study [11], which predicts a strongly bound first shell of helium in Li^+-He_{70} droplets. Our He-DFT results seem to support the latter, although the overpredicted number of atoms in the snowball may suggest that this conclusion should be taken with caution.

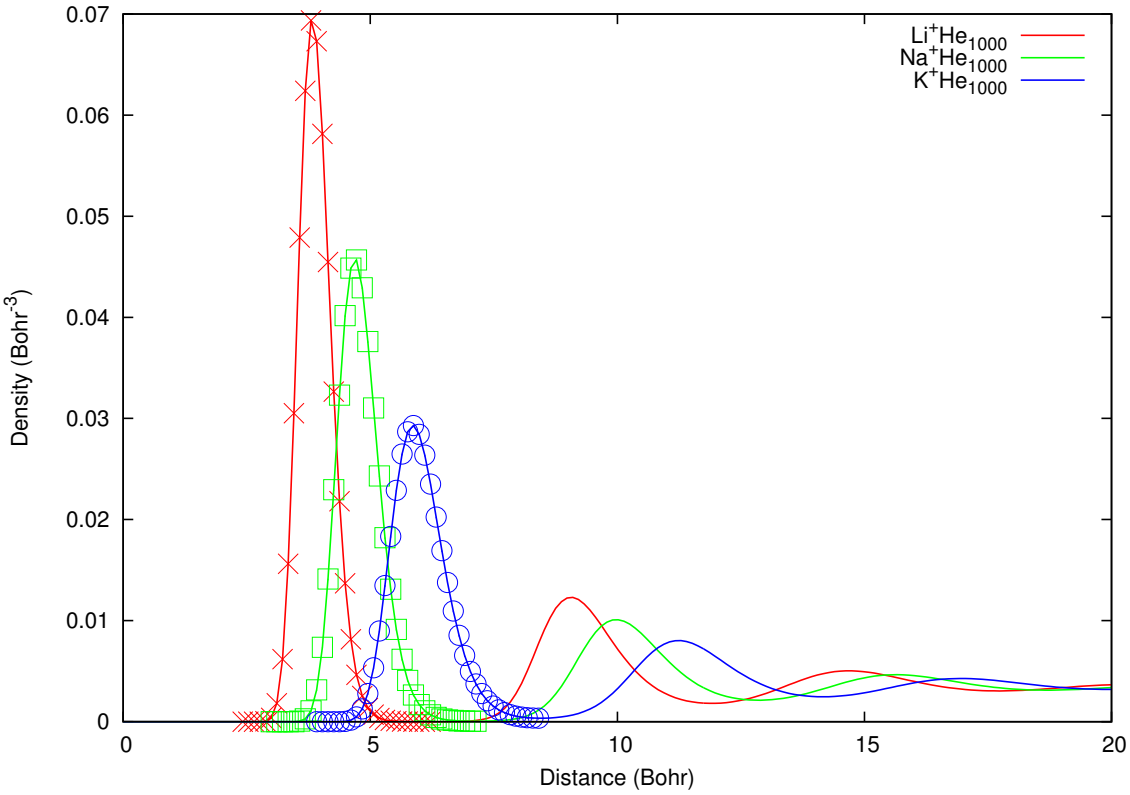


Figure 4.3: Density profiles for the cation-containing helium droplets with N=1000 atoms. The dotted points indicate the 'snowball' solvation peak.

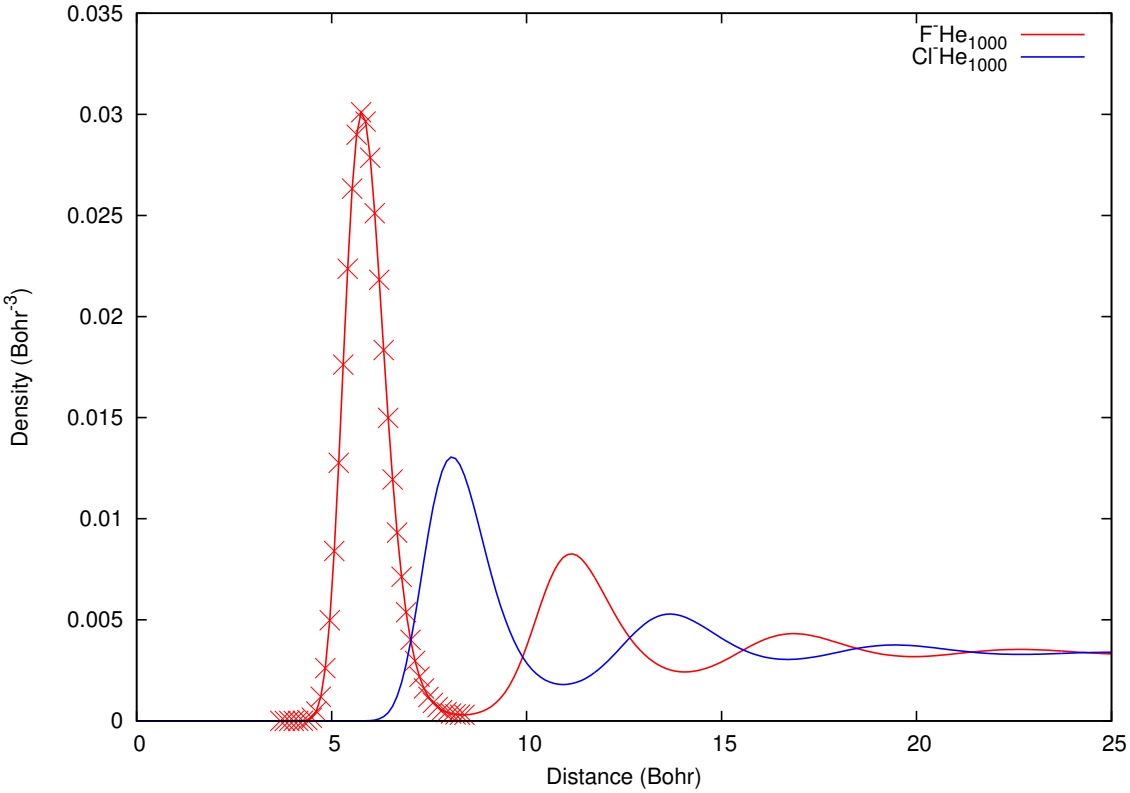


Figure 4.4: Density profiles for the anion-containing helium droplets with N=1000 atoms. The dotted points indicate the 'snowball' solvation peak. Note that the chloride ion does not induce a high-density first solvation shell similar to the other ions studied; we interpret this as a lack of a compact, solid-like snowball in the center of the droplet.

Table 4.5: Upper Half: Calculated structural properties of helium droplets solvating a single ionic dopant for an N=5000 atom droplet and a time step $\Delta\tau = 100$ a.u. **Lower Half:** Literature values for the same quantities. VMC data is from reference [8], DMC data is from reference [9], PIMC data is from reference [10], PIGS data is from reference [11], and experimental data is from reference [12]. Quantities with an asterisk were visually approximated from graphical He radial density distribution functions with an approximate accuracy of ±0.1. Quantities with a double asterisk are from size N=128 atom droplets at T=1 K from reference [10].

Orsay-Paris	Li^+	Na^+	K^+	F^-	Cl^-
R_1 (Å)	2.01	2.50	3.11	3.05	4.27
N_1	10.91	13.60	17.66	17.63	25.54
E_{bind} (K)	7550.1	4835.7	3191.0	3470.4	1666.3
Catalonia	Li^+	Na^+	K^+	F^-	Cl^-
R_1 (Å)	2.01	2.50	3.11	3.05	4.27
N_1	10.81	13.40	17.26	17.22	24.42
E_{bind} (K)	7440.7	4740.0	3111.5	3387.0	1625.4
R_1 (Å)	2.52 (VMC) 2.1±0.1* (PIGS)	2.70 (VMC) 2.57** (PIMC)	2.98 (VMC) 3.14** (PIMC)	3.4±0.1* (DMC)	4.3±0.1* (DMC)
N_1	8 (VMC) 8.24 (PIGS)	10 (VMC) 12.02 (PIGS) 12.0±0.01** (PIMC)	12 (VMC) 15.00±0.05** (PIMC)	18.3±0.9 (Exp.)	19.5±0.2 (Exp.)
E_{bind} (K)	4245.3±0.8 (VMC)	2936.6±0.2 (VMC)	2588.6±0.2 (VMC)		

In the case of anionic dopants, the He-DFT method correctly predicts the formation of larger cavities surrounding the ion ('bubbles', instead of 'snowballs') when comparing isoelectronic pairs, although the size of the chloride-induced bubble is in better agreement with the Monte Carlo data than the fluoride-induced one [9]. This is at odds with the predicted number of helium atoms in the first solvation shell; the fluoride calculations agree with experimental studies[12], while the chloride calculations predict an extra six atoms relative to experiment.

This discrepancy may be explained by noting that the density functional method predicts a more energetically stable droplet system than do the DMC calculations. This is especially evident in the case of the fluoride dopant, where the difference in calculated energies is 1435 K for an N=30 atom system and 1478 K for an N=40 He atom system (approximately double the DMC value!). The chloride dopant energies are in much better agreement across methods; the energetic difference for an N=30 atom system is 127 K and 139 K in the N=40 case, representing slightly more than 10% of the total energy in both instances. The chemical potentials calculated in the DMC studies are visually approximated to be -14 K for both the N=30 and N=40 droplets with fluorine dopants; these values compare favorably to the He-DFT results of -18.41 K and -15.68 K, respectively, particularly when considering the smallness of the droplets studied (see Sec. 4.1). The chloride-doped droplets are even better described in this respect - we calculate chemical potentials of -13.26 K and -11.47 K for the N=30,40 droplets, in comparison to the visually approximated value of -12 K from the DMC study.

To conclude this section, we note that the He-DFT method can be accurate in modeling systems containing regions where the helium density is far above bulk value; however, it suffers with issues of consistency in many cases. For example, the structural properties predicted by He-DFT seem to be in agreement with external sources for either the location of the first solvation layer (the size of the snowball or bubble) or the number of helium atoms it contains, but not both. The same is true for the calculated droplet energetics - the total droplet energies and dopant binding

energies are uniformly much larger in magnitude relative to Monte Carlo studies, but the chemical potential values seem to be in good agreement with the (somewhat limited) available data.

Chapter 5

Conclusions

5.1 Conclusions

Throughout the course of this document, we have presented a thorough analysis of the He-DFT method in the context of planar surfaces and droplets with and without atomic dopants. Furthermore, we test the ability of He-DFT to accurately model helium systems far from the bulk description in both the low- and high-density limits with mixed results. In performing these analyses, we provide an assessment of the accuracy and computational expense of four of the most popular helium functionals of the past three decades - a survey that is, to our knowledge, not well-represented in the literature. Because the process of choosing a functional is a central part of any DFT calculation, we expect that the inclusion of all three non-local helium functionals in our calculations will be useful in guiding future He-DFT simulations of new and exciting systems.

In Chapter 2, we showed that the zero-range helium functional of Stringari and Treiner is adequate in describing homogeneous planar He-4 slabs, but its omission of He-He correlations results in a failure to capture the more complex structural features present in artificial Lennard-Jones attractive surfaces. Because the infinitely smooth one-dimensional attractive surface is one of the simplest inhomogeneous systems

studied, we conclude that the Stringari-Treiner functional will be unable to accurately model other more complex helium systems, and thus abandon it on this basis. The Orsay-Paris functional is found to be successful in describing the solvation layers expected from the LJ surfaces, although the exact values are meaningless because of their artificial nature. To amend this, the structural properties of graphite are computed for a 1-D surface with an appropriate potential, and found to be in good agreement with the neutron scattering data of Carneiro et al. [13].

The graphite system is then used as a benchmark to compare two distinct computational algorithms, the 'mixing' and ITP methods. We show that these methods are equally capable of predicting the location and corresponding per-particle surface area of the first solvation layer, but the calculated chemical potential and overall convergence are much better in the ITP formalism due to the helium 'sloshing' seen in the mixing method. Finally, we make an attempt to model pure helium droplets in a fully three-dimensional grid, but find that small numerical inaccuracies and high computational cost render this system description of little use in its current state.

The one-dimensional droplet calculations of Chapter 3 see the addition of the other two non-local helium functionals, namely, the Catalonia and Orsay-Trento variants. Using these functionals, along with the Orsay-Paris functional from Chapter 2 and the ITP method, we compute energetic properties of droplets ranging in size from 50-5000 helium atoms with the intent of generating an accurate fitting function for the droplet energies and chemical potentials. Introduction of a single atomic dopant into these droplets then yields information as to its binding energy. The simulations of doped droplets also reveal that the chemical potential of the system is insensitive to the identity of the dopant for large (N>500) droplet sizes, but is still significantly different than the bulk chemical potential, even for large droplets where there is a significant central bulk density region. This is attributed to the presence of destabilizing effects intrinsic to the helium surface of the droplets. From the fitted chemical potential equation and the calculated binding energies of the dopants, we can create accurate

lookup tables describing initial and final distributions of droplet sizes representative of a variety of observed experimental helium droplet calorimetry data, which in turn will be useful in guiding the intelligent design and interpretation of those experiments.

Finally, the ability of He-DFT to describe syatems that deviate greatly from bulk behavior was assessed in Chapter 4. We find that the accuracy of the He-DFT method begins to break down for pure droplets smaller than N=20 atoms from benchmarking to DMC data, although subsequent calculations involving two types of silver dopant reveal that the choice of He-dopant pair potential can more greatly impact the accuracy of the results than the choice of computational method. The ability of He-DFT to describe super-bulk helium phenomena is frustratingly inconsistent; our calculations on ion-doped systems can only accurately describe a subset of the structural and energetic properties at any given time. This indicates that the He-DFT method can be used in such cases, but rigorous functional testing should be done beforehand to assess the possible shortcomings of the calculations.

5.2 Outlook and Future Directions

The immediate future of this project is in modeling droplets with small molecular (i.e. non-spherical) dopants, so as to assess the capabilities of He-DFT in the context of very inhomogeneous systems. These calculations will also extend the database of theoretically determined binding energies, which can then be used to help guide and interpret experimental observations from droplet calorimetry techniques. A solid groundwork has already been established towards this goal in the code describing the three-dimensional droplets; switching from Cartesian to spherical coordinates could resolve the numerical inadequacies that currently hinder this approach, although the computational demand would still be high. Alternatively, the three-dimensional Cartesian description is already conducive to modeling helium slabs adsorbed onto more complex attractive surfaces, meaning that future calculations could observe the effects of surface corrugation on the local liquid helium structure. In the very long

run, it is possible to envision a combining of the droplet and surface systems, resulting in a time-dependent He-DFT approach that could model the deposition process of captured dopants onto realistic surfaces. The applicability of He-DFT to this diversity of systems offers an exciting theoretical tool to bridge the gap between the systems currently appropriate to theoretical methods and experimental observations.

Bibliography

[1] M. Barranco and E. Hernandez, *Physical Review B* **49**, 78 (1994). x, xvii, 11, 42, 43, 44, 46, 49, 69, 72

[2] J. Dupont-Roc, M. Himbert, N. Pavloff, and J. Treiner, *Journal of Low Temperature Physics* **81**, 31 (1990). x, 11, 15, 16, 46, 47, 49

[3] F. Dalfovo, A. Lastri, L. Pricaupenko, S. Stringari, and J. Treiner, *Physical Review B* **52**, 1193 (1995). x, 11, 46, 48, 49

[4] S. Gatica, E. Hernandez, and M. Barranco, *The Journal of Chemical Physics* **107**, 927 (1997). xi, 77, 79

[5] M. Lewerenz, B. Schilling, and J. Toennies, *The Journal of Chemical Physics* **102**, 8191 (1995). xii, 95, 96, 97, 106

[6] W. Lewis, B. Haruff-Miller, P. Leatherman, M. Gord, and C. Bunker, *Review of Scientific Instruments* **85**, 094102 (2014). xii, 95, 96, 97

[7] M. Mella, M. Colombo, and G. Morosi, *The Journal of Chemical Physics* **117**, 9695 (2002), doi: 10.1063/1.1518472. xiii, 108, 109, 111, 113, 114

[8] M. Buzzacchi, D. Galli, and L. Reatto, *Physical Review B* **64**, 094512 (2001), doi: 10.1103/PhysRevB.64.094512. xiv, 119, 120

[9] E. Coccia, F. Marinetti, E. Bodo, and F. Gianturco, *ChemPhysChem* **9**, 1323 (2008), doi: 10.1002/cphc.200800132. xiv, 120, 123

[10] D. Galli, D. Ceperley, and L. Reatto, *The Journal of Physical Chemistry A* **115**, 7300 (2011). xiv, 119, 120

[11] S. Paolini, F. Ancilotto, and F. Toigo, *The Journal of Chemical Physics* **126**, 124317 (2007), doi: 10.1063/1.2711813. xiv, 119, 120

[12] F. da Silva, P. Waldburger, S. Jaksch, A. Mauracher, S. Denifl, O. Echt, T. Märk, and P. Scheier, *Chemistry - A European Journal* **15**, 7101 (2009), doi: 10.1002/chem.200802554. xiv, 120, 123

[13] K. Carneiro, L. Passell, W. Thomlinson, and H. Taub, *Physical Review B* **24**, 1170 (1981). xvi, 30, 31, 126

[14] P. Kapitza, *Nature* **141**, 74 (1938). 3

[15] J. Allen and A. Misener, *Nature* **141**, 75 (1938). 3

[16] F. London, *Nature* **141**, 643 (1938). 3

[17] S. Bose, *Zeitschrift für Physik* **26**, 178 (1924), doi:10.1007/BF01327326. 3

[18] A. Einstein, *Sitzungsberichte der Preussischen Akademie der Wissenschaften* **1**, 3 (1925). 3

[19] V. Bagnato, D. Frantzeskakis, P. Kevrekidis, B. Malomed, and D. Mihalach, *Romanian Reports in Physics* **67**, 5 (2015). 3

[20] A. Leggett, *Science* **319**, 1203 (2008). 3

[21] L. Landau, *Physical Review* **60**, 356 (1941). 3, 4

[22] V. Peshkov, *Soviet Physics JETP* **11**, 580 (1960). 3

[23] R. Donnelly, *Physics Today* , 34 (2009). 3, 4

[24] W. Vinen, The physics of superfluid helium, School of Physics and Astronomy, University of Birmingham, Birmingham B15 2TT, UK, 2004. 3

