Path Integral Monte Carlo Simulations of Positronium in Argon

Lisa M. Larrimore

June 7, 2002

Abstract

Positronium (Ps) is modeled inside a solid Argon lattice using the path integral Monte Carlo (PIMC) technique. Statistics on site occupancy, Ps structure, and energies are collected for argon both with and without a monovacancy. The Ps atom is found to avoid the monovacancy and to have only modest increases in lifetime when the vacancy is available. Although the calculated lifetimes are somewhat different from experimental measurements, this model correctly predicts a longer lifetime for triplet state o-Ps (700 ± 15 ps) than for a bare positron (510 ± 5 ps) inside a defect-free Ar lattice. The p-Ps lifetime in Ar is predicted to be under 90 ps. The calculated wave function for isolated Ps is exactly fit to theory. The internal contact density of Ps in Ar, which describes the ratio of electron-positron overlap at the origin compared to that of free Ps, is determined from the modified Ps wave functions. The internal contact density is $\kappa = 1.25 \pm 0.02$ in Ar with or without a monovacancy. This result contrasts with the measurements of $\kappa \leq 1$ in many solids. Instead of being polarized by the surrounding atoms, Ps is slightly compressed by the potential model used.

Contents

1 Introduction

1.1 Positrons and Positronium

The Dirac equation, published in 1928, can accurately describe the relativistic behavior of particles such as the electron. It also produces negative energy solutions, which were initially considered an unphysical annoyance. After publishing his equation, Dirac realized that the negative energy solutions for electrons were not unphysical, but instead represented particles of positive charge. In 1930, Oppenheimer showed that these particles must have the mass of the electron. This predicted particle, a positively charged electron, is exactly what Carl Anderson observed when he applied a magnetic field in his cloud chamber at Caltech in 1932, and this new particle was named the positron [1].

Like electrons, positrons are leptons with mass 511 keV and spin 1/2. The difference is that their charges and magnetic moments, while equal in magnitude, are opposite in sign. When the wave functions of an electron and positron overlap, the two particles annihilate; due to conservation laws, they must release at least two photons with a total energy of 1022 keV.

In 1943, Mohorovicic postulated the existence of positronium (Ps), a bound state of an electron and a positron, which was first experimentally observed by Deutsch in 1951 [2]. Since the electron and positron interact via the Coulomb potential, Ps can be solved in the nonrelativistic limit much like hydrogen. Where the reduced mass of hydrogen is approximately equal to the mass of the electron, m_e, the reduced mass of Ps is $m_e/2$. This doubles the Bohr radius and halves the energy levels of Ps in comparison with hydrogen, as will be demonstrated in Section 2.1.

There are several important differences, however, between Ps and hydrogen. One is that Ps, unlike hydrogen, has a finite lifetime. Another is that there are different higher-order corrections to the energies of both Ps and hydrogen due to effects such as relativity and spin-orbit coupling; the corrected energy levels are known as fine and hyperfine structure. These differences will be discussed further in Section 2.1. The important implications of these results on this research will be seen in Section 2.2.

When the spins of the electron and positron forming Ps are antiparallel, the total

spin is zero, and the state is known as singlet or para-positronium (p-Ps). When the spins are parallel, they sum to one, and the state is triplet or ortho-positronium (o-Ps). In Dirac's braket notation, each of these states can be represented by a ket of the form $|m_s, s\rangle$, where m_s and s are the spin quantum numbers: p-Ps is represented as $|0, 0\rangle$, and o-Ps is represented as $|0, 1\rangle$ or $|\pm 1, 1\rangle$. In vacuum, the singlet state generally annihilates in 0.125 ns to form two 511 keV photons traveling in approximately opposite directions [3]. Due to conservation of angular momentum, the triplet state cannot decay to two photons; at least three are needed [4]. Because three-photon annihilation is less likely than two-photon annihilation, the mean lifetime of isolated o-Ps is 140 ns. In most solids, this is reduced to a few nanoseconds due to pick-off annihilation from other electrons [3], as will be described in Section 2.2.