[25] J. Brooks and R. Donnelly, *Journal of Physical and Chemical Reference Data* **6**, 51 (1977). 4

[26] R. Feynman and M. Cohen, *The Physical Review* **102**, 1189 (1956). 4

[27] V. Kruglov and M. Collett, *Physical Review Letters* **87**, 185302 (2001). 4

[28] G. Rayfield and F. Reif, *Physical Review* **136**, A1194 (1964). 4

[29] C. Barenghi, R. Donnelly, and W. Vinen, *Journal of Low Temperature Physics* **52**, 189 (1983). 4

[30] T. Buttke, Turbulence and Vortices in Superfluid Helium, in *Vortex Methods and Vortex Motion*, edited by K. Gustafson and J. Sethian, chapter 7, pp. 171–194, Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, Pennsylvania 19104-2688, 1991. 4

[31] G. Bewley, D. Lathrop, and K. Sreenivasan, *Nature* **441**, 588 (2006). 4

[32] L. Gomez, E. Loginov, and A. Vilesov, *Physical Review Letters* **108**, 155302 (2012), doi: 10.1103/PhysRevLett.108.155302. 4

[33] J. Toennies, *Molecular Physics* **111**, 1879 (2013), doi: 10.1080/00268976.2013.802039. 5, 6, 45

[34] J. Harms, J. Toennies, and F. Dalfovo, *Physical Review B* **58**, 3341 (1998). 5, 92, 93

[35] D. Brink and S. Stringari, *Zeitschrift für Physik D - Atoms, Molecules and Clusters* **15**, 257 (1990). 5

[36] S. Grebenev, J. Toennies, and A. Vilesov, *Science* **279**, 2083 (1998). 5

[37] W. Lewis, B. Harruff-Miller, M. Gord, J. Gord, E. Guliants, and C. Bunker, *Review of Scientific Instruments* **83**, 073109 (2012), doi: 10.1063/1.4738664. 6

[38] S. Krasnokutski, M. Kuhn, M. Renzler, C. Jäger, T. Henning, and P. Scheier, *The Astrophysical Journal Letters* **818** (2016). 6

[39] E. Lugovoj, J. Toennies, and A. Vilesov, *The Journal of Chemical Physics* **112**, 8217 (2000), doi: 10.1063/1.481426. 6

[40] S. Krasnokutski and F. Huisken, *The Journal of Physical Chemistry A* **114**, 7292 (2010), doi: 10.1021/jp103947z. 6

[41] T. Döppner, T. Diederich, S. Göde, A. Przystawik, J. Tiggesbäumker, and K.-H. Meiwes-Broer, *The Journal of Chemical Physics* **126**, 244513 (2007), doi: 10.1063/1.2745294. 6

[42] J. Poms, A. Hauser, and W. Ernst, *Physical Chemistry Chemical Physics* **14**, 15158 (2012), doi: 10.1039/c2cp42333b. 6

[43] D. Jin, W. Guo, W. Wei, and H. Maris, *Journal of Low Temperature Physics* **158**, 307 (2010), doi: 10.1007/s10909-009-9935-6. 6

[44] J. Toennies and A. Vilesov, *Annual Review of Physical Chemistry* **49**, 1 (1998). 6, 45

[45] F. Stienkemeier and K. Lehmann, *Journal of Physics B: Atomic, Molecular and Optical Physics* **39**, R127 (2006), doi: 10.1088/0953-4075/39/8/R01. 6, 45

[46] W. McMillan, *Physical Review* **138**, A442 (1965). 6, 7

[47] R. Aziz, V. Nain, J. Carley, W. Taylor, and G. McConville, *The Journal of Chemical Physics* **70**, 4330 (1979), doi: 10.1063/1.438007. 7

[48] R. Aziz, F. McCourt, and C. Wong, *Molecular Physics* **61**, 1487 (1987). 7, 108

[49] K. Schmidt, M. Kalos, M. Lee, and G. Chester, *Physical Review Letters* **45**, 573 (1980). 7, 9

[50] D. Ceperley and B. Alder, *Science, New Series* **231**, 555 (1986). 8

[51] J. Boronat, Monte Carlo Simulations at Zero Temperature: Helium in One, Two, and Three Dimensions, in *Microscopic Approaches to Quantum Liquids in Confined Geometries*, edited by E. Krotscheck and J. Navarro, chapter 2, pp. 21–90, World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 2002. 8

[52] D. Ceperley, *Reviews of Modern Physics* **67**, 279 (1995). 8

[53] A. Sarsa, K. Schmidt, and W. Magro, *The Journal of Chemical Physics* **113**, 1366 (2000), doi: 10.1063/1.481926. 8, 9

[54] R. Hinde, *Computer Physics Communications* **182**, 2339 (2011). 8

[55] M. Rossi, M. Nava, L. Reatto, and D. Galli, *The Journal of Chemical Physics* **131**, 154108 (2009), doi: 10.1063/1.3247833. 9

[56] A. Barnes and R. Hinde, *The Journal of Chemical Physics* **146**, 094510 (2017), doi: 10.1063/1.4977525. 9

[57] P. Hohenberg and W. Kohn, *Physical Review* **136**, B864 (1964). 9, 45

[58] S. Stringari and J. Treiner, *Physical Review B* **36**, 8369 (1987). 11, 15, 71

[59] S. Stringari and J. Treiner, *The Journal of Chemical Physics* **87**, 5021 (1987). 11, 69, 72, 110

[60] J. de Boer and A. Michels, *Physica* **5**, 945 (1938). 16, 47

[61] L. Szybisz, *The European Physical Journal B* **14**, 733 (2000). 16

[62] L. Lehtovaara, J. Toivanen, and J. Eloranta, *Journal of Computational Physics* **221**, 148 (2007). 19, 21, 59

[63] P. Swarztrauber, Vectorizing the FFTs, in *Parallel Computations*, edited by G. Rodrigue, pp. 51–83, Academic Press, 1982, ISBN 978-0-12-592101-5. 22, 41

[64] W. Carlos and M. Cole, *Surface Science Letters* **91**, 339 (1980). 27

[65] R. Bisseling and R. Kosloff, *Journal of Computational Physics* **59**, 136 (1985). 64

[66] V. Serov, *Computer Physics Communications* **in press** (2017), http://doi.org/10.1016/j.cpc.2017.03.001. 65

[67] M. Iino, M. Suzuki, and A. Ikushima, *Journal of Low Temperature Physics* **61**, 155 (1985). 71

[68] J. Lopez Cacheiro, B. Fernndez, D. Marchesan, S. Coriani, C. Hättig, and A. Rizzo, *Molecular Physics* **102**, 101 (2004). 76, 77

[69] A. Cvetko, A. Morgante, F. Tommasini, P. Cortona, and M. Dondi, *The Journal of Chemical Physics* **100**, 2052 (1994), doi: 10.1063/1.466505. 76

[70] F. Dalfovo, *Zeitschrift fr Physik D* **29**, 61 (1994). 77, 79

[71] J. Poms, A. Hauser, and W. Ernst, *Physical Chemistry Chemical Physics* **14**, 15158 (2012). 79

[72] K. T. Tang and J. P. Toennies, *Zeitschrift fr Physik D* **1**, 91 (1986). 80

[73] X. Tong, C. Yang, Y. An, M. Wang, X. Ma, and D. Wang, *Journal of Chemical Physics* **131**, 244304 (2009). 85

[74] D. Bressanini, M. Zavaglia, M. Massimo, and G. Morosi, *The Journal of Chemical Physics* **112**, 717 (2000), doi: 10.1063/1.480604. 108, 109, 111

[75] K. Tang, J. Toennies, and C. Yiu, *Physical Review Letters* **74**, 1546 (1995). 108

[76] T. Korona, H. Williams, R. Bukowski, B. Jeziorski, and K. Szalewicz, *Journal of Chemical Physics* **106**, 5109 (1997). 108

[77] Z. Jakubek and M. Takami, *Chemical Physics Letters* **265**, 653 (1997). 113

[78] R. Ahlrichs, H. Böhm, S. Brode, K. Tang, and J. Toennies, *The Journal of Chemical Physics* **88**, 6290 (1988), doi: 10.1063/1.454467. 117

Appendix

Appendix A

3-D Droplet He-DFT Code

The following is the code used in calculating the properties of 3-D helium droplets as described in Chapter 2. This code is also transferrable to systems of helium slabs against attractive surfaces with minor modification. It depends upon a user-generated input file called "input_He".

A.1 Main Program and Functionals

```
PROGRAM itprep
implicit real*8 (a-h, o-z)

parameter (pi=4.0d0*atan(1.0d0))
!ihandle is a 4-digit integer used to specify output files. This integer appears
!at the end of each output file generated by running this program.

include 'mpif.h'
!rho=density matrix, u=meanfield matrix for an individual processor, psi=wavefunction matrix
real*8, allocatable :: rho(:,:,:), u(:,:,:), psi(:,:,:), newpsi(:,:,:)
!rhonew=density matrix returned after one step of itp, is set to rho at the end
!of each step
!u_new=total meanfield matrix passed to the itp component of the calculation
real*8, allocatable :: rhonew(:,:,:), u_new(:,:,:), temp_rhobar(:,:,:), delsqrho(:,:,:)
!The x, y, and z direction matrices
real*8, allocatable :: x(:), y(:), z(:)
!The momentum-space x, y, and z matrices
real*8, allocatable :: p_x(:), p_y(:), p_z(:)
!A matrix required by the dfftpack library for the calculation of the FFT
real*8, allocatable :: wsave(:,:)
real*8 :: maxdiff
integer :: rank
!A 1x3 time matrix containing hour, minute, and second data in each column,
!respectively
integer, dimension(3) :: time_old
!The character equivalent of 'ihandle', for use in naming files
character(len=4) :: handle
logical :: iscale

!Initialize MPI
call MPI_INIT (ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, nproc, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)

!itime is an intel-specific subroutine that returns hour, minute, and second
!data to a 1x3 matrix (time_old)
```

```
call itime(time_old)

open (1, file='input_He')
read(1,*) ihandle
read(1,*) nx_per_h
read(1,*) ny_per_h
read(1,*) nz_per_h
read(1,*) cutoff_lj
read(1,*) rmid
read(1,*) iter
!read() functional
read(1,*) dt
close (1)

!Write the four-digit handle to the batch file.
if (rank.eq.0) then
 write(*,*) "tag = ", ihandle
end if

!Redefine ihandle as a character called 'handle' of length 4 for use in naming files
write(handle,1000) ihandle
1000 format(I4.4)

!iscale controls whether or not the starting density is scaled to the nearest
!integer (rounded up). It is automatically set to true, but will not scale a
!density due to a restarted/checkpointed job.
iscale=.true.

!h is the cutoff radius in the Orsay-Paris and Orsay-Trento DFT schemes
h=2.376728d0*1.889725989d0

!Define the number of points to be sampled in the x, y, and z directions within
!the cutoff radius h. This includes a point at the very center (radius=0)

!Define the spacings in the x, y, and z directions by the number of points
!desired per length of h. The division is one less than the number of points per
!h because there are n-1 spaces between n points.
dx=h/(nx_per_h-1)
dy=h/(ny_per_h-1)
dz=h/(nz_per_h-1)

!Define the number of points necessary for the simulation.
nmax=ceiling(4.0*rmid/dx+1)

if (mod(nmax,2).eq.1) then
 nmax=nmax+1
end if

!h-bar is equal to 1 in atomic units
hbar=1.0d0

!Initialize the x, y, and z direction arrays
x_init = 0.0d0
y_init = 0.0d0
z_init = 0.0d0

!matcube will be used in MPI commands, it is the total number of elements in a
!cubic box sampled nmax times in each direction
matcube=nmax*nmax*nmax

!Create the x, y, and z direction arrays. Store in the variable n_cutoff the
!sampling point at which the Lennard-Jones 12-6 potential is essentially zero
!(for use in speeding up the meanfield calculations). This cutoff was chosen
!somewhat arbitrarily and can be changed if desired.

allocate(x(nmax))
allocate(y(nmax))
allocate(z(nmax))

do n=1, nmax
 x(n) = x_init+(n-1)*dx
 if (x(n-1).lt.cutoff_lj.and.x(n).ge.cutoff_lj) then
  n_cutoff=n
 end if
 y(n) = y_init+(n-1)*dy
 z(n) = z_init+(n-1)*dz
end do

!Record some useful data regarding the system's physical dimensions.

if (rank.eq.0) then
 open (1, file='SystemData'//trim(handle))
  write(1,*) "-----------------------------------------------------"
  write(1,*) "SYSTEM DIMENSIONS"
  write(1,*) "-----------------------------------------------------"
  write(1,*) "NMAX = ", nmax
  write(1,*) "Total number of points = ", matcube
  write(1,*) "n per h (x) = ", nx_per_h
```

```
  write(1,*) "X-spacing = ", dx, "bohr"
  write(1,*) "n per h (y) = ", ny_per_h
  write(1,*) "Y-spacing = ", dy, "bohr"
  write(1,*) "n per h (z) = ", nz_per_h
  write(1,*) "Z-spacing = ", dz, "bohr"
  write(1,*) "X dimension length = ", x(nmax), "bohr"
  write(1,*) "Y dimension length = ", y(nmax), "bohr"
  write(1,*) "Z dimension length = ", z(nmax), "bohr"
  write(1,*) "LJ cutoff length = ", x(n_cutoff), "bohr"
 close (1)
end if

allocate(rho(nmax,nmax,nmax))
allocate(u(nmax,nmax,nmax))
allocate(delsqrho(nmax,nmax,nmax))
allocate(temp_rhobar(nmax,nmax,nmax))
allocate(u_new(nmax,nmax,nmax))

if (rank.eq.0) then
 allocate(psi(nmax,nmax,nmax))
 allocate(newpsi(nmax,nmax,nmax))
 allocate(wsave((2*nmax+15),(2*nmax+15)))
end if

!The first 8 processors (rank 0-7) call the initialize subroutine. This
!subroutine creates the initial density profile according to Stringari and
!Treiner's 1987 paper (Equation 26). It also checks for a saved density profile
!from previous iterations to load as an initial profile.
if (rank.lt.8) then
 call initialize(nmax, x, y, z, dx, dy, dz, rho, rmid, rank, handle, iscale)
end if

if (rank.eq.0.and.iscale.eq..true.) then
 call whole_atom_scaling(nmax, dx, dy, dz, rho)
end if

!Distribute the initial density profile (rho) to all processors from root (rank
!0)
call MPI_BCAST(rho, matcube, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD, ierr)

!Begin imaginary time propagation. Iter keeps track of the number of iterations,
!while counter is the user-declared number of iterations when prompted.

do iter=1, 100

!Call the meanfield (found below) and itp  subroutines to perform the
!calculation.

if (rank.eq.0) then
 write(*,*) "Starting iteration number", iter
end if

!All processors call the subroutine makeclean to wipe any previous values stored
!in the arrays u (meanfield) and temp_rhobar (averaged density profile). This is
!important because later on, u and temp_rhobar are recursively added across
!processors, and any non-calculated values need to be zero.

call makeclean(nmax, u, delsqrho, temp_rhobar)

!All processors call the subroutine meanfield, which calculates the mean field
!potential for the system. The output of meanfield is an array u_new storing the
!mean field potential information.

!call  mf_ST(rho, u, u_new, delsqrho, nmax, dx, dy, dz, nproc, rank, handle)
!call mf_OP(rho, u, temp_rhobar, u_new, nmax, n_cutoff, x, y, z, dx, dy, dz, nproc, rank, handle)
call mf_BH(rho, u, temp_rhobar, u_new, nmax, n_cutoff, x, y, z, dx, dy, dz, nproc, rank, handle)

!All processors call the subroutine symmetry, which copies the meanfield
!information stored in one octant of u_new to the rest of u_new. This subroutine
!must be changed if a symmetry other than spherical is being considered.
!call symmetry(u_new, nmax)

call MPI_BARRIER(MPI_COMM_WORLD, ierr)

!Only the root processor calls the subroutines itp (the imaginary time
!propagation step), update (calculates energetic values and updates the density
!profile and wavefunction), and checkpoint (saves the density profile
!periodically for restarting purposes).
if (rank.eq.0) then
call symmetry(u_new, nmax, handle)
call itp(nmax, dx, dy, dz, dt, x, y, z, rho, u_new, psi, newpsi, chempot, totrho1)
call update(nmax, z, rho, psi, newpsi, chempot, iter, maxdiff, handle)
call checkpoint(nmax, time_old, rho, handle)
end if

!Root updates the other processors with the new density profile (still stored in
!the array rho).
call MPI_BCAST(rho, matcube, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD, ierr)
```

```
if (rank.eq.0) then
 write(*,*) "Ending iteration number", iter
end if

end do

if (rank.eq.0) then
 open (1, file='SystemData'//trim(handle), access='append', status='old')
  write(1,*) "Total number of atoms = ", totrho1
  write(1,*) "-----------------------------------------------------"
  write(1,*) "ENERGETIC DATA"
  write(1,*) "-----------------------------------------------------"
  write(1,*) "Imaginary time step = ", dt
  write(1,*) "Final chemical potential = ", chempot*3.1577504d5, "Kelvin"
  write(1,*) "Maximum psi difference = ", maxdiff
  write(1,*) "-----------------------------------------------------"
 close (1)
end if

deallocate(rho)
deallocate(u)
deallocate(delsqrho)
deallocate(temp_rhobar)
deallocate(u_new)

if (rank.eq.0) then
 deallocate(psi)
 deallocate(newpsi)
end if

call MPI_FINALIZE(ierr)

END PROGRAM

!-----------------------------------------------------------------------

subroutine initialize(nmax, x, y, z, dx, dy, dz, rho, rmid, rank, handle, iscale)
implicit real*8 (a-h, o-z)

include 'mpif.h'
dimension rho(nmax,nmax,nmax), smallrho(nmax/2,nmax/2,nmax/2)
dimension x(nmax), y(nmax), z(nmax)
dimension ijk(3), ibegin_end(2)
real*8 :: maximum
integer :: istat(MPI_STATUS_SIZE), rank, recv_count
character(len=4) :: handle
logical :: iscale

!Define the initial density function from Eq. (26) of Stringari
!and Treiner.  Note that although the article says this is a good
!approximation for 3-He, the authors also use the same function
!for 4-He, only with different parameters.  The parameter values
!for 4-He are listed in the middle of the left column of p. 8374
!(just above Fig. 3).

!Define the approximate R value where the slab's density is 50%
!of the maximum density.

!Now check if we have an existing input density in the file "saveden".
!STATUS=OLD with ERR=999 means that we jump ahead to line 999 if the file does not exist.
!This allows us to use the approximate density from an earlier simulation to speed things up.

!Note that the density units are Angstrom**(-3) in the Stringari paper.
!Converted to bohr (brho):

brho=0.021836d0/1.889725989d0**3.0d0

aparam=x(nmax)/2
bparam=y(nmax)/2
cparam=z(nmax)/2

!Define the approximate R value where the slab's density is 50%
!of the maximum density.

ibegin_end(1)=1
ibegin_end(2)=nmax/2+1

if (mod(rank,2).eq.1) then
 ix=1
else
 ix=2
end if

if(mod(rank,4).eq.1.or.mod(rank,4).eq.2) then
 jx=1
else
 jx=2
```

```
end if

if (rank.ge.1.and.rank.le.4) then
 kx=1
else
 kx=2
end if

ijk(1)=ibegin_end(ix)
ijk(2)=ibegin_end(jx)
ijk(3)=ibegin_end(kx)

do i=1,nmax/2
 do j=1,nmax/2
  do k=1,nmax/2
   smallrho(i,j,k)=0.0
  end do
 end do
end do

do i=1,nmax/2
 do j=1,nmax/2
  do k=1,nmax/2
   ii=(i-1)+ibegin_end(ix)
   jj=(j-1)+ibegin_end(jx)
   kk=(k-1)+ibegin_end(kx)

   smallrho(i,j,k)=brho/((1.0d0+exp((sqrt((x(ii)-aparam)**2+(y(jj)-bparam)**2+(z(kk)-cparam)**2)-rmid)
   /(1.96d0*1.889725989d0)))**2.5d0)
  end do
 end do
end do

int_smallrho=(nmax/2)**3

if (rank.eq.0) then
 do i=1 ,nmax/2
  do j=1, nmax/2
   do k=1, nmax/2
    ii=(i-1)+ijk(1)
    jj=(j-1)+ijk(2)
    kk=(k-1)+ijk(3)

    rho(ii,jj,kk)=smallrho(i,j,k)
   end do
  end do
 end do

 do recv_count=1,7
  ibegin=1
  call MPI_SEND(ibegin, 1, MPI_INT, recv_count, 0000, MPI_COMM_WORLD, ierr)
  call MPI_RECV(ijk, 3 , MPI_DOUBLE_PRECISION, recv_count, 0001, MPI_COMM_WORLD, istat, ierr)
  call MPI_RECV(smallrho, int_smallrho, MPI_DOUBLE_PRECISION, recv_count, 0002, MPI_COMM_WORLD, istat, ierr)
  write(*,*) "Receiving rank ", recv_count
  do i=1 ,nmax/2
   do j=1, nmax/2
    do k=1, nmax/2
     ii=(i-1)+ijk(1)
     jj=(j-1)+ijk(2)
     kk=(k-1)+ijk(3)
      rho(ii,jj,kk)=smallrho(i,j,k)
     end do
    end do
   end do
 end do

else if (0.lt.rank.lt.8) then
 call MPI_RECV(ibegin, 1, MPI_INT, 0, 0000, MPI_COMM_WORLD, istat, ierr)
 call MPI_SEND(ijk, 3, MPI_DOUBLE_PRECISION, 0, 0001, MPI_COMM_WORLD, ierr)
 call MPI_SEND(smallrho, int_smallrho, MPI_DOUBLE_PRECISION, 0, 0002, MPI_COMM_WORLD, ierr)
end if

if (rank.eq.0) then

!icount=0
!do i=1, nmax*mesh, mesh
! icount=icount+1
! jcount=0
!do j=1, nmax*mesh, mesh
! jcount=jcount+1
! kcount=0
!do k=1, nmax*mesh, mesh
! kcount=kcount+1
! rho(icount,jcount,kcount)=totbigrho(i,j,k)
!end do
!end do
!end do
```

```
open(2,file='initden')
 do i=1, nmax
  write(2,2000) x(i), y(i), z(i), rho(i,nmax/2,nmax/2), rho(nmax/2,i,nmax/2),  rho(nmax/2,nmax/2,i)
 end do
close(2)

2000 format (6(1pe15.8, 1x))

!Now check if we have an existing input density in the file "saveden".
!STATUS=OLD with ERR=999 means that we jump ahead to line 999 if the file does
!not exist.
!This allows us to use the approximate density from an earlier simulation to
!speed things up.

open (999, file='save_den'//trim(handle), status='old', err=999)
do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   read (999, *) i_check, j_check, k_check, rho(i,j,k)

   if (i_check.ne.i.or.j_check.ne.j.or.k_check.ne.k) then
    write(*,*) "Error reading in saved density file"
    stop
   end if

  end do
 end do
end do
close (999)

iscale=.false.
write(*,*) "Successful save_den initialization"

999 continue

end if

return

end subroutine

!-----------------------------------------------------------------------

subroutine whole_atom_scaling(nmax, dx, dy, dz, rho)
implicit real*8 (a-h, o-z)

dimension rho(nmax,nmax,nmax)

rho_unscaled=0.0

do i=1, nmax-1
do j=1, nmax-1
do k=1, nmax-1
rho_unscaled_int=dx*dy*dz/8.0d0*(rho(i,j,k)+rho(i+1,j,k)+rho(i,j+1,k)+rho(i+1,j+1,k)+rho(i,j,k+1)
+rho(i+1,j,k+1)+rho(i,j+1,k+1)+rho(i+1,j+1,k+1))

rho_unscaled=rho_unscaled+rho_unscaled_int
end do
end do
end do

write(*,*) rho_unscaled
!c_rho_unscaled=ceiling(rho_unscaled)
scaling_factor=ceiling(rho_unscaled)/rho_unscaled

write(*,*) scaling_factor

!brho=0.021836d0/1.889725989d0**3.0d0

do i=1, nmax
do j=1, nmax
do k=1, nmax

rho(i,j,k)=scaling_factor*rho(i,j,k)

end do
end do
end do

rho_scaled=0.0

do i=1, nmax-1
do j=1, nmax-1
do k=1, nmax-1

rho_scaled_int=dx*dy*dz/8.0d0*(rho(i,j,k)+rho(i+1,j,k)+rho(i,j+1,k)+rho(i+1,j+1,k)+rho(i,j,k+1)
+rho(i+1,j,k+1)+rho(i,j+1,k+1)+rho(i+1,j+1,k+1))
```

```
rho_scaled=rho_scaled+rho_scaled_int
end do
end do
end do

write(*,*) "Total number of atoms is: ", rho_scaled, rho_unscaled
end subroutine

!-----------------------------------------------------------------------
subroutine makeclean(nmax, u, delsqrho, temp_rhobar)
implicit real*8 (a-h, o-z)

include 'mpif.h'
dimension u(nmax,nmax,nmax), temp_rhobar(nmax,nmax,nmax)
dimension delsqrho(nmax,nmax,nmax)

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   u(i,j,k)=0.0d0
   temp_rhobar(i,j,k)=0.0d0
   delsqrho(i,j,k)=0.0d0
  end do
 end do
end do

end subroutine
!-----------------------------------------------------------------

subroutine mf_ST(rho, u, u_new, delsqrho, nmax, dx, dy, dz, nproc, rank, handle)
implicit real*8 (a-h, o-z)

include 'mpif.h'
parameter (pi=4.0d0*atan(1.0d0))

dimension rho(nmax,nmax,nmax)
dimension delsqrho(nmax,nmax,nmax)
dimension delsqrho_tot(nmax,nmax,nmax)
dimension u(nmax,nmax,nmax), u_new(nmax,nmax,nmax)
integer :: rank
character(len=4) :: handle

!Now define the mean field.

b=-8.88810d2/3.1577504d5*1.889725989d0**3
c4=(1.04554d7/3.1577504d5)*1.889725989d0**(3.0*(2.8d0+1.0d0))
d=2.383d3/3.1577504d5*1.889725989d0**5
gamm4=2.8d0
coeff=(2.0d0+gamm4)/2.0d0

do i=rank+1, nmax, nproc
do j=1, nmax
do k=1, nmax

!This assumes that the density is flat at both end points, so that we can set the Laplacian to zero at the
!two end points.
! if (i.le.2.or.j.le.2.or.k.le.2) then
!  delsqrho(i,j,k)=0.0d0
! else if (i.ge.(nmax-1).or.j.ge.(nmax-1).or.k.ge.(nmax-1)) then

! else
!  delsqrx=(-rho(i-2,j,k)+16.0*rho(i-1,j,k)-30.0*rho(i,j,k)+16.0*rho(i+1,j,k)-rho(i+2,j,k))/(12.0*dx*dx)

!  delsqry=(-rho(i,j-2,k)+16.0*rho(i,j-1,k)-30.0*rho(i,j,k)+16.0*rho(i,j+1,k)-rho(i,j+2,k))/(12.0*dy*dy)

!  delsqrz=(-rho(i,j,k-2)+16.0*rho(i,j,k-1)-30.0*rho(i,j,k)+16.0*rho(i,j,k+1)-rho(i,j,k+2))/(12.0*dz*dz)

!  delsqrho(i,j,k)=delsqrx+delsqry+delsqrz
! end if

 if (i.eq.1.or.j.eq.1.or.k.eq.1) then
  delsqrho(i,j,k)=0.0d0
 else if (i.eq.nmax.or.j.eq.nmax.or.k.eq.nmax) then
  delsqrho(i,j,k)=0.0d0
 else
  delsqrx=(rho(i-1,j,k)-2.0*rho(i,j,k)+rho(i+1,j,k))/(dx*dx)
  delsqry=(rho(i,j-1,k)-2.0*rho(i,j,k)+rho(i,j+1,k))/(dy*dy)
  delsqrz=(rho(i,j,k-1)-2.0*rho(i,j,k)+rho(i,j,k+1))/(dz*dz)

  delsqrho(i,j,k)=delsqrx+delsqry+delsqrz
 end if
end do
end do
end do

mpi_red_int=nmax*nmax*nmax
call MPI_ALLREDUCE(delsqrho, delsqrho_tot, mpi_red_int, MPI_DOUBLE_PRECISION, MPI_SUM, MPI_COMM_WORLD, ierr)
```

```
if (rank.eq.0) then
open(333,file='delsqrho')
do i=1, nmax
 write(333,3333) i, b*rho(i,nmax/2,nmax/2), coeff*c4*rho(i,nmax/2,nmax/2)**(1.0d0+gamm4),
 -2.0d0*d*delsqrho_tot(i,nmax/2,nmax/2)

end do
close (333)
end if
3333 format (i6, 3(1x, 1pe12.5))

!Now define the mean field.

do i=rank+1, nmax/2, nproc
do j=1, nmax/2
do k=1, nmax/2
 u(i,j,k)=b*rho(i,j,k)+coeff*c4*rho(i,j,k)**(1.0d0+gamm4)-(2.0d0*d*delsqrho_tot(i,j,k))
end do
end do
end do

call MPI_REDUCE(u, u_new, mpi_red_int, MPI_DOUBLE_PRECISION, MPI_SUM, 0, MPI_COMM_WORLD, ierr)

return
end subroutine

!----------------------------------------------------------------------

subroutine mf_OP(rho, u, temp_rhobar, u_new, nmax, n_cutoff, x, y, z, dx, dy, dz, nproc, rank, handle)
implicit real*8 (a-h, o-z)

include 'mpif.h'
parameter (pi=4.0d0*atan(1.0d0))

dimension rho(nmax,nmax,nmax), rhobarint(nmax,nmax,nmax)
dimension u(nmax,nmax,nmax), u_new(nmax,nmax,nmax),  rhobar(nmax,nmax,nmax)
dimension temp_rhobar(nmax,nmax,nmax)
dimension cterm_array(nmax,nmax,nmax)
dimension x(nmax), y(nmax), z(nmax)
dimension dist(8), rhotemp(8), rhobartemp(8)
integer :: rank
character(len=4) :: handle

!real*8, dimension(nmax) :: lj
!dimension zeta1(nmax), zeta2(nmax)

!Now define the mean field.
!1 Hartree = 3.1577504d5 K = 27.21138505 eV
!1 Angstrom = 1.889725989 bohr

b=-8.88810d2/3.1577504d5*1.889725989d0**3.0d0
c4=(1.04554d7/3.1577504d5)*1.889725989d0**(3.0d0*(2.8d0+1.0d0))
h=2.376728d0*1.889725989d0
hsq=h*h
sig=2.556d0*1.889725989d0
sig6=sig**6
sig12=sig**12
ep=10.22d0/3.1577504d5
gamm4=2.8d0
Vhop=4.0d0*ep*((sig/h)**12.0d0-(sig/h)**6.0d0)
rhobar_prefactor=3.0/(4.0*pi*h**3.0)*(dx*dy*dz)/8.0

!The local potential within a sphere is given by Equation 2.11 in the Szybisz paper:

do i=rank+1, nmax, nproc
do j=1, nmax
do k=1, nmax

fullrhobar=0.0

do iprime=i-h/dx, i+h/dx
do jprime=j-h/dy, j+h/dy
do kprime=k-h/dz, k+h/dz

if (iprime.ge.1.and.iprime.lt.nmax) then
if (jprime.ge.1.and.jprime.lt.nmax) then
if (kprime.ge.1.and.kprime.lt.nmax) then

dist(1)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(2)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(3)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(4)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(5)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(6)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(7)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2
dist(8)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2

!Each corner of the cube is also assigned a density value.
```

```
rhotemp(1)=rho(iprime,jprime,kprime)
rhotemp(2)=rho(iprime+1,jprime,kprime)
rhotemp(3)=rho(iprime,jprime+1,kprime)
rhotemp(4)=rho(iprime+1,jprime+1,kprime)
rhotemp(5)=rho(iprime,jprime,kprime+1)
rhotemp(6)=rho(iprime+1,jprime,kprime+1)
rhotemp(7)=rho(iprime,jprime+1,kprime+1)
rhotemp(8)=rho(iprime+1,jprime+1,kprime+1)

rhobarscalar=0.0

do n=1, 8
 if (dist(n).lt.hsq) then
  rhobarscalar=rhobarscalar+rhobar_prefactor*rhotemp(n)
 end if
end do

fullrhobar=fullrhobar+rhobarscalar

end if
end if
end if

end do
end do
end do

temp_rhobar(i,j,k)=fullrhobar

end do
end do
end do

mpi_red_int=nmax*nmax*nmax
call MPI_ALLREDUCE(temp_rhobar, rhobar, mpi_red_int, MPI_DOUBLE_PRECISION,MPI_SUM, MPI_COMM_WORLD, ierr)

if (rank.eq.0) then
open(11,file='rhobar'//trim(handle)//'.out')
do i=1, nmax
 write(11,*) i, z(i), rhobar(nmax/2,nmax/2,i)
end do
close (11)
end if

!Begin calculation of the bterm.

do i=rank+1, nmax/2, nproc
do j=1, nmax/2