1.2 Positronium Experiments

Positrons can serve as a useful and non-destructive tool for probing materials because the electrons with which they annihilate are quickly replaced by others. Because there are no positrons already present in a material being probed, it is fairly easy to determine the history of the probe positrons. For this reason, positron probes can resolve small defects that other probes, such as electrons, cannot distinguish. Positron experiments have a wide variety of uses, from detecting changes in weapons materials [5] to determining whether a diamond is natural or synthetic [6].

In some experiments, positrons are created from the beta decay of sources such as ^{22}Na. Because the radioisotopes used have a high proton/neutron ratio, protons in their nuclei will spontaneously become neutrons, in which case conservation laws require the release of a positron and a neutrino. This reaction is described by the equation

$$\mathrm{Na}^{22} \rightarrow \mathrm{Ne}^{22} + \mathrm{e}^{+} + \nu^{-} + \gamma_{(1.27\mathrm{MeV})}. \tag{1.1}$$

Beta decay can be used to produce low-intensity positron beams for small experiments. It can also be used by larger facilities to create very high energy positrons; for example, the Pelletron Accelerator at Lawrence Livermore National Laboratory (LLNL) uses beta decay to create a MeV positron beam [5].

1.2 Positronium Experiments

Positrons can also be created through pair production, in which a photon turns into an electron-positron pair. Any photon with sufficient energy can do this, though the rate at which positrons are produced in atmospheric background radiation is not very useful for experiments. To increase the rate of pair production, it is necessary to have a large amount of radiation and many nuclei with which to interact. At large experimental facilities, such as LLNL, this can be done with particle accelerators, as seen in Figure 1. When a fast-moving electron hits a metal target, it quickly slows down,

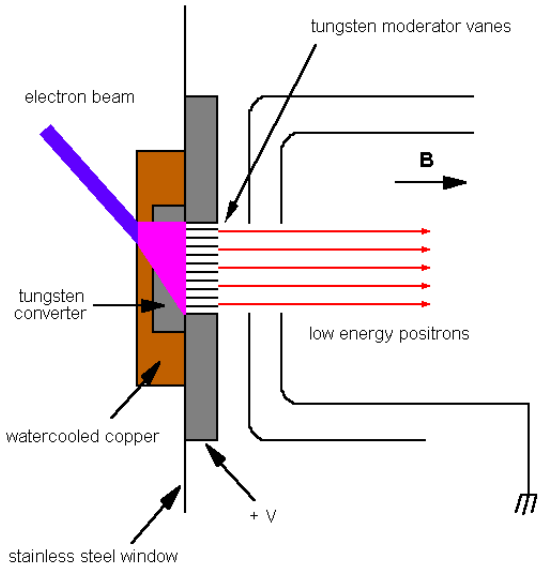

Figure 1: Positron production at LLNL. The intense *bremsstrahlung* radiation produced from decelerating electrons interacts with the nuclei in the metal, resulting in pair production of positrons. [8]

giving off *bremsstrahlung*, or "braking radiation." The power radiated by a point charge is in the forward direction and is proportional to the square of its acceleration [7]. This radiation then interacts with the nuclei in the metal, resulting in pair production of positrons. The Electron-Positron Beam Facility at LLNL produces the most intense beam of positrons in the world, at about 10^{10} positrons per second [8].

Inside an insulating solid, it is generally assumed that the positron can form a bound state with an electron and reach thermal equilibrium inside its host within about 10 ps after entering the material [9]; we can therefore model Ps in the equilibrium state to

study its behavior inside a solid. As you will see in Section 4, the wavefunction of this state is very similar to the ground state. We must also consider the perturbing effects of other electrons in the solid, since unperturbed Ps only exists in large cavities in any condensed medium [10]. The annihilation rate of a state of Ps in a solid is given by

$$\Gamma = \kappa\Gamma_0 + \Gamma_{\text{p.o.}}, \tag{1.2}$$

where Γ_0 is the annihilation rate for unperturbed Ps and $\Gamma_{\text{p.o.}}$ is the pick-off annihilation rate due to other electrons in the solid. κ is known as the "internal contact density" and describes the spatial extent of the Ps atom [10]. For isolated Ps, $\kappa = 1$, but due to the potentials inside solids, κ can decrease or increase as Ps is polarized or compressed. The behavior of Ps inside solids will be described further in Section 2.2.