do k=1, nmax/2

full_rep=0.0
full_att=0.0
full_straddle=0.0

do iprime=i-n_cutoff, i+n_cutoff
do jprime=j-n_cutoff, j+n_cutoff
do kprime=k-n_cutoff, k+n_cutoff

if (iprime.ge.1.and.iprime.le.(nmax-1)) then
if (jprime.ge.1.and.jprime.le.(nmax-1)) then
if (kprime.ge.1.and.kprime.le.(nmax-1)) then

!The different values of dist correspond to the different distances to the
!corners of the cube being evaluated.

dist(1)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(2)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(3)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(4)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(5)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(6)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(7)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2
dist(8)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2

!Each corner of the cube is also assigned a density value.

rhotemp(1)=rho(iprime,jprime,kprime)
rhotemp(2)=rho(iprime+1,jprime,kprime)
rhotemp(3)=rho(iprime,jprime+1,kprime)
rhotemp(4)=rho(iprime+1,jprime+1,kprime)
rhotemp(5)=rho(iprime,jprime,kprime+1)
rhotemp(6)=rho(iprime+1,jprime,kprime+1)
rhotemp(7)=rho(iprime,jprime+1,kprime+1)
rhotemp(8)=rho(iprime+1,jprime+1,kprime+1)

!If the cube is entirely within the cutoff radius h, use the short-range portion
!of the potential.
```

```
icount=0

do n=1, 8
 if (dist(n).lt.hsq) then
  icount=icount+1
 end if
end do

if (icount.eq.8) then
 do n=1,8
  cubeint_rep=Vhop/(hsq*hsq)*(dx*dy*dz)/8.0d0*dist(n)*dist(n)*rhotemp(n)
  full_rep=full_rep+cubeint_rep
 end do
else if (icount.eq.0) then
 do n=1,8
  cubeint_att=4.0d0*ep*(dx*dy*dz)/8.0d0*((sig12/dist(n)**6)-(sig6/dist(n)**3))*rhotemp(n)
  full_att=full_att+cubeint_att
 end do
else
 do n=1,8
  if (dist(n).lt.hsq) then
   cubeint_straddle=Vhop/(hsq*hsq)*(dx*dy*dz)/8.0d0*dist(n)*dist(n)*rhotemp(n)
  else
   cubeint_straddle=4.0d0*ep*(dx*dy*dz)/8.0d0*((sig12/dist(n)**6)-(sig6/dist(n)**3))*rhotemp(n)
  end if
  full_straddle=full_straddle+cubeint_straddle
 end do
end if

end if
end if
end if

end do
end do
end do

u(i,j,k)=u(i,j,k)+full_rep+full_att+full_straddle

end do
end do
end do

!Begin calculation of the cterm.

cubeint2_prefactor=3.0/(4.0*pi*h**3.0)*(1+gamm4)*(dx*dy*dz)/8.0

do i=1, nmax
do j=1, nmax
do k=1, nmax

cterm_array(i,j,k)=rho(i,j,k)*rhobar(i,j,k)**gamm4

end do
end do
end do

do i=rank+1, nmax/2, nproc
do j=1, nmax/2
do k=1, nmax/2

fullint2=0.0

do iprime=1, nmax-1
do jprime=1, nmax-1
do kprime=1, nmax-1

dist(1)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(2)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(3)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(4)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(5)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(6)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(7)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2
dist(8)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2

rhotemp(1)=cterm_array(iprime,jprime,kprime)
rhotemp(2)=cterm_array(iprime+1,jprime,kprime)
rhotemp(3)=cterm_array(iprime,jprime+1,kprime)
rhotemp(4)=cterm_array(iprime+1,jprime+1,kprime)
rhotemp(5)=cterm_array(iprime,jprime,kprime+1)
rhotemp(6)=cterm_array(iprime+1,jprime,kprime+1)
rhotemp(7)=cterm_array(iprime,jprime+1,kprime+1)
rhotemp(8)=cterm_array(iprime+1,jprime+1,kprime+1)

icount=0
```

```
do n=1,8
 if (dist(n).lt.hsq) then
  icount=icount+1
 end if
end do

if (icount.eq.8) then
 cubeint2=0.0
 do n=1,8
  cubeint2=cubeint2+cubeint2_prefactor*rhotemp(n)
 end do
else if (icount.gt.0.and.icount.lt.8) then
 cubeint2=0.0
 do n=1,8
  if (dist(n).lt.hsq) then
   cubeint2=cubeint2+cubeint2_prefactor*rhotemp(n)
  end if
 end do
else
 cubeint2=0.0
end if
fullint2=fullint2+cubeint2

end do
end do
end do

u(i,j,k)=u(i,j,k)+c4/2.0*rhobar(i,j,k)**(gamm4+1)+c4/2.0*fullint2

end do
end do
end do

call MPI_REDUCE(u, u_new, mpi_red_int, MPI_DOUBLE_PRECISION, MPI_SUM, 0, MPI_COMM_WORLD, ierr)

if (rank.eq.0) then
open (10, file='u'//trim(handle)//'.out')
do i=1, nmax
 write(10,*) i, z(i), u_new(nmax/2,nmax/2,i)
end do
close (10)
end if

return
end subroutine

!-------------------------------------------------------------------------------

subroutine symmetry(u_new, nmax, handle)
implicit real*8 (a-h, o-z)

include 'mpif.h'
dimension u_new(nmax,nmax,nmax)
character(len=4) :: handle

!Quad 2
do i=nmax/2+1, nmax
 do j=1, nmax/2
  do k=1, nmax/2
   u_new(i,j,k)=u_new(nmax-i+1,j,k)
  end do
 end do
end do

!Quad 3
do i=1, nmax/2
 do j=nmax/2+1, nmax
  do k=1, nmax/2
   u_new(i,j,k)=u_new(i,nmax-j+1,k)
  end do
 end do
end do

!Quad 4
do i=nmax/2+1, nmax
 do j=nmax/2+1, nmax
  do k=1, nmax/2
   u_new(i,j,k)=u_new(nmax-i+1,nmax-j+1,k)
  end do
 end do
end do

!Quad 5
do i=1, nmax/2
 do j=1, nmax/2
  do k=nmax/2+1, nmax
   u_new(i,j,k)=u_new(i,j,nmax-k+1)
  end do
```

```
 end do
end do

!Quad 6
do i=nmax/2+1, nmax
 do j=1, nmax/2
  do k=nmax/2+1, nmax
   u_new(i,j,k)=u_new(nmax-i+1,j,nmax-k+1)
  end do
 end do
end do

!Quad 7
do i=1, nmax/2
 do j=nmax/2+1, nmax
  do k=nmax/2+1, nmax
   u_new(i,j,k)=u_new(i,nmax-j+1,nmax-k+1)
  end do
 end do
end do

!Quad 8
do i=nmax/2+1, nmax
 do j=nmax/2+1, nmax
  do k=nmax/2+1, nmax
   u_new(i,j,k)=u_new(nmax+1-i,nmax+1-j,nmax+1-k)
  end do
 end do
end do


if (rank.eq.0) then
open (10, file='u'//trim(handle)//'.out')
do i=1, nmax
 write(10,4500) i,  u_new(i,nmax/2,nmax/2), u_new(nmax/2,i,nmax/2), u_new(nmax/2,nmax/2,i)
end do
4500 format (i6, 3(1x, 1pe12.5))
close (10)
end if

return
end subroutine

!-----------------------------------------------------------------------------------------------------------

subroutine itp(nmax, dx, dy, dz, dt, x, y, z, rho, u_new, psi, newpsi, chempot, totrho1)
implicit real*8 (a-h, o-z)

dimension x(nmax), y(nmax), z(nmax), p_x(nmax), p_y(nmax), p_z(nmax)
dimension psi(nmax, nmax, nmax), temppsi(nmax, nmax, nmax), temppsi2(nmax, nmax, nmax)
real*8, dimension(nmax,nmax,nmax) ::  newpsi, newrho, rho
dimension oldrho(nmax, nmax, nmax), chemrho(nmax, nmax, nmax), u_new(nmax, nmax, nmax)
dimension ratio(nmax), oldpsi(nmax), backpsi(nmax)
dimension wsave((2*nmax+15), (2*nmax+15))
parameter (pi=4.0*atan(1.0))
parameter (hbar=1.0)
real*8 :: l

!This is M = the mass of He-4 in atomic units.

cc=7296.29

!This is the imaginary time step, tau.

!dt=5000.0

!Generate psi.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   psi(i,j,k)=sqrt(rho(i,j,k))
  end do
 end do
end do

!Generate the momentum array for the FFT.

x_len=x(nmax)+dx
y_len=y(nmax)+dy
z_len=z(nmax)+dz

p_x(1)=0.0
p_y(1)=0.0
p_z(1)=0.0

do n=2, nmax-1, 2
 p_x(n)=p_x(n-1)+(1.0/(2.0*dx))*4.0*pi*dx/x_len
```

```
 p_x(n+1)=p_x(n)

 p_y(n)=p_y(n-1)+(1.0/(2.0*dy))*4.0*pi*dy/y_len
 p_y(n+1)=p_y(n)

 p_z(n)=p_z(n-1)+(1.0/(2.0*dz))*4.0*pi*dz/z_len
 p_z(n+1)=p_z(n)
end do

!Perform the first potential energy step in the itp formalism.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   temppsi(i,j,k)=exp(-u_new(i,j,k)*dt/(2.0*hbar))*psi(i,j,k)
  end do
 end do
end do

!Perform the FFT for the kinetic energy operator.

call dffti(nmax, wsave)
!Perform the FFT in the z direction.

do i=1, nmax
do j=1, nmax

do k=1, nmax
 oldpsi(k)=temppsi(i,j,k)
end do

call dfftf(nmax, oldpsi, wsave)

do k=1, nmax
 temppsi(i,j,k)=oldpsi(k)
end do

end do
end do

!Perform the FFT in the y direction.

do i=1, nmax
do k=1, nmax

do j=1, nmax
 oldpsi(j)=temppsi(i,j,k)
end do

call dfftf(nmax, oldpsi, wsave)

do j=1, nmax
 temppsi(i,j,k)=oldpsi(j)
end do

end do
end do

!Perform the FFT in the x direction.

do j=1, nmax
do k=1, nmax

do i=1, nmax
 oldpsi(i)=temppsi(i,j,k)
end do

call dfftf(nmax, oldpsi, wsave)

do i=1, nmax
 temppsi(i,j,k)=oldpsi(i)
end do

end do
end do

!Evaluate the kinetic energy operator in three dimensions.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   newpsi(i,j,k)=exp(-(p_x(i)**2+p_y(j)**2+p_z(k)**2)*dt/(2.0*cc*hbar))*temppsi(i,j,k)
  end do
 end do
end do
do i=1, nmax
do j=1, nmax
```

```
do k=1, nmax
 backpsi(k)=newpsi(i,j,k)
end do

call dfftb(nmax, backpsi, wsave)

do k=1, nmax
 newpsi(i,j,k)=backpsi(k)/nmax
end do

end do
end do

do i=1, nmax
do k=1, nmax

do j=1, nmax
 backpsi(j)=newpsi(i,j,k)
end do

call dfftb(nmax, backpsi, wsave)

do j=1, nmax
 newpsi(i,j,k)=backpsi(j)/nmax
end do

end do
end do
do j=1, nmax
do k=1, nmax

do i=1, nmax
 backpsi(i)=newpsi(i,j,k)
end do

call dfftb(nmax, backpsi, wsave)

do i=1, nmax
 newpsi(i,j,k)=backpsi(i)/nmax
end do

end do
end do

!Perform the second potential energy step in the itp formalism.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   newpsi(i,j,k)=exp(-u_new(i,j,k)*dt/(2.0*hbar))*newpsi(i,j,k)
  end do
 end do
end do

!Normalize the new wavefunction. This ensures the old and new integrals are
!equal.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   oldrho(i,j,k)=psi(i,j,k)**2.0
   newrho(i,j,k)=newpsi(i,j,k)**2.0
   chemrho(i,j,k)=psi(i,j,k)*newpsi(i,j,k)
  end do
 end do
end do

totrho1=0.0d0
totrho2=0.0d0
totrho3=0.0d0

!Use trapezoid rule integration to compute the integral.

do i=1, nmax-1
do j=1, nmax-1
do k=1, nmax-1

rhoint1=dx*dy*dz/8.0d0*(oldrho(i,j,k)+oldrho(i+1,j,k)+oldrho(i,j+1,k)+oldrho(i+1,j+1,k)+oldrho(i,j,k+1)
+oldrho(i+1,j,k+1)+oldrho(i,j+1,k+1)+oldrho(i+1,j+1,k+1))

rhoint2=dx*dy*dz/8.0d0*(newrho(i,j,k)+newrho(i+1,j,k)+newrho(i,j+1,k)+newrho(i+1,j+1,k)+newrho(i,j,k+1)
+newrho(i+1,j,k+1)+newrho(i,j+1,k+1)+newrho(i+1,j+1,k+1))

rhoint3=dx*dy*dz/8.0d0*(chemrho(i,j,k)+chemrho(i+1,j,k)+chemrho(i,j+1,k)+chemrho(i+1,j+1,k)+chemrho(i,j,k+1)
+chemrho(i+1,j,k+1)+chemrho(i,j+1,k+1)+chemrho(i+1,j+1,k+1))

totrho1=totrho1+rhoint1
totrho2=totrho2+rhoint2
```

```
totrho3=totrho3+rhoint3

end do
end do
end do

!do n=1, nmax
! ratio(n)=newpsi(n)/psi(n)
!end do

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   newpsi(i,j,k)=(newpsi(i,j,k)/sqrt(totrho2))*sqrt(totrho1)
  end do
 end do
end do

2000 format (i6, 3(1x, 1pe12.5))

!The eigenvalue to the corresponding eigenvector (psi) is the chemical
!potential of the system.

chempot=-log(totrho3/totrho1)*hbar/(dt)
chempot2=-log(totrho2/totrho1)*hbar/(2.0*dt)

open (715, file='test_chempots')
 write(715,7150) chempot*3.1577504d5, chempot2*3.1577504d5
close (715)

7150 format(4(1x, 1pe12.5))

write(*,*) "Chemical potential is: ", chempot*3.1577504d5
write(*,*) "Total number of atoms is: ", totrho1
return
end subroutine

!-------------------------------------------------------------------------------

subroutine update(nmax, z, rho, psi, newpsi, chempot, iter, maxdiff, handle)
implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))

dimension z(nmax)
dimension rho(nmax,nmax,nmax), psi(nmax,nmax,nmax)
dimension rhonew(nmax,nmax,nmax)
real*8, dimension(nmax,nmax,nmax) :: newpsi
real*8 :: maxdiff
character(len=4) :: handle

maxdiff=0.0
totrho=0.0
brho=0.021836d0/1.889725989d0**3.0d0

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   rhonew(i,j,k)=newpsi(i,j,k)**2.0d0
!   if (rhonew(i,j,k).gt.brho) then
!    rhonew(i,j,k)=brho
!   end if

   if (abs(newpsi(i,j,k)-psi(i,j,k)).gt.maxdiff) then
    maxdiff=abs(newpsi(i,j,k)-psi(i,j,k))
   end if

  end do
 end do
end do

!Update psi and the density.

open(3,file='sphericalpsi'//trim(handle)//'.out')
do n=1, nmax
 write(3,3000) n, z(n), rhonew(n,nmax/2,nmax/2), rhonew(nmax/2,n,nmax/2), rhonew(nmax/2,nmax/2,n)
end do

!Include this cutoff when using LJ parameters.

!do n=1, nmax
! if (rhonew(n).lt.1e-30) then
!  rhonew(n)=0.0
! end if
!end do

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
```

```
   psi(i,j,k)=newpsi(i,j,k)
   rho(i,j,k)=rhonew(i,j,k)
  end do
 end do
end do

3000 format (i6, 5(1x, 1pe12.5))
close(3)

write(*,*) "Maximum psi difference is", maxdiff

end subroutine

!-------------------------------------------------------------------------------

subroutine checkpoint(nmax, time_old, rho, handle)
implicit real*8 (a-h, o-z)

include 'mpif.h'
integer, dimension(3) :: time_old, time_new
dimension rho(nmax,nmax,nmax)
character(len=4) :: handle

update_interval=30.0

call itime(time_new)

minutes_old=60*time_old(1)+time_old(2)
minutes_new=60*time_new(1)+time_new(2)

if ((minutes_new-minutes_old).ge.update_interval) then
 do n=1,3
  time_old(n)=time_new(n)
 end do

 open (4, file='save_den'//trim(handle))
  do i=1, nmax
   do j=1, nmax
    do k=1, nmax
     write(4,4000) i, j, k, rho(i,j,k)
    end do
   end do
  end do
  write(4,*)
  write(4,*) time_new(1),":",time_new(2),":",time_new(3)
 close (4)
end if

4000 format (i4, i4, i4, (1x, 1pe12.5))
end subroutine

!-------------------------------------------------------------------------

subroutine mf_BH(rho, u, temp_rhobar, u_new, nmax, n_cutoff, x, y, z, dx, dy, dz, nproc, rank, handle)
implicit real*8 (a-h, o-z)

include 'mpif.h'
parameter (pi=4.0d0*atan(1.0d0))

dimension rho(nmax,nmax,nmax), rhobarint(nmax,nmax,nmax)
dimension u(nmax,nmax,nmax), u_new(nmax,nmax,nmax), rhobar(nmax,nmax,nmax)
dimension temp_rhobar(nmax,nmax,nmax)
dimension cterm_array(nmax,nmax,nmax)
dimension x(nmax), y(nmax), z(nmax)
dimension dist(8), rhotemp(8), rhobartemp(8)
integer :: rank
character(len=4) :: handle

!real*8, dimension(nmax) :: lj
!dimension zeta1(nmax), zeta2(nmax)

!Now define the mean field.
!1 Hartree = 3.1577504d5 K = 27.21138505 eV
!1 Angstrom = 1.889725989 bohr

b=-8.88810d2/3.1577504d5*1.889725989d0**3.0d0
c4=(1.04554d7/3.1577504d5)*1.889725989d0**(3.0d0*(2.8d0+1.0d0))
h=2.376728d0*1.889725989d0
hsq=h*h
sig=2.556d0*1.889725989d0
sig6=sig**6
sig12=sig**12
ep=10.22d0/3.1577504d5
b_lj=(33.0/8.0)*(8.0/9.0*ep+b/(4.0*pi*sig**3))
gamm4=2.8d0
rhobar_prefactor=3.0/(4.0*pi*h**3.0)*(dx*dy*dz)/8.0

!The local potential within a sphere is given by Equation 2.11 in the Szybisz
```

```
!paper:

do i=rank+1, nmax, nproc
do j=1, nmax
do k=1, nmax

fullrhobar=0.0

do iprime=i-h/dx, i+h/dx
do jprime=j-h/dy, j+h/dy
do kprime=k-h/dz, k+h/dz

if (iprime.ge.1.and.iprime.lt.nmax) then
if (jprime.ge.1.and.jprime.lt.nmax) then
if (kprime.ge.1.and.kprime.lt.nmax) then

dist(1)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(2)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(3)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(4)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(5)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(6)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(7)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2
dist(8)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2

rhotemp(1)=rho(iprime,jprime,kprime)
rhotemp(2)=rho(iprime+1,jprime,kprime)
rhotemp(3)=rho(iprime,jprime+1,kprime)
rhotemp(4)=rho(iprime+1,jprime+1,kprime)
rhotemp(5)=rho(iprime,jprime,kprime+1)
rhotemp(6)=rho(iprime+1,jprime,kprime+1)
rhotemp(7)=rho(iprime,jprime+1,kprime+1)
rhotemp(8)=rho(iprime+1,jprime+1,kprime+1)

rhobarscalar=0.0

do n=1, 8
 if (dist(n).le.hsq) then
  rhobarscalar=rhobarscalar+rhobar_prefactor*rhotemp(n)
 end if
end do

fullrhobar=fullrhobar+rhobarscalar

end if
end if
end if

end do
end do
end do

temp_rhobar(i,j,k)=fullrhobar

end do
end do
end do

mpi_red_int=nmax*nmax*nmax

call MPI_ALLREDUCE(temp_rhobar, rhobar, mpi_red_int, MPI_DOUBLE_PRECISION,MPI_SUM, MPI_COMM_WORLD, ierr)

if (rank.eq.0) then
open(11,file='rhobar'//trim(handle)//'.out')
do i=1, nmax
 write(11,*) i, z(i), rhobar(nmax/2,nmax/2,i)
end do
close (11)
end if

!Begin calculation of the bterm.

do i=rank+1, nmax/2, nproc
do j=1, nmax/2
do k=1, nmax/2

full_rep=0.0
full_att=0.0
full_straddle=0.0

do iprime=i-n_cutoff, i+n_cutoff-1
do jprime=j-n_cutoff, j+n_cutoff-1
do kprime=k-n_cutoff, k+n_cutoff-1

if (iprime.ge.1.and.iprime.le.(nmax-1)) then
if (jprime.ge.1.and.jprime.le.(nmax-1)) then
if (kprime.ge.1.and.kprime.le.(nmax-1)) then
```

```
!The different values of dist correspond to the different distances to the
!corners of the cube being evaluated.

dist(1)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(2)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(3)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(4)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(5)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(6)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(7)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2
dist(8)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2

!Each corner of the cube is also assigned a density value.

rhotemp(1)=rho(iprime,jprime,kprime)
rhotemp(2)=rho(iprime+1,jprime,kprime)
rhotemp(3)=rho(iprime,jprime+1,kprime)
rhotemp(4)=rho(iprime+1,jprime+1,kprime)
rhotemp(5)=rho(iprime,jprime,kprime+1)
rhotemp(6)=rho(iprime+1,jprime,kprime+1)
rhotemp(7)=rho(iprime,jprime+1,kprime+1)
rhotemp(8)=rho(iprime+1,jprime+1,kprime+1)

!If the cube is entirely within the cutoff radius h, use the short-range portion
!of the potential.

icount=0

do n=1, 8
 if (dist(n).lt.(sig*sig)) then
  icount=icount+1
 end if
end do

if (icount.eq.8) then
 do n=1,8
  cubeint_rep=(dx*dy*dz)/8.0d0*b_lj*(1.0-(dist(n)**4/sig**8))*rhotemp(n)
  full_rep=full_rep+cubeint_rep
 end do
else if (icount.eq.0) then
 do n=1,8
  cubeint_att=4.0d0*ep*(dx*dy*dz)/8.0d0*((sig12/dist(n)**6)-(sig6/dist(n)**3))*rhotemp(n)
  full_att=full_att+cubeint_att
 end do
else
 do n=1,8
  if (dist(n).lt.(sig*sig)) then
   cubeint_straddle=(dx*dy*dz)/8.0d0*b_lj*(1.0-(dist(n)**4/sig**8))*rhotemp(n)
  else
   cubeint_straddle=4.0d0*ep*(dx*dy*dz)/8.0d0*((sig12/dist(n)**6)-(sig6/dist(n)**3))*rhotemp(n)
  end if
  full_straddle=full_straddle+cubeint_straddle
 end do
end if

end if
end if
end if

end do
end do
end do

u(i,j,k)=u(i,j,k)+full_rep+full_att+full_straddle

end do
end do
end do

!Begin calculation of the cterm.

cubeint2_prefactor=3.0/(4.0*pi*h**3)*(1.0+gamm4)*(dx*dy*dz)/8.0

do i=1, nmax
do j=1, nmax
do k=1, nmax

cterm_array(i,j,k)=rho(i,j,k)*rhobar(i,j,k)**gamm4

end do
end do
end do

do i=rank+1, nmax/2, nproc
do j=1, nmax/2
do k=1, nmax/2

fullint2=0.0
```

```
do iprime=1, nmax-1
do jprime=1, nmax-1
do kprime=1, nmax-1

dist(1)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(2)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime))**2
dist(3)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(4)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime))**2
dist(5)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(6)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime))**2+(dz*dble(k-kprime-1))**2
dist(7)=(dx*dble(i-iprime))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2
dist(8)=(dx*dble(i-iprime-1))**2+(dy*dble(j-jprime-1))**2+(dz*dble(k-kprime-1))**2

rhotemp(1)=cterm_array(iprime,jprime,kprime)
rhotemp(2)=cterm_array(iprime+1,jprime,kprime)
rhotemp(3)=cterm_array(iprime,jprime+1,kprime)
rhotemp(4)=cterm_array(iprime+1,jprime+1,kprime)
rhotemp(5)=cterm_array(iprime,jprime,kprime+1)
rhotemp(6)=cterm_array(iprime+1,jprime,kprime+1)
rhotemp(7)=cterm_array(iprime,jprime+1,kprime+1)
rhotemp(8)=cterm_array(iprime+1,jprime+1,kprime+1)

icount=0

do n=1,8
 if (dist(n).lt.hsq) then
  icount=icount+1
 end if
end do

if (icount.eq.8) then
 cubeint2=0.0
 do n=1,8
  cubeint2=cubeint2+cubeint2_prefactor*rhotemp(n)
 end do
else if (icount.gt.0.and.icount.lt.8) then
 cubeint2=0.0
 do n=1,8
  if (dist(n).lt.hsq) then
   cubeint2=cubeint2+cubeint2_prefactor*rhotemp(n)
  end if
 end do
else
 cubeint2=0.0
end if
fullint2=fullint2+cubeint2

end do
end do
end do

u(i,j,k)=u(i,j,k)+c4/2.0*rhobar(i,j,k)**(gamm4+1)+c4/2.0*fullint2

end do
end do
end do

call MPI_REDUCE(u, u_new, mpi_red_int, MPI_DOUBLE_PRECISION, MPI_SUM, 0, MPI_COMM_WORLD, ierr)

if (rank.eq.0) then
open (10, file='u'//trim(handle)//'.out')
do i=1, nmax
 write(10,*) i, z(i), u_new(nmax/2,nmax/2,i)
end do
close (10)
end if

return
end subroutine
```

A.2 ITP Algorithm for 3-D He-DFT Code

```
subroutine itp(nmax, dx, dy, dz, x, y, z, u, psi, newpsi, chempot)
implicit real*8 (a-h, o-z)
integer, parameter :: matsize=20
real*8, dimension(1:matsize) :: x, y, z, p_x, p_y, p_z
real*8, dimension(matsize, matsize, matsize) :: psi, temppsi, temppsi2, newpsi
real*8, dimension(matsize, matsize, matsize) :: oldrho, newrho, chemrho, u
real*8, dimension(1:matsize) :: ratio, oldpsi, backpsi
real*8, dimension((2*matsize+15), (2*matsize+15)) :: wsave
real*8, parameter :: pi=4.0*atan(1.0)
real*8, parameter :: hbar=1.0
real*8 :: l
!This is M = the mass of He-4 in atomic units.

cc=7296.29

!This is the imaginary time step, tau.

dt=1000.0

!Check that the dimensions of the arrays are correct.

if (nmax.ne.matsize) then
 write(*,*) "Matsize wrong in itp subroutine"
 stop
end if

!Generate the momentum array for the FFT.

x_len=x(nmax)+dx
y_len=y(nmax)+dy
z_len=z(nmax)+dz

p_x(1)=0.0
p_y(1)=0.0
p_z(1)=0.0

do n=2, nmax-1, 2
 p_x(n)=p_x(n-1)+(1.0/(2.0*dx))*4.0*pi*dx/x_len
 p_x(n+1)=p_x(n)

 p_y(n)=p_y(n-1)+(1.0/(2.0*dy))*4.0*pi*dy/y_len
 p_y(n+1)=p_y(n)

 p_z(n)=p_z(n-1)+(1.0/(2.0*dz))*4.0*pi*dz/z_len
 p_z(n+1)=p_z(n)
end do

!Perform the first potential energy step in the itp formalism.

!open(200, file='1')
do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   temppsi(i,j,k)=exp(-u(i,j,k)*dt/(2.0*hbar))*psi(i,j,k)
  end do
 end do
end do
!close(200)

!Perform the FFT for the kinetic energy operator.

call dffti(matsize, wsave)

!Perform the FFT in the z direction.

do i=1, nmax
do j=1, nmax

do k=1, nmax
 oldpsi(k)=temppsi(i,j,k)
end do

call dfftf(matsize, oldpsi, wsave)

do k=1, nmax
 temppsi(i,j,k)=oldpsi(k)
end do

end do
end do

!Perform the FFT in the y direction.

do i=1, nmax
```

```
do k=1, nmax

do j=1, nmax
 oldpsi(j)=temppsi(i,j,k)
end do

call dfftf(matsize, oldpsi, wsave)

do j=1, nmax
 temppsi(i,j,k)=oldpsi(j)
end do

end do
end do

!Perform the FFT in the x direction.

do j=1, nmax
do k=1, nmax

do i=1, nmax
 oldpsi(i)=temppsi(i,j,k)
end do

call dfftf(matsize, oldpsi, wsave)

do i=1, nmax
 temppsi(i,j,k)=oldpsi(i)
end do

end do
end do

!Evaluate the kinetic energy operator in three dimensions.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   newpsi(i,j,k)=exp(-(p_x(i)**2.0+p_y(j)**2.0+p_z(k)**2.0)*dt/(2.0*cc*hbar))*temppsi(i,j,k)
  end do
 end do
end do

do i=1, nmax
do j=1, nmax

do k=1, nmax
 backpsi(k)=newpsi(i,j,k)
end do

call dfftb(matsize, backpsi, wsave)

do k=1, nmax
 newpsi(i,j,k)=backpsi(k)/nmax
end do

end do
end do

do i=1, nmax
do k=1, nmax

do j=1, nmax
 backpsi(j)=newpsi(i,j,k)
end do

call dfftb(matsize, backpsi, wsave)

do j=1, nmax
 newpsi(i,j,k)=backpsi(j)/nmax
end do

end do
end do

do j=1, nmax
do k=1, nmax

do i=1, nmax
 backpsi(i)=newpsi(i,j,k)
end do

call dfftb(matsize, backpsi, wsave)

do i=1, nmax
 newpsi(i,j,k)=backpsi(i)/nmax
end do
```

```
end do
end do

!Perform the second potential energy step in the itp formalism.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   newpsi(i,j,k)=exp(-u(i,j,k)*dt/(2.0*hbar))*newpsi(i,j,k)
  end do
 end do
end do

!Normalize the new wavefunction. This ensures the old and new integrals are equal.

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   oldrho(i,j,k)=psi(i,j,k)**2.0
   newrho(i,j,k)=newpsi(i,j,k)**2.0
   chemrho(i,j,k)=psi(i,j,k)*newpsi(i,j,k)
  end do
 end do
end do

totrho1=0.0d0
totrho2=0.0d0
totrho3=0.0d0

!Use trapezoid rule integration to compute the integral.

do i=1, nmax-1
do j=1, nmax-1
do k=1, nmax-1

rhoint1=dx*dy*dz/8.0d0*(oldrho(i,j,k)+oldrho(i+1,j,k)+oldrho(i,j+1,k)+oldrho(i+1,j+1,k)+oldrho(i,j,k+1)
+oldrho(i+1,j,k+1)+oldrho(i,j+1,k+1)+oldrho(i+1,j+1,k+1))

rhoint2=dx*dy*dz/8.0d0*(newrho(i,j,k)+newrho(i+1,j,k)+newrho(i,j+1,k)+newrho(i+1,j+1,k)+newrho(i,j,k+1)
+newrho(i+1,j,k+1)+newrho(i,j+1,k+1)+newrho(i+1,j+1,k+1))

rhoint3=dx*dy*dz/8.0d0*(chemrho(i,j,k)+chemrho(i+1,j,k)+chemrho(i,j+1,k)+chemrho(i+1,j+1,k)+chemrho(i,j,k+1)
+chemrho(i+1,j,k+1)+chemrho(i,j+1,k+1)+chemrho(i+1,j+1,k+1))

totrho1=totrho1+rhoint1
totrho2=totrho2+rhoint2
totrho3=totrho3+rhoint3

end do
end do
end do

!do n=1, nmax
! ratio(n)=newpsi(n)/psi(n)
!end do

do i=1, nmax
 do j=1, nmax
  do k=1, nmax
   newpsi(i,j,k)=(newpsi(i,j,k)/sqrt(totrho2))*sqrt(totrho1)
  end do
 end do
end do

1000 format (i6, 3(1x, 1pe12.5))

!The eigenvalue to the corresponding eigenvector (psi) is the chemical potential!of the system.

chempot=-log(totrho3/totrho1)*hbar/(dt)

write(*,*) "Chemical potential is: ", chempot*3.1577504d5
return
end subroutine
```