Because the photons emitted when a positron and electron annihilate have a distinct energy, they can be easily detected by only looking for photons with an energy around 511 keV, which signals p-Ps decay. Information about these photons reveals information about the annihilation event. That is, the time between when a positron is born and sent inside a material and when the 511 keV photons are detected gives the lifetime of the positron, which helps in determining the electron charge density around the positron. This is discussed more formally in Section 2.2.1. As seen in Figure 2, the experimental lifetime is much longer if the positron is in a region of low electronic density. This correlation can result in valuable information about the material being probed. For example, if stainless steel is strained less than 10% of the way to failure, this early fatigue damage cannot be detected by standard nondestructive evaluation techniques, but a 210 ps positron lifetime increases in intensity from 15% to over 70% due to the increased number of defects [12]. Other measurement techniques, angular correlation of annihilation radiation (ACAR) and Doppler-broadening spectroscopy, give information about the momentum and energy distribution of the positron and annihilating electron. These two phenomena were not directly simulated for this thesis work, but would be interesting grounds for a future study.

There are many unanswered questions about the behavior of positrons in solids, and, correspondingly, experimentalists sometimes use inaccurate models to interpret

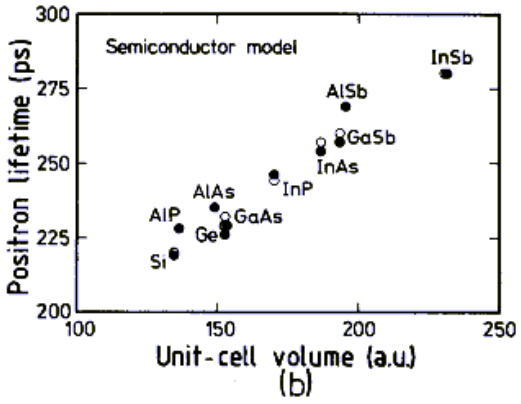

Figure 2: Positron lifetime in semiconductor lattices. Open circles are experimental values, closed circles are theoretical values based on the semiconductor model of Puska *et al.* [11]. Note that the positron lifetime increases when it is inside a material with larger cavities, and thus lower electronic density.

their data. Nakanishi *et al.* have parameterized a correlation curve relating positron lifetime to the size of open volumes in systems with well-defined open regions [13], and this method is now extensively used to measure the free volumes in other insulating systems [14]. The free-volume models underlying this approach, however, are based on simplistic assumptions, such as assuming simple geometric shapes for the free volume regions. Although the free-volume approach has produced correct results in many systems, it may be obscuring our physical understanding of Ps in solids. The problem of a positron in a crystal is too complex for an analytic solution, but it may be treated using computational methods.

1.3 Computational Methods

The increasing processing power of computers has led many physicists to examine ways to turn problems that cannot be solved analytically into ones that may be easily solved computationally. Since the 1970s, physicists have employed so-called Monte Carlo algorithms to study quantum many-particle systems. Named for the Mediterranean casino town, a "Monte Carlo" method is any algorithm that involves a pseudorandom number generator. Monte Carlo techniques can be used for everything from calculating inte-

grals over high-dimensional volumes to modeling traffic (treating the behavior of cars with random numbers) [15]. Quantum Monte Carlo methods include variational Monte Carlo (VMC), in which a trial wave function is adjusted using variational methods, and diffusion Monte Carlo (DMC), in which the similarity between the Schrödinger equation and the diffusion equation is exploited [15]. The technique used in the work discussed here, path-integral Monte Carlo (PIMC), is one of many variants on the Monte Carlo technique; it is designed for systems at finite temperatures [16].