Appendix B

1-D Droplet He-DFT Code

The following is the code used in calculating the properties of 1-D helium droplets as described in Chapter 3. It depends upon a user-generated input file called "input_He2".

```
PROGRAM hankel1D

!This program computes the density profile of a liquid helium slab that
!has a finite thickness but is infinitely large in the other
!two dimensions.

!Set all floating-point variables to be double precision by default.

implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))

!Dimension all of the vectors and matrices.

!rho=density matrix, u=meanfield matrix, psi=wavefunction matrix

real*8, allocatable :: rho(:), u(:), psi(:), newpsi(:)

!rhonew=density matrix returned after one step of itp, is set to rho at the end
!of each step
!u_new=total meanfield matrix passed to the itp component of the calculation

real*8, allocatable :: rhonew(:), r(:), rhobar(:), e_pot(:)

!The momentum-space x, y, and z matrices:

real*8, allocatable :: p_x(:), p_y(:), p_z(:)

!The value corresponding to the maximum difference between wf's between
!iterations (explicitly stated because m is otherwise an integer):
real*8 :: maxdiff

!A 1x3 time matrix containing hour, minute, and second data in each column,
!respectively

integer, dimension(3) :: time_old

!The character equivalent of 'ihandle', for use in naming files

character(len=4) :: handle

logical :: iscale
real*8 :: ljterm

!Open the input file "input_He2" and read relevant information from it -
!ihande = 4-number identifier attached to output files
!nr_per_h = number of points per cutoff distance h
!n_guess = the number of desired helium atoms in the droplet
```

```
!iter = number of iterations to be run
!dt = imaginary time step, delta tau

open (1, file='input_He2')
read(1,*) ihandle
read(1,*) nr_per_h
read(1,*) n_guess
read(1,*) iter
!read() functional
read(1,*) dt
close (1)

!Redefine ihandle as a character called 'handle' of length 4 for use in naming
!files
write(handle,1000) ihandle
1000 format(I4.4)

!iscale controls whether or not the starting density is scaled to the nearest
!integer (rounded up). It is automatically set to true, but will not scale a
!density due to a restarted/checkpointed job. This variable is now antiquated
!due to the subroutine rmid_opt.

iscale=.true.

!h is the cutoff radius in the Orsay-Paris and Orsay-Trento DFT schemes.
!The current value is for the Orsay-Paris h, scaled to units of bohr.

h=2.376728d0*1.889725989d0

!Define the spacing between points based on nr_per_h from earlier:

dr=h/dble(nr_per_h-1)

!Calculate the value rmid for the density profile corresponding to the number
!of desired helium atoms in the droplet through the subroutine rmid_opt.

call rmid_opt(dr, n_guess, rmid)

!Define the number of points necessary for the simulation.

nmax=ceiling(2.0*rmid/dr+1)

if (mod(nmax,2).eq.1) then
 nmax=nmax+1
end if

!h-bar is equal to 1 in atomic units.
hbar=1.0d0

!Initialize the radial array:
r_init = 0.0d0

allocate(r(nmax))

do i=1, nmax
 r(i)=(i-1.0)*dr
end do

rmax=r(nmax)
imax=(rmax-rmin)/dr+1

!Record some useful data regarding the system's physical dimensions.

open (1, file='SystemData'//trim(handle))
 write(1,*) "-----------------------------------------------------"
 write(1,*) "SYSTEM DIMENSIONS"
 write(1,*) "-----------------------------------------------------"
 write(1,*) "NMAX = ", nmax
 write(1,*) "Total number of points = ", nmax
 write(1,*) "n per h (z) = ", nr_per_h
 write(1,*) "R dimension length = ", r(nmax), "bohr"
close (1)

!Allocate memory for the rest of the vectors needed for the calculation:

allocate(rho(nmax))
allocate(u(nmax))
allocate(psi(nmax))
allocate(newpsi(nmax))
allocate(rhobar(nmax))
allocate(e_pot(nmax))

!Define the initial density function from Eq. (26) of Stringari
!and Treiner.  Note that although the article says this is a good
!approximation for 3-He, the authors also use the same function
!for 4-He, only with different parameters.  The parameter values
!for 4-He are listed in the middle of the left column of p. 8374
!(just above Fig. 3).
```

```
!Note that the density units are bohr**(-3).

brho=0.021836d0/1.889725989d0**3

!This is an antiquated test file for observing the initial density profile.

open (1,file='initden.utest')
do n=1, nmax
 rho(n)=brho/((1.0d0+exp((r(n)-rmid)/1.96d0))**2.5d0)
 psi(n)=r(n)*sqrt(rho(n))
 write(1,*) n, r(n), rho(n)
end do
close (1)

!-----This section is antiquated, but can be reactivated if restarting the-------
!--------------------------job is necessary-------------------------------------
!Now check if we have an existing input density in the file
!"testden2.utest".  STATUS=OLD with ERR=999 means that we jump ahead
!to line 999 if the file does not exist.

!This allows us to use the approximate density from an earlier
!simulation to speed things up.

open (3, file='testden2.utest', status='old', err=999)

!At this point, we know the file exists.  Read the third column
!as the starting density.  We assume that the grid of Z points is
!the same.

do n=1, nmax
 read (3, *) aaa, r(n), rho(n)
end do

close (3)
iscale=.false.

999 continue
!-------------------------------------------------------------------------------

!if (iscale.eq..true.) then
! call whole_atom_scaling(nmax, dr, rho)
!end if

!The variable iter tells us how many iterations we have to go before
!we ask the user if we should continue the calculation.
!The variable niter counts how many iterations we have done.

niter=0

!The next line is the beginning of the iteration loop.

do l=1, iter

!Call the subroutine that computes the mean-field potential for the simulation.
!The choices are 'mf_OP' (Orsay-Paris functional), 'mf-BH' (Catalonia functional), or
!mf_OT (Orsay-Trento functional).

call mf_OP(rho, u, rhobar, e_pot, nmax, imax, r, dr, nr_per_h, handle)

!Call the subroutine that computes the imaginary-time propagation step of the algorithm.

call hankel(nmax, dr, r, u, dt, psi, newpsi, chempot, atoms)

!Call the subroutine that computes the total and component energies of the droplet.

call ecalc(nmax, r, dr, newpsi, rhobar, e_pot, atoms)

!Call the subroutine that updates the vectors with the new iteration's information.

call update(nmax, r, rho, psi, newpsi, chempot, iter, maxdiff, handle)

!Increment the iteration counter by 1.

niter=niter+1

end do

!At the end of the program, deallocate the vectors used to free up memory.

deallocate(r)
deallocate(rho)
deallocate(u)
deallocate(psi)
deallocate(newpsi)
deallocate(rhobar)
deallocate(e_pot)
```

```
END PROGRAM

!-----------------------------------------------------------------------------------------
!This subroutine computes the value of rmid needed to obtain an initial density profile
!corresponding to the number of atoms desired, as given by the variable n_guess.
!-----------------------------------------------------------------------------------------
subroutine rmid_opt(dr, n_guess, rmid)
implicit real*8 (a-h, o-z)
parameter (n_rmid_opt=10000)
parameter (pi=4.0*atan(1.0))

dimension rho_rmid(n_rmid_opt), r_rmid(n_rmid_opt)

!Define the initial value of rmid to be 100 bohr. This corresponds to a large droplet, but
!can be adjusted to even larger values if necessary.

rmid=100.0d0

!The convergence tolerance is set to be 5e-12 bohr^{-3} between the desired number of atoms
!in n_guess and the computed number of atoms from the density profile.

tol=5e-12

!This parameter will be used later to increment the value of rmid.
step=0.001d0

!brho is the bulk density of helium.

brho=0.021836d0/1.889725989d0**3

!Define the radial direction vector for the subroutine.

do i=1, n_rmid_opt
 r_rmid(i)=(i-1.0)*dr
end do

!Compute the density profile according to Stringari and Treiner:

do n=1, n_rmid_opt
 rho_rmid(n)=brho/((1.0d0+exp((r_rmid(n)-rmid)/1.96d0))**2.5d0)
end do

!Calculate the integral, corresponding to the number of atoms.

rhoint1=0.0
do n=1, n_rmid_opt-1
 rhoint1=rhoint1+0.5*dr*(r_rmid(n)**2*rho_rmid(n)+r_rmid(n+1)**2*rho_rmid(n+1))
end do

!Scale by 4*pi, to analytically account for the angular integrals.

atoms=4.0d0*pi*rhoint1

!Calculate the difference between the computed and desired number of atoms.

diff=atoms-dble(n_guess)
abs_diff=abs(diff)

!Check to see if the value for rmid needs to be increased or decreased.

if (diff.gt.0) then
 dir=-1
else if (diff.lt.0) then
 dir=1
else
 stop
end if

!Increment the value of rmid by the parameter 'step' until an acceptable value is
!found.

do while (abs_diff.gt.tol)
 rmid=rmid+dir*step

do n=1, n_rmid_opt
 rho_rmid(n)=brho/((1.0d0+exp((r_rmid(n)-rmid)/1.96d0))**2.5d0)
end do

rhoint1=0.0d0
do n=1, n_rmid_opt-1
 rhoint1=rhoint1+0.5d0*dr*(r_rmid(n)**2*rho_rmid(n)+r_rmid(n+1)**2*rho_rmid(n+1))
end do

atoms=4.0*pi*rhoint1

diff=atoms-dble(n_guess)
abs_diff=abs(diff)
```

```
if (diff.gt.0.and.dir.eq.1) then
 rmid=rmid-dir*step
 step=0.1*step
else if (diff.lt.0.and.dir.eq.-1) then
 rmid=rmid-dir*step
 step=0.1d0*step
end if

!write(*,*) atoms, dble(abs_diff)
end do

write(*,*) "Total number of atoms is: ", atoms

return
end subroutine

!-------------------------------------------------------------------------
!This subroutine computes the mean-field potential according to the Catalonia
!functional.
!-------------------------------------------------------------------------
subroutine mf_BH(rho, u, rhobar, e_pot, nmax, imax, r, dr, nr_per_h, handle)

implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))

dimension rho(nmax), rhogamma(nmax), u(nmax), e_pot(nmax)
dimension r(nmax), rhobar(nmax), term1(nmax), term2(nmax)
dimension term3(nmax), term4(nmax), term5(nmax)
real*8 :: ljterm
character(len=4) :: handle

!Define some unit conversions.
!1 Hartree = 3.1577504d5 K = 27.21138505 eV
!1 Angstrom = 1.889725989 bohr

!Define the appropriate constants for the functional.
b=-8.88810e2/3.1577504e5*1.889725989d0**3
c=(1.04554e7/3.1577504e5)*1.889725989d0**(3.0d0*(2.8d0+1.0d0))
hop=2.376728d0*1.889725989d0
sig=2.556d0*1.889725989d0
ep=10.22d0/3.1577504e5
gamm4=2.8d0

!The Catalonia functional has a second cutoff distance of sig (LJ sigma), and
!so also requires another variable to account for the number of points inside
!this distance.

nr_per_sig=floor(nr_per_h*sig/hop)

!The local potential within a sphere is given by Equation 2.11 in the
!Szybisz paper, and is expressed for a planar slab of He here:

!Calculate the integral for the averaging sphere when rz is larger than the
!averaging sphere radius.

do n=1, nmax
 if (n.eq.1) then
  rhobar3=0.0d0

  do nprime=1, nr_per_h-1
   tempint3=dr/2.0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
   rhobar3=rhobar3+tempint3
  end do

  rhobar(n)=3.0/(hop**3)*rhobar3

 else if (n.ge.nr_per_h) then
  rhobar4=0.0d0
  do nprime=n-nr_per_h+1, n+nr_per_h-2
  if (nprime.gt.0.and.nprime.lt.nmax) then
    if (nprime.eq.1) then
     tempint=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
    else
     tempint=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime)**2.0d0-hop**2.0d0)
     /(2.0d0*r(n)*r(nprime)))+rho(nprime+1)*r(nprime+1)**2.0d0
     *(1.0d0-(r(n)**2.0d0+r(nprime+1)**2.0d0-hop**2.0d0)/(2.0d0*r(n)
     *r(nprime+1))))
     rhobar4=rhobar4+tempint
    end if
   end if
  end do

  rhobar(n)= 3.0d0/(2.0d0*hop**3.0d0)*rhobar4

!Otherwise, calculate the integral for the averaging sphere when rz is
!less than the radius of the averaging sphere.

 else
```

```
  rhobar1=0.0d0
  rhobar2=0.0d0
  do nprime=nr_per_h-n+1, n+nr_per_h-2
  if (nprime.gt.0.and.nprime.le.nmax) then
   tempint1=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime)**2.0d0-hop**2.0d0)
   /(2.0d0*r(n)*r(nprime)))+rho(nprime+1)*r(nprime+1)**2.0d0*(1.0d0-(r(n)**2.0d0
   +r(nprime+1)**2.0d0-hop**2.0d0)/(2.0d0*r(n)*r(nprime+1))))
   rhobar1=rhobar1+tempint1
  end if
  end do

  rhobar1=3.0d0/(2.0*hop**3.0d0)*rhobar1

  do nprime=1, nr_per_h-n
  if (nprime.gt.0.and.nprime.le.nmax) then
   tempint2=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
   rhobar2=rhobar2+tempint2
  end if
  end do

  rhobar2=3.0/(hop**3)*rhobar2
  rhobar(n)=rhobar1+rhobar2
 end if
end do

if (rhobar(1).gt.rhobar(2)) rhobar(1)=rhobar(2)

!Output the coarse-grained density profile to the file 'rhobar####.out', where #### corresponds
!to the chosen value for ihandle.

open(11,file='rhobar'//trim(handle)//'.out')
do n=1, nmax
 write(11,*) n, r(n), rhobar(n)
end do
close (11)

!The droplet system is divided into three parts: r=0, 0<r<h, and r>h.
!First, define more terms:

b_lj=33.0d0/8.0d0*(8.0d0/9.0d0*ep+b/(4.0d0*pi*sig**3))
h_lj=((sig/hop)**12-(sig/hop)**6)

!Begin the calculation of the 1-D integral over r', denoted as r(nprime).

do n=1, nmax
atot1=0.0d0
atot2=0.0d0
atot3_1=0.0d0
atot3_2=0.0d0
atot4_1=0.0d0
atot4_2=0.0d0
atot5=0.0d0
atot6=0.0d0
atot7=0.0d0
atot8_1=0.0d0
atot8_2=0.0d0
atot9=0.0d0
atot10=0.0d0

!Evaluate the n=1 case (a=att, c=rep)
 if (n.eq.1) then

  do nprime=1, nr_per_sig-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
   a1=(dr/2.0)*((1.0-(r(nprime)/sig)**8)*r(nprime)**2*rho(nprime)+(1.0-(r(nprime+1)/sig)**8)*r(nprime+1)**2
   *rho(nprime+1))
   atot1=atot1+a1
   end if
  end do

  do nprime=nr_per_sig, nmax-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
   a2=(dr/2.0)*(rho(nprime)*((sig**12/r(nprime)**10)-(sig**6/r(nprime)**4))+rho(nprime+1)*((sig**12/r(nprime+1)**10)
   -(sig**6/r(nprime+1)**4)))
   atot2=atot2+a2
   end if
  end do

 term1(n)=4.0d0*pi*b_lj*atot1+16.0d0*pi*ep*atot2
 term2(n)=0.0
 term3(n)=0.0

 !Evaluate the case when 0 < r(n) < sigma.

 else if (n.gt.1.and.n.lt.nr_per_sig) then
  ilow=nr_per_sig-n
  ihigh=nr_per_sig+n-1
  do nprime=1, ilow
```

```
   if (nprime.gt.0.and.nprime.le.nmax) then
   a3_1=(dr/2.0)*2.0*(r(nprime)**2*rho(nprime)+r(nprime+1)**2*rho(nprime+1))
   atot3_1=atot3_1+a3_1

   a3_2=(dr/2.0)*(1.0/(10.0*sig**8))*(r(nprime)*rho(nprime)/r(n)*((r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**5
   -(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**5)+r(nprime+1)*rho(nprime+1)/r(n)*((r(n)**2+r(nprime+1)**2
   +2.0*r(n)*r(nprime+1))**5-(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**5))
   atot3_2=atot3_2+a3_2
   end if
  end do

  do nprime=ilow+1, ihigh-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
    a4_1=(dr/2.0)*(r(nprime)**2*rho(nprime)*(1.0-(r(n)**2+r(nprime)**2-sig**2)/(2.0*r(n)*r(nprime)))+r(nprime+1)**2
    *rho(nprime+1)*(1.0-(r(n)**2+r(nprime+1)**2-sig**2)/(2.0*r(n)*r(nprime+1))))
   atot4_1=atot4_1+a4_1

   a4_2=(dr/2.0)*(1.0/(10.0*sig**8))*(r(nprime)*rho(nprime)/r(n)*(sig**10-(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**5)
   +r(nprime+1)*rho(nprime+1)/r(n)*(sig**10-(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**5))
   atot4_2=atot4_2+a4_2

   a5=(dr/2.0)*(r(nprime)*rho(nprime)*((sig**12/10.0)*((1.0/sig**10)-(1.0/(r(n)**2+r(nprime)**2+2.0*r(n)
   *r(nprime))**5))-(sig**6/4.0)*((1.0/sig**4)-(1.0/(r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**2)))
   +r(nprime+1)*rho(nprime+1)*((sig**12/10.0)*((1.0/sig**10)-(1.0/(r(n)**2+r(nprime+1)**2
   +2.0*r(n)*r(nprime+1))**5))-(sig**6/4.0)*((1.0/sig**4)-(1.0/(r(n)**2+r(nprime+1)**2
   +2.0*r(n)*r(nprime+1))**2))))
   atot5=atot5+a5
   end if
  end do

do nprime=ihigh, nmax-1
   if (nprime.gt.0.and.nprime.le.nmax) then
   a6=(dr/2.0)*(r(nprime)*rho(nprime)*((sig**12/10.0)*((1.0/(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**5)-
   (1.0/(r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**5))-(sig**6/4.0)*((1.0/(r(n)**2+r(nprime)**2
   -2.0*r(n)*r(nprime))**2)-(1.0/(r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)
   *((sig**12/10.0)*((1.0/(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**5)
   -(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**5))-(sig**6/4.0)
   *((1.0/(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**2)-(1.0/(r(n)**2
   +r(nprime+1)**2+2.0*r(n)*r(nprime+1))**2))))
   atot6=atot6+a6
   end if
  end do

 term1(n)=0.0
 term2(n)=2.0d0*pi*b_lj*(atot3_1-atot3_2)+2.0d0*pi*b_lj*(atot4_1-atot4_2)+8.0d0*pi*ep/r(n)*(atot5+atot6)
 term3(n)=0.0

!Evaluate the case when r(n) > sigma.

 else
  ilow=n-nr_per_sig
  ihigh=n+nr_per_sig-1
  do nprime=1, ilow
  if (nprime.gt.0.and.nprime.lt.nmax) then
   a7=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**5)
   -(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**5))-(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime)**2
   -2.0d0*r(n)*r(nprime))**2)-(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))
   +r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**5)
   -(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*
   ((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**2)-(1.0d0/(r(n)**2
   +r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
   atot7=atot7+a7
   end if
  end do

  do nprime=ilow+1, ihigh-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
    if (nprime.eq.1) then
    a8_1=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/5.0d0)*((1.0d0/sig**10)-(1.0d0/(r(n)**2+r(nprime)**2
    +2.0d0*r(n)*r(nprime))**5))-(sig**6/2.0d0)*((1.0d0/sig**4)-(1.0d0/(r(n)**2+r(nprime)**2
    +2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/5.0d0)*((1.0d0/sig**10)-(1.0d0/(r(n)**2
    +r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/2.0d0)*((1.0d0/sig**4)
    -(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
    else
    a8_1=(dr/2.0)*(r(nprime)**2*rho(nprime)*(1.0-(r(n)**2+r(nprime)**2-sig**2)/(2.0*r(n)*r(nprime)))
    +r(nprime+1)**2*rho(nprime+1)*(1.0-(r(n)**2+r(nprime+1)**2-sig**2)/(2.0*r(n)*r(nprime+1))))
    end if
   atot8_1=atot8_1+a8_1

   a8_2=(dr/2.0)*(1.0/(10.0*sig**8))*(r(nprime)*rho(nprime)/r(n)*(sig**10-(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**5)
   +r(nprime+1)*rho(nprime+1)/r(n)*(sig**10-(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**5))
   atot8_2=atot8_2+a8_2

   a9=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/sig**10)-(1.0d0/(r(n)**2+r(nprime)**2
   +2.0d0*r(n)*r(nprime))**5))-(sig**6/4.0d0)*((1.0d0/sig**4)-(1.0d0/(r(n)**2+r(nprime)**2
   +2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)*
   ((1.0d0/sig**10)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*
   ((1.0d0/sig**4)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
```

```
   atot9=atot9+a9
   end if
  end do

  do nprime=ihigh, nmax-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
   a10=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**5)
   -(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**5))-(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime)**2
   -2.0d0*r(n)*r(nprime))**2)-(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))
   +r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**5)
   -(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*((1.0d0/(r(n)**2
   +r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**2)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
   atot10=atot10+a10
   end if
  end do

 term1(n)=0.0
 term2(n)=0.0
 term3(n)=8.0d0*pi*ep/r(n)*(atot7+atot9+atot10)+2.0d0*pi*b_lj*(atot8_1-atot8_2)
 end if
end do

!Calculate the first c-term.

do n=1, nmax
 term4(n)=c/2.0d0*rhobar(n)**(gamm4+1.0d0)
end do

!Calculate the second c-term.

do n=1, nmax

ctot1=0.0d0
ctot2=0.0d0
ctot3=0.0d0
ctot4=0.0d0

 if (n.eq.1) then
  do nprime=1, nr_per_h-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
    c1=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4)
    ctot1=ctot1+c1
   end if
  end do
 else if (n.gt.1.and.n.lt.nr_per_h) then
  ilow=nr_per_h-n
  ihigh=nr_per_h+n-1
  do nprime=1, ilow
   c2=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4)
   ctot2=ctot2+c2
  end do
  do nprime=ilow+1, ihigh-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
    c3=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4*(1.0-(r(n)**2+r(nprime)**2-hop**2)
    /(2.0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4*
    (1.0-(r(n)**2+r(nprime+1)**2-hop**2)/(2.0*r(n)*r(nprime+1))))
    ctot3=ctot3+c3
   end if
  end do
 else
  ilow=n-nr_per_h
  ihigh=n+nr_per_h-1
  do nprime=ilow+1, ihigh-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
    if (nprime.eq.1) then
     c4=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4*rho(nprime)*r(nprime)**2+r(nprime+1)**2
     *rho(nprime+1)*rhobar(nprime+1)**gamm4*rho(nprime+1)*r(nprime+1)**2)
    else
     c4=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4*(1.0d0-(r(n)**2+r(nprime)**2-hop**2)
     /(2.0d0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4*
     (1.0d0-(r(n)**2+r(nprime+1)**2-hop**2)/(2.0d0*r(n)*r(nprime+1))))
    end if
   ctot4=ctot4+c4
   end if
  end do
 end if
term5(n)=3.0d0*c/(2.0d0*hop**3)*(1.0d0+gamm4)*(ctot1+ctot2)+3.0d0*c/(4.0d0*hop**3)*(1.0d0+gamm4)*(ctot3+ctot4)
end do

!Write the terms of the mean-field calculation to a file called "terms####.out", where #### is
!the value of ihandle.

open(9, file='terms'//trim(handle)//'.out')
do n=1, nmax
 write(9,9000) n, r(n), term1(n), term2(n), term3(n), term4(n), term5(n)
! write(9,9000) n, r(n), thing1(n), thing2(n), thing3(n), thing4(n), term3(n)
end do
close (9)
```

```
9000 format (i6, 6(1x, 1pe12.5))

!Define the total mean field.

do n=1, nmax
 u(n)=term1(n)+term2(n)+term3(n)+term4(n)+term5(n)
 e_pot(n)=term1(n)+term2(n)+term3(n)
end do

call something(nmax, r, u, tot)
u(1)=tot
call something(nmax, r, e_pot, tot)
e_pot(1)=tot

!Write the mean-field function to a file called "u####.out", where #### is
!the value of ihandle.
open (10, file='u'//trim(handle)//'.out')
do n=1, nmax
 write(10,*) n, r(n), u(n)
end do
close (10)

return
end subroutine

!-------------------------------------------------------------------------------------------
!This subroutine calculates the mean-field potential according to the Orsay-Paris functional.
!-------------------------------------------------------------------------------------------
subroutine mf_OP(rho, u, rhobar, e_pot, nmax, imax, r, dr, nr_per_h, handle)

implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))

dimension rho(nmax), rhogamma(nmax), u(nmax), e_pot(nmax)
dimension r(nmax), rhobar(nmax), term1(nmax), term2(nmax)
dimension term3(nmax), term4(nmax), term5(nmax)
dimension thing1(nmax), thing2(nmax), thing3(nmax), thing4(nmax)
real*8 :: ljterm
character(len=4) :: handle


!Define some convenient unit conversions.
!1 Hartree = 3.1577504d5 K = 27.21138505 eV
!1 Angstrom = 1.889725989 bohr

!Define the constants appropriate to this functional.

b=-8.88810e2/3.1577504e5*1.889725989d0**3
c=(1.04554e7/3.1577504e5)*1.889725989d0**(3.0d0*(2.8d0+1.0d0))
hop=2.376728d0*1.889725989d0
sig=2.556d0*1.889725989d0
ep=10.22d0/3.1577504e5
gamm4=2.8d0

!The local potential within a sphere is given by Equation 2.11 in the
!Szybisz paper, and is expressed for a planar slab of He here:

!Calculate the integral for the averaging sphere when r is larger than the
!averaging sphere radius.

do n=1, nmax
 if (n.eq.1) then
  rhobar3=0.0d0

  do nprime=1, nr_per_h-1
   tempint3=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
   rhobar3=rhobar3+tempint3
  end do

  rhobar(n)=3.0/(hop**3)*rhobar3

 else if (n.ge.nr_per_h) then
  rhobar4=0.0d0
  do nprime=n-nr_per_h+1, n+nr_per_h-2
   if (nprime.gt.0.and.nprime.lt.nmax) then
    if (nprime.eq.1) then
     tempint=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
    else
     tempint=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime)**2.0d0-hop**2.0d0)
     /(2.0d0*r(n)*r(nprime)))+rho(nprime+1)*r(nprime+1)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime+1)**2.0d0
     -hop**2.0d0)/(2.0d0*r(n)*r(nprime+1))))
    end if
    rhobar4=rhobar4+tempint
   end if
  end do

  rhobar(n)= 3.0d0/(2.0d0*hop**3.0d0)*rhobar4
```

```
!Otherwise, calculate the integral for the averaging sphere when r is
!less than the radius of the averaging sphere.

 else
  rhobar1=0.0d0
  rhobar2=0.0d0
  do nprime=nr_per_h-n+1, n+nr_per_h-2 !Are you sure about the end limit?
  if (nprime.gt.0.and.nprime.le.nmax) then !why is this 'le' when above it is
                                          !'lt'
   tempint1=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime)**2.0d0-hop**2.0d0)/
   (2.0d0*r(n)*r(nprime)))+rho(nprime+1)*r(nprime+1)**2.0d0*(1.0d0-(r(n)**2.0d0
   +r(nprime+1)**2.0d0-hop**2.0d0)/(2.0d0*r(n)*r(nprime+1))))
   rhobar1=rhobar1+tempint1
  end if
  end do

  rhobar1=3.0d0/(2.0*hop**3.0d0)*rhobar1

  do nprime=1, nr_per_h-n
  if (nprime.gt.0.and.nprime.le.nmax) then
   tempint2=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
   rhobar2=rhobar2+tempint2
  end if
  end do

  rhobar2=3.0d0/(hop**3.0d0)*rhobar2
  rhobar(n)=rhobar1+rhobar2
 end if
end do

if (rhobar(1).gt.rhobar(2)) rhobar(1)=rhobar(2)

!Record the coarse-grained density profile in the file "rhobar####.out", where #### is the chosen
!value for ihandle.

open(11,file='rhobar'//trim(handle)//'.out')
do n=1, nmax
 write(11,*) n, r(n), rhobar(n)
end do
close (11)

!The droplet system is divided into three parts: r=0, 0<r<h, and r>h.

h_lj=((sig/hop)**12.0d0-(sig/hop)**6.0d0)

!Begin the integration over r', denoted as r(nprime).

do n=1, nmax
atot1=0.0d0
atot2=0.0d0
atot3=0.0d0
atot4=0.0d0
atot5=0.0d0
atot6=0.0d0
atot7=0.0d0
atot8=0.0d0
atot9=0.0d0
atot10=0.0d0

ctot1=0.0d0
ctot2=0.0d0
ctot3=0.0d0
ctot4=0.0d0

!Evaluate the n=1 case (a=att, c=rep)
 if (n.eq.1) then

  do nprime=1, nr_per_h-1
!   if (nprime.gt.0.and.nprime.lt.nmax) then
   a1=(dr/2.0d0)*(r(nprime)**6*rho(nprime)+r(nprime+1)**6*rho(nprime+1))
   atot1=atot1+a1

   c1=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)
   **gamm4)
   ctot1=ctot1+c1
!   end if
  end do

  do nprime=nr_per_h, nmax-1
!   if (nprime.gt.0.and.nprime.lt.nmax) then !do you need statement after and?
   a2=(dr/2.0d0)*(rho(nprime)*((sig**12/r(nprime)**10)-(sig**6/r(nprime)**4))+rho(nprime+1)
   *((sig**12/r(nprime+1)**10)-(sig**6/r(nprime+1)**4)))
   atot2=atot2+a2
!   end if
  end do

 term1(n)=(16.0d0*pi*ep/hop**4)*h_lj*atot1+16.0d0*pi*ep*atot2
```

```
 term2(n)=0.0
 term3(n)=0.0
 else if (n.gt.1.and.n.lt.nr_per_h) then
  ilow=nr_per_h-n
  ihigh=nr_per_h+n-1
  do nprime=1, ilow
   if (ilow.gt.0) then !do you need first statement?
   a3=(dr/2.0)*(r(nprime)*rho(nprime)*((r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**3-(r(n)**2+r(nprime)**2
   -2.0*r(n)*r(nprime))**3)+r(nprime+1)*rho(nprime+1)*((r(n)**2+r(nprime+1)**2
   +2.0*r(n)*r(nprime+1))**3-(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**3))
   atot3=atot3+a3

   c2=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4)
   ctot2=ctot2+c2
   end if
  end do

  !Evaluate the case where 0 < r < h.

  do nprime=ilow+1, ihigh-1
   if (nprime.lt.nmax) then
   a4=(dr/2.0)*(r(nprime)*rho(nprime)*(hop**6-(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**3)+r(nprime+1)
   *rho(nprime+1)*(hop**6-(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**3))
   atot4=atot4+a4

   a5=(dr/2.0)*(r(nprime)*rho(nprime)*((sig**12/10.0)*((1.0/hop**10)-(1.0/(r(n)**2+r(nprime)**2
   +2.0*r(n)*r(nprime))**5))-(sig**6/4.0)*((1.0/hop**4)-(1.0/(r(n)**2+r(nprime)**2
   +2.0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0)*((1.0/hop**10)
   -(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**5))-(sig**6/4.0)*((1.0/hop**4)
   -(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**2))))
   atot5=atot5+a5

   c3=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4*(1.0-(r(n)**2+r(nprime)**2-hop**2)
   /(2.0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4*
   (1.0-(r(n)**2+r(nprime+1)**2-hop**2)/(2.0*r(n)*r(nprime+1))))
   ctot3=ctot3+c3
   end if
  end do

  do nprime=ihigh, nmax-1
   a6=(dr/2.0)*(r(nprime)*rho(nprime)*((sig**12/10.0)*((1.0/(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**5)
   -(1.0/(r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**5))-(sig**6/4.0)*((1.0/(r(n)**2
   +r(nprime)**2-2.0*r(n)*r(nprime))**2)-(1.0/(r(n)**2+r(nprime)**2
   +2.0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0)*((1.0/(r(n)**2+r(nprime+1)**2
   -2.0*r(n)*r(nprime+1))**5)-(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**5))
   -(sig**6/4.0)*((1.0/(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**2)-(1.0/(r(n)**2+r(nprime+1)**2
   +2.0*r(n)*r(nprime+1))**2))))
   atot6=atot6+a6
  end do

 term1(n)=0.0
 term2(n)=4.0d0*pi*ep*h_lj/(3.0d0*hop**4*r(n))*(atot3+atot4)+8.0d0*pi*ep/r(n)*(atot5+atot6)
 term3(n)=0.0

!Evaluate the case where r > h.
 else
  ilow=n-nr_per_h
  ihigh=n+nr_per_h-1
  do nprime=1, ilow
  if (ilow.gt.0) then
   a7=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime)**2
   -2.0d0*r(n)*r(nprime))**5)-(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**5))
   -(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**2)-(1.0d0/(r(n)**2
   +r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)
   *((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**5)-(1.0d0/(r(n)**2
   +r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime+1)**2
   -2.0d0*r(n)*r(nprime+1))**2)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
   atot7=atot7+a7
   end if
  end do

  do nprime=ilow+1, ihigh-1
   if (nprime.lt.nmax) then
   a8=(dr/2.0d0)*(r(nprime)*rho(nprime)*(hop**6-(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**3)
   +r(nprime+1)*rho(nprime+1)*(hop**6-(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**3))
   atot8=atot8+a8

   a9=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/hop**10)-(1.0d0/(r(n)**2+r(nprime)**2
   +2.0d0*r(n)*r(nprime))**5))-(sig**6/4.0d0)*((1.0d0/hop**4)-(1.0d0/(r(n)**2+r(nprime)**2
   +2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)*((1.0d0/hop**10)
   -(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*((1.0d0/hop**4)
   -(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
   atot9=atot9+a9

   if (nprime.eq.1) then
    c4=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4*rho(nprime)*r(nprime)**2+r(nprime+1)**2
    *rho(nprime+1)*rhobar(nprime+1)**gamm4*rho(nprime+1)*r(nprime+1)**2)
   else
```

```
    c4=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**gamm4*(1.0d0-(r(n)**2+r(nprime)**2-hop**2)
    /(2.0d0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**gamm4*
    (1.0d0-(r(n)**2+r(nprime+1)**2-hop**2)/(2.0d0*r(n)*r(nprime+1))))
    end if
   ctot4=ctot4+c4
   end if
  end do

  do nprime=ihigh, nmax-1
!   if (nprime.gt.0.and.nprime.lt.nmax) then
   a10=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime)**2
   -2.0d0*r(n)*r(nprime))**5)-(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**5))
   -(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**2)
   -(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)
   *((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**5)
   -(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)
   *((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**2)-(1.0d0/(r(n)**2+r(nprime+1)**2
   +2.0d0*r(n)*r(nprime+1))**2))))
   atot10=atot10+a10
!   end if
  end do

 term1(n)=0.0
 term2(n)=0.0
 term3(n)=8.0*pi*ep/r(n)*(atot7+atot9+atot10)+4.0*pi*ep*h_lj/(3.0*hop**4*r(n))*atot8
 thing1(n)=8.0*pi*ep/r(n)*atot7
 thing2(n)=4.0*pi*ep*h_lj/(3.0*hop**4*r(n))*atot8
 thing3(n)=8.0*pi*ep/r(n)*atot9
 thing4(n)=8.0*pi*ep/r(n)*atot10
 end if
 term5(n)=3.0d0*c/(2.0d0*hop**3)*(1.0d0+gamm4)*(ctot1+ctot2)+3.0d0*c/(4.0d0*hop**3)*(1.0d0+gamm4)*(ctot3+ctot4)
end do

!Compute the first c-term.

do n=1, nmax
 term4(n)=c/2.0d0*rhobar(n)**(gamm4+1.0d0)
end do

!Store the terms of the mean field in a file called "terms####.out", where #### is the choice of number
!for ihandle.

open(9, file='terms'//trim(handle)//'.out')
do n=1, nmax
 write(9,9000) n, r(n), term1(n), term2(n), term3(n), term4(n), term5(n)
! write(9,9000) n, r(n), thing1(n), thing2(n), thing3(n), thing4(n), term3(n)
end do
close (9)

9000 format (i6, 6(1x, 1pe12.5))

!Compute the mean field.

do n=2, nmax
 u(n)=term1(n)+term2(n)+term3(n)+term4(n)+term5(n)
 e_pot(n)=term1(n)+term2(n)+term3(n)
end do

call something(nmax, r, u, tot)
u(1)=tot
call something(nmax, r, e_pot, tot)
e_pot(1)=tot

!Record the mean field in a file called "u####.out", where #### is the value of ihandle.
open (10, file='u'//trim(handle)//'.out')
do n=1, nmax
 write(10,*) n, r(n), u(n)
end do
close (10)

return
end subroutine

!-------------------------------------------------------------------------------
!This subroutine computes the mean field according to the Orsay-Trento functional
!-------------------------------------------------------------------------------
subroutine mf_OT(rho, u, rhobar, e_pot, nmax, imax, r, dr, nr_per_h, handle)

implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))
parameter (hbar=1.0d0)

dimension rho(nmax), rhogamma(nmax), u(nmax), e_pot(nmax)
dimension r(nmax), rhobar(nmax), term1(nmax), term2(nmax)
dimension term3(nmax), term4(nmax), term5(nmax)
dimension del_rho(nmax),  e_gg(nmax)
real*8 :: ljterm
character(len=4) :: handle
```

```
!Define unit conversions.
!1 Hartree = 3.1577504d5 K = 27.21138505 eV
!1 Angstrom = 1.889725989 bohr

!Define constants appropriate to the functional.

b=-7.18990e2/3.1577504e5*1.889725989d0**3
cprime=(-2.411857e4/3.1577504e5)*1.889725989d0**6
cdprime=(1.858496e6/3.1577504e5)*1.889725989d0**9
hot=2.190323d0*1.889725989d0
sig=2.556d0*1.889725989d0
ep=10.22d0/3.1577504e5
alpha=54.31*1.889725989d0**3
rho_os=0.04/1.889725989d0**3
cc=7296.29

!Compute the gradient of the density.

do n=2, nmax-1
 del_rho(n)=(rho(n+1)-rho(n-1))/(2.0d0*dr)
end do
del_rho(1)=(rho(2)-rho(1))/dr
del_rho(nmax)=(rho(nmax)-rho(nmax-1))/dr

!Compute the coarse-grained density.

do n=1, nmax

!Calculate the integral for the averaging sphere when r is
!greater than the radius of the averaging sphere.

 if (n.eq.1) then
  rhobar3=0.0d0

  do nprime=1, nr_per_h-1
   tempint3=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
   rhobar3=rhobar3+tempint3
  end do

  rhobar(n)=3.0/(hot**3)*rhobar3

 else if (n.ge.nr_per_h) then
  rhobar4=0.0d0
  do nprime=n-nr_per_h+1, n+nr_per_h-2
   if (nprime.gt.0.and.nprime.lt.nmax) then
    if (nprime.eq.1) then
     tempint=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
    else
     tempint=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime)**2.0d0-hot**2.0d0)
     /(2.0d0*r(n)*r(nprime)))+rho(nprime+1)*r(nprime+1)**2.0d0*(1.0d0-(r(n)**2.0d0
     +r(nprime+1)**2.0d0-hot**2.0d0)/(2.0d0*r(n)*r(nprime+1))))
    end if
    rhobar4=rhobar4+tempint
   end if
  end do

  rhobar(n)= 3.0d0/(2.0d0*hot**3.0d0)*rhobar4

!Otherwise, calculate the integral for the averaging sphere when r is
!less than the radius of the averaging sphere.

 else
  rhobar1=0.0d0
  rhobar2=0.0d0
  do nprime=nr_per_h-n+1, n+nr_per_h-2
  if (nprime.gt.0.and.nprime.le.nmax) then

   tempint1=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0*(1.0d0-(r(n)**2.0d0+r(nprime)**2.0d0-hot**2.0d0)
   /(2.0d0*r(n)*r(nprime)))+rho(nprime+1)*r(nprime+1)**2.0d0*(1.0d0-(r(n)**2.0d0
   +r(nprime+1)**2.0d0-hot**2.0d0)/(2.0d0*r(n)*r(nprime+1))))
   rhobar1=rhobar1+tempint1
  end if
  end do

  rhobar1=3.0d0/(2.0*hot**3.0d0)*rhobar1

  do nprime=1, nr_per_h-n
  if (nprime.gt.0.and.nprime.le.nmax) then
   tempint2=dr/2.0d0*(rho(nprime)*r(nprime)**2.0d0+rho(nprime+1)*r(nprime+1)**2.0d0)
   rhobar2=rhobar2+tempint2
  end if
  end do

  rhobar2=3.0d0/(hot**3.0d0)*rhobar2
  rhobar(n)=rhobar1+rhobar2
 end if
end do
```

```
if (rhobar(1).gt.rhobar(2)) rhobar(1)=rhobar(2)

!Store the coarse-grained density in a file called "rhobar####.out", where #### is the
!chosen value of ihandle.

open(11,file='rhobar'//trim(handle)//'.out')
do n=1, nmax
 write(11,*) n, r(n), rhobar(n)
end do
close (11)

!Compute the kinetic energy correction term, e_gg.

do n=2, nmax
 etot1=0.0d0
 do nprime=1, nmax-1

 etot1=etot1+dr/2.0d0*(del_rho(nprime)*(1.0d0-rho(nprime)/rho_os)*r(nprime)*((1.0d0+(1.0d0
 +2.0d0*r(n)*r(nprime))/(2.0d0*r(n)**2))*exp(-(r(n)**2+r(nprime)**2
 +2.0d0*r(n)*r(nprime)))-(1.0d0+(1.0d0-2.0d0*r(n)*r(nprime))/(2.0d0*r(n)**2))
 *exp(-(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))))+del_rho(nprime+1)*(1.0d0-rho(nprime+1)
 /rho_os)*r(nprime+1)*((1.0d0+(1.0d0+2.0d0*r(n)*r(nprime+1))/(2.0d0*r(n)**2))
 *exp(-(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1)))-(1.0d0
 +(1.0d0-2.0d0*r(n)*r(nprime+1))/(2.0d0*r(n)**2))*exp(-(r(n)**2+r(nprime+1)**2
 -2.0d0*r(n)*r(nprime+1)))))
 end do
 e_gg(n)=hbar**2*alpha/(cc*sqrt(pi))*(1.0d0-rho(n)/rho_os)*etot1
end do

call something(nmax, r, e_gg, tot)
e_gg(1)=tot

!Store this term in a file called "e_gg####.out", where #### is the chosen value for ihandle.

open(12, file='e_gg'//trim(handle)//'.out')
do n=1, nmax
 write(12,*) n, r(n), e_gg(n)
end do
close (12)

!Compute the integral over r', denoted as r(nprime).

do n=1, nmax
atot1=0.0d0
atot2=0.0d0
atot3=0.0d0
atot4=0.0d0
atot5=0.0d0
atot6=0.0d0
atot7=0.0d0
atot8=0.0d0
atot9=0.0d0
atot10=0.0d0

ctot1=0.0d0
ctot2=0.0d0
ctot3=0.0d0
ctot4=0.0d0
ctot5=0.0d0
ctot6=0.0d0
ctot7=0.0d0
ctot8=0.0d0

!Evaluate the n=1 case (a=att, c=rep)
 if (n.eq.1) then

  do nprime=1, nr_per_h-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
   atot1=0.0

   c1=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1))
   ctot1=ctot1+c1

   c2=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**2+r(nprime+1)**2*rho(nprime+1)
   *rhobar(nprime+1)**2)
   ctot2=ctot2+c2
   end if
  end do

  do nprime=nr_per_h, nmax-1
   if (nprime.gt.0.and.nprime.lt.nmax) then
   a2=(dr/2.0d0)*(rho(nprime)*((sig**12/r(nprime)**10)-(sig**6/r(nprime)**4))+rho(nprime+1)
   *((sig**12/r(nprime+1)**10)-(sig**6/r(nprime+1)**4)))
   atot2=atot2+a2
   end if
  end do
```

```
 term1(n)=atot1+16.0d0*pi*ep*atot2
 term2(n)=0.0
 term3(n)=0.0

!Evaluate the case where 0 < r < h.

 else if (n.gt.1.and.n.lt.nr_per_h) then
  ilow=nr_per_h-n
  ihigh=nr_per_h+n-1
  do nprime=1, ilow
   if (ilow.gt.0) then
   atot3=0.0d0

   c3=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1))
   ctot3=ctot3+c3

   c4=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**2+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**2)
   ctot4=ctot4+c4
   end if
  end do

  do nprime=ilow+1, ihigh-1
   if (nprime.lt.nmax) then
   atot4=0.0d0

a5=(dr/2.0)*(r(nprime)*rho(nprime)*((sig**12/10.0)*((1.0/hot**10)-(1.0/(r(n)**2+r(nprime)**2+
   2.0*r(n)*r(nprime))**5))-(sig**6/4.0)*((1.0/hot**4)-(1.0/(r(n)**2+r(nprime)**2
   +2.0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0)*((1.0/hot**10)
   -(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**5))-(sig**6/4.0)*((1.0/hot**4)
   -(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**2))))
   atot5=atot5+a5

   c5=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)*(1.0-(r(n)**2+r(nprime)**2-hot**2)
   /(2.0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)*(1.0-(r(n)**2
   +r(nprime+1)**2-hot**2)/(2.0*r(n)*r(nprime+1))))
   ctot5=ctot5+c5

   c6=(dr/2.0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**2*(1.0-(r(n)**2+r(nprime)**2-hot**2)
   /(2.0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**2*(1.0
   -(r(n)**2+r(nprime+1)**2-hot**2)/(2.0*r(n)*r(nprime+1))))
   ctot6=ctot6+c6
   end if
  end do

  do nprime=ihigh, nmax-1
   a6=(dr/2.0)*(r(nprime)*rho(nprime)*((sig**12/10.0)*((1.0/(r(n)**2+r(nprime)**2-2.0*r(n)*r(nprime))**5)
   -(1.0/(r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**5))-(sig**6/4.0)*((1.0/(r(n)**2+r(nprime)**2
   -2.0*r(n)*r(nprime))**2)-(1.0/(r(n)**2+r(nprime)**2+2.0*r(n)*r(nprime))**2)))
   +r(nprime+1)*rho(nprime+1)*((sig**12/10.0)*((1.0/(r(n)**2+r(nprime+1)**2-2.0*r(n)*r(nprime+1))**5)
   -(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**5))-(sig**6/4.0)*((1.0/(r(n)**2
   +r(nprime+1)**2-2.0*r(n)*r(nprime+1))**2)-(1.0/(r(n)**2+r(nprime+1)**2+2.0*r(n)*r(nprime+1))**2))))
   atot6=atot6+a6
  end do

 term1(n)=0.0
 term2(n)=atot3+atot4+8.0d0*pi*ep/r(n)*(atot5+atot6)
 term3(n)=0.0

!Evaluate the case where r > h.

 else
  ilow=n-nr_per_h
  ihigh=n+nr_per_h-1
  do nprime=1, ilow
  if (ilow.gt.0) then
   a7=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime)**2
   -2.0d0*r(n)*r(nprime))**5)-(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**5))
   -(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**2)-(1.0d0/(r(n)**2
   +r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)
   *((1.0d0/(r(n)**2+r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**5)-(1.0d0/(r(n)**2
   +r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*((1.0d0/(r(n)**2
   +r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**2)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)
   *r(nprime+1))**2))))
   atot7=atot7+a7
   end if
  end do

  do nprime=ilow+1, ihigh-1
   if (nprime.lt.nmax) then
   atot8=0.0d0

a9=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/hot**10)-(1.0d0/(r(n)**2+r(nprime)**2
   +2.0d0*r(n)*r(nprime))**5))-(sig**6/4.0d0)*((1.0d0/hot**4)-(1.0d0/(r(n)**2
   +r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)
   *((1.0d0/hot**10)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**5))
   -(sig**6/4.0d0)*((1.0d0/hot**4)-(1.0d0/(r(n)**2+r(nprime+1)**2+2.0d0*r(n)*r(nprime+1))**2))))
   atot9=atot9+a9
```

```
   if (nprime.eq.1) then
    c7=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)*rho(nprime)*r(nprime)**2+r(nprime+1)**2
    *rho(nprime+1)*rhobar(nprime+1)*rho(nprime+1)*r(nprime+1)**2)

    c8=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**2*rho(nprime)*r(nprime)**2+r(nprime+1)**2
    *rho(nprime+1)*rhobar(nprime+1)**2*rho(nprime+1)*r(nprime+1)**2)
   else
    c7=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)*(1.0d0-(r(n)**2+r(nprime)**2-hot**2)
    /(2.0d0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)*(1.0d0
    -(r(n)**2+r(nprime+1)**2-hot**2)/(2.0d0*r(n)*r(nprime+1))))

    c8=(dr/2.0d0)*(r(nprime)**2*rho(nprime)*rhobar(nprime)**2*(1.0d0-(r(n)**2+r(nprime)**2-hot**2)
    /(2.0d0*r(n)*r(nprime)))+r(nprime+1)**2*rho(nprime+1)*rhobar(nprime+1)**2*(1.0d0
    -(r(n)**2+r(nprime+1)**2-hot**2)/(2.0d0*r(n)*r(nprime+1))))
    end if
   ctot7=ctot7+c7
   ctot8=ctot8+c8
   end if
  end do

  do nprime=ihigh, nmax-1
!   if (nprime.gt.0.and.nprime.lt.nmax) then
   a10=(dr/2.0d0)*(r(nprime)*rho(nprime)*((sig**12/10.0d0)*((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)
   *r(nprime))**5)-(1.0d0/(r(n)**2+r(nprime)**2+2.0d0*r(n)*r(nprime))**5))-(sig**6/4.0d0)
   *((1.0d0/(r(n)**2+r(nprime)**2-2.0d0*r(n)*r(nprime))**2)-(1.0d0/(r(n)**2
   +r(nprime)**2+2.0d0*r(n)*r(nprime))**2)))+r(nprime+1)*rho(nprime+1)*((sig**12/10.0d0)*((1.0d0/(r(n)**2
   +r(nprime+1)**2-2.0d0*r(n)*r(nprime+1))**5)-(1.0d0/(r(n)**2+r(nprime+1)**2
   +2.0d0*r(n)*r(nprime+1))**5))-(sig**6/4.0d0)*((1.0d0/(r(n)**2+r(nprime+1)**2
   -2.0d0*r(n)*r(nprime+1))**2)-(1.0d0/(r(n)**2+r(nprime+1)**2
   +2.0d0*r(n)*r(nprime+1))**2))))
   atot10=atot10+a10
!   end if
  end do

 term1(n)=0.0
 term2(n)=0.0
 term3(n)=8.0*pi*ep/r(n)*(atot7+atot9+atot10)+atot8
 end if
 term5(n)=3.0d0*cprime/(hot**3)*(ctot1+ctot3)+3.0d0*cdprime/(hot**3)*(ctot2+ctot4)+3.0d0*cprime
 /(2.0d0*hot**3)*(ctot5+ctot7)+3.0d0*cdprime/(2.0d0*hot**3)*(ctot6+ctot8)
end do

!Compute the two c-terms.

do n=1, nmax
 term4(n)=cprime/2.0d0*rhobar(n)**2+cdprime/3.0d0*rhobar(n)**3
end do

!Store the components of the mean field in a file called "terms####.out", where #### is the value
!associated with ihandle.

open(9, file='terms'//trim(handle)//'.out')
do n=1, nmax
 write(9,9000) n, r(n), term1(n), term2(n), term3(n), term4(n), term5(n)
! write(9,9000) n, r(n), thing1(n), thing2(n), thing3(n), thing4(n), term3(n)
end do
close (9)

9000 format (i6, 6(1x, 1pe12.5))

!Compute the mean field.

do n=2, nmax
 u(n)=term1(n)+term2(n)+term3(n)+term4(n)+term5(n)+e_gg(n)
 e_pot(n)=term1(n)+term2(n)+term3(n)
end do

call something(nmax, r, u, tot)
u(1)=tot
call something(nmax, r, e_pot, tot)
e_pot(1)=tot

!Store the mean field in a file called "u####.out", where #### is the number chosen for ihandle.

open (10, file='u'//trim(handle)//'.out')
do n=1, nmax
 write(10,*) n, r(n), u(n)
end do
close (10)

return
end subroutine

!------------------------------------------------------------------------------
!This subroutine calculates the chemical potential of a planar He-4 system using
!the ITP method with a HT to convert to momentum space.
!------------------------------------------------------------------------------
```

```
subroutine hankel(nmax, dr, r, u, dt, psi, newpsi, chempot, atoms)
implicit real*8 (a-h, o-z)
real*8, dimension(1:nmax) :: r, k, psi, temppsi, temppsi2, newpsi
real*8, dimension(1:nmax) :: oldrho, newrho, chemrho, u, ft
real*8, dimension(1:nmax, 1:nmax) :: F
real*8, parameter :: pi=4.0*atan(1.0)
real*8, parameter :: hbar=1.0

!This is M = the mass of He-4 in atomic units.
cc=7296.29

!Generate the momentum array for the FFT.

dk=pi/(nmax*dr)
do n=1, nmax
 k(n)=(n)*dk
end do

do n=1, nmax
do i=1, nmax
 F(n,i)= sqrt(2.0/nmax)*sin(k(n)*r(i))
end do
end do

!Compute the first half of the potential energy in the ITP formalism.

open(200, file='1')
do n=1, nmax
 temppsi(n)=exp(-u(n)*dt/(2.0*hbar))*psi(n)
 write(200,*) n, r(n), temppsi(n)
end do
close(200)

!Hankel Transform into momentum space and compute the kinetic energy.

do n=1, nmax
 tot=0.0
 do i=1, nmax
  tot=tot+F(n,i)*temppsi(i)
 end do
 ft(n)=exp(-k(n)*k(n)*dt/(2.0*cc))*tot
end do

!Convert back to position space.

do n=1, nmax
 tot2=0.0
 do i=1, nmax
  tot2=tot2+F(i,n)*ft(i)
 end do
 temppsi2(n)=tot2
end do

!Compute the second half of the potential energy, and square to generate
!the new density.

do n=1, nmax
 newpsi(n)=exp(-0.5*u(n)*dt)*temppsi2(n)
 newrho(n)=newpsi(n)*newpsi(n)
 oldrho(n)=psi(n)*psi(n)
end do

!Use trapezoid rule integration to compute the density integral.

rhoint1=0.0
rhoint2=0.0
rhoint3=0.0
do n=1, nmax-1
 rhoint2=rhoint2+0.5*dr*(newrho(n)+newrho(n+1))
 rhoint1=rhoint1+0.5*dr*(oldrho(n)+oldrho(n+1))
 rhoint3=rhoint3+0.5*dr*(newpsi(n)*psi(n)+newpsi(n+1)*psi(n+1))
end do

!Store the new helium wavefunction in a file called 'itppsi.out'.

open(500, file='itppsi.out')
do n=1, nmax
 newpsi(n)=(newpsi(n)/sqrt(rhoint2))*sqrt(rhoint1)
 write(500,1000) n, r(n), newpsi(n)
end do
close (500)

1000 format (i6, 3(1x, 1pe12.5))

!The eigenvalue to the corresponding eigenvector (psi) is the chemical potential of the system.

chempot=-log(rhoint3/rhoint1)*hbar/(dt)
atoms=4.0*pi*rhoint1
```

```
write(*,*) "Chemical potential is: ", chempot*3.1577504d5
write(*,*) "Total number of atoms is: ", atoms

return
end subroutine

!-----------------------------------------------------------------------------------------
!This subroutine computes the energy and energetic components for the droplet system from
!the helium wavefunction generated in the subroutine hankel.
!-----------------------------------------------------------------------------------------
subroutine ecalc(nmax, r, dr, newpsi, rhobar, e_pot, atoms)
implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))
real*8, parameter :: hbar=1.0

dimension r(nmax)
dimension rho(nmax), rhobar(nmax), del_rho(nmax), e_pot(nmax)
real*8, dimension(nmax) :: newpsi, temppsi
character(len=4) :: handle

!This is the mass of He-4 in atomic units.
cc=7296.29

!These are parameters for the functional.
c=(1.04554e7/3.1577504e5)*1.889725989d0**(3.0d0*(2.8d0+1.0d0))
gamm4=2.8d0

!Compute the new density profile.

do n=2, nmax
 rho(n)=newpsi(n)**2.0d0/r(n)**2
 temppsi(n)=newpsi(n)/r(n)
end do

call something(nmax, r, rho, tot)
rho(1)=tot
call something(nmax, r, temppsi, tot)
temppsi(1)=tot

!Compute the gradient of the density profile.

do n=2, nmax-1
 del_rho(n)=(rho(n+1)-rho(n-1))/(2.0d0*dr)
end do

del_rho(1)=(rho(2)-rho(1))/dr
del_rho(nmax)=(rho(nmax)-rho(nmax-1))/dr

!Compute the energetic components of the droplet from the components of the mean field. Note that
!the Orsay-Trento components are not supported in this version of the code, although they can be
!added in from other versions.

e_kin=0.0
e_rep=0.0
e_att=0.0
do n=1, nmax-1
 e_kin=e_kin+(dr/2.0)*(hbar**2/(2.0*cc))*(r(n)**2*del_rho(n)**2/(4.0*(rho(n)+1e-35))+r(n+1)**2
 *del_rho(n+1)**2/(4.0*(rho(n+1)+1e-35)))
 e_rep=e_rep+(dr/2.0)*(c/2.0)*(r(n)**2*rho(n)*rhobar(n)**(gamm4+1)+r(n+1)**2
 *rho(n+1)*rhobar(n+1)**(gamm4+1))
 e_att=e_att+(dr/2.0)*(1.0/2.0)*(r(n)**2*rho(n)*e_pot(n)+r(n+1)**2*rho(n+1)*e_pot(n+1))
end do

!Compute the total energy from the components.

e_tot=4.0*pi*(e_kin+e_rep+e_att)

write(*,*) "E/N is: ", e_tot*3.1577504d5/atoms
write(*,*) "E = ", e_tot*3.1577504d5

end subroutine

!-----------------------------------------------------------------------------------------
!This subroutine updates the calculation with the newest iteration's density and calculated
!energetic values.
!-----------------------------------------------------------------------------------------
subroutine update(nmax, r, rho, psi, newpsi, chempot, iter, maxdiff, handle)
implicit real*8 (a-h, o-z)
parameter (pi=4.0d0*atan(1.0d0))

dimension r(nmax)
dimension rho(nmax), psi(nmax)
dimension rhonew(nmax)
real*8, dimension(nmax) :: newpsi
real*8 :: maxdiff
character(len=4) :: handle

maxdiff=0.0
```

```
totrho=0.0
brho=0.021836d0/1.889725989d0**3.0d0

do n=2, nmax
 rhonew(n)=newpsi(n)**2.0d0/r(n)**2

 if (abs((newpsi(n)-psi(n))/r(n)).gt.maxdiff) then
  maxdiff=abs((newpsi(n)-psi(n))/r(n))
 end if
end do

!USE A QUADRATIC FIT OF POINTS n=2..4 TO CALCULATE THE DENSITY AT r(1)=0.0 TO AVOID DIVIDING BY
!ZERO.

call something(nmax, r, rhonew, tot)

rhonew(1)=tot

!Update psi and the density.

open(3,file='sphericalpsi'//trim(handle)//'.out')
do n=1, nmax
 write(3,3000) n, r(n), rhonew(n)
end do

!Include this cutoff when using LJ parameters.

!do n=1, nmax
! if (rhonew(n).lt.1e-30) then
!  rhonew(n)=0.0
! end if
!end do

do n=1, nmax
 psi(n)=newpsi(n)
 rho(n)=rhonew(n)
end do

3000 format (i6, 5(1x, 1pe12.5))
close(3)

write(*,*) "Maximum psi difference is", maxdiff

end subroutine

!---------------------------------------------------------------------

subroutine something(nmax, r, y, tot)
implicit real*8 (a-h, o-z)
real*8, dimension (1:nmax) :: r, y

tot=0.0
x=0.0
lim=5

do i=2, lim
prod=1.0
do j=2, lim
 if (i.ne.j) then
  prod=prod*(x-r(j))/(r(i)-r(j))
 end if
end do
tot=tot+prod*y(i)
end do

return
end subroutine
```

Vita

Matthew Francis Dutra was born in small-town North Carolina to his parents, James and Lori Dutra, in 1990. He attended high school for two years at J.F. Webb in Oxford, North Carolina, before transferring to the North Carolina School of Science and Mathematics in Durham, North Carolina, where he graduated in 2008. He continued his education at the University of North Carolina at Chapel Hill, where he performed undergraduate research in the field of physical computational chemistry under Dr. Yosuke Kanai. He graduated with his B.S. in Chemistry in the spring of 2012 before moving to Knoxville, Tennessee to start his graduate studies at the University of Tennessee under the mentorship of Dr. Robert Hinde. Matt hopes to continue his academic career post-graduation through a postdoctoral position, and aims to begin a research group of his own someday.