PIMC relies on a useful isomorphism between a quantum system and a classical system. The partition function for a quantum system can be expressed as a Feynman path integral, save that the real time t is replaced by an imaginary time $\tau = it$; this form will be discussed in Section 3.1. When the potential does not vary over a distance greater than the thermal wavelength, this path integral becomes classical [17]. The partition function needed to calculate thermal averages for a quantum system of N particles is shown to be mathematically equivalent to the partition function for N classical ring polymers [16].

Using this result, we can represent our two-particle Ps quantum system as $N = 2$ classical ring polymers; in other words, two closed chains of several hundred to several thousand "beads." PIMC has been used by others to model the entire Ps atom as a single chain [18, 19], but we can obtain more information (such as κ or the binding energy) by treating each particle separately. The interaction between the two chains was guided by the "Pollock" propagator, which will be discussed in Section 3.2.1. The potential felt by the chains of beads due to the surrounding solid was determined using both phenomenological potential models and density functional theory (DFT), which will be discussed in Sections 3.2.2 and 3.2.3.

In earlier work, we have used this method to illustrate systematic differences between the positron distribution in a hard cavity and the corresponding free-volume model [20]. Proceeding to the task of modeling a more realistic solid, we chose to simulate solid argon (Ar) in this work, both because it is a relatively simple system and because there are a number of disputed facts about Ar in the literature that we hoped to address. The results from this research are presented in Section 4.

2 Theory

2.1 Quantum Mechanical Description of Ps

2.1.1 Wavefunctions and Energies from Schrödinger Equation

Free Ps, like free hydrogen, is a two-body system with Hamiltonian

$$\hat{H} = \frac{\hat{\boldsymbol{p}}_+^2}{2m_+} + \frac{\hat{\boldsymbol{p}}_-^2}{2m_-} - \frac{e^2}{\boldsymbol{r}}, \tag{2.1}$$

where $\boldsymbol{r} = \boldsymbol{r}_+ - \boldsymbol{r}_-$. For Ps, $m_- = m_+ = m_e$, which means that the reduced mass is $\frac{m_e}{2}$, not m_e as in the case of hydrogen. We will briefly review the solution to this system, for it serves as a check for our numerical simulation.

We are interested in solving the time-independent Schrödinger equation,

$$\hat{H}\phi(\boldsymbol{r}_+, \boldsymbol{r}_-) = E\phi(\boldsymbol{r}_+, \boldsymbol{r}_-), \tag{2.2}$$

where $\phi(\boldsymbol{r}_+, \boldsymbol{r}_-)$ is the two-particle wavefunction. Since we are considering free Ps, it is useful to convert to center of mass coordinates, in terms of which Eq. (2.2) becomes

$$-\left(\frac{\hbar^2}{m_e}\nabla^2 + \frac{e^2}{r}\right)\psi(\boldsymbol{r}) = E\psi(\boldsymbol{r}). \tag{2.3}$$

In spherical coordinates, $\boldsymbol{r} = (r, \theta, \phi)$,

$$\nabla^2 = \frac{1}{r^2}\frac{\partial}{\partial r}r^2\frac{\partial}{\partial r} + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\sin\theta\frac{\partial}{\partial\theta} + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}, \tag{2.4}$$

and the Schrödinger equation is separable, which means that the wave function can be separated into a product of functions of single variables, $\psi(r, \theta, \phi) = R(r)\Theta(\theta)\Phi(\phi)$.

Equation (2.3) can then be written as

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) = \left[\frac{\ell(\ell+1)}{r^2} - \frac{m_e}{\hbar^2}\left(\frac{e^2}{r}+E\right)\right]R \tag{2.5}$$

$$\sin\theta\frac{d}{d\theta}\left(\sin\theta\frac{d\Theta}{d\theta}\right) = \left[m_\ell^2 - \ell(\ell+1)\sin^2\theta\right]\Theta \tag{2.6}$$

$$\frac{d^2\Phi}{d\phi^2} = -\,m_\ell^2\Phi. \tag{2.7}$$

Equations (2.6) and (2.7) are familiar; the Φ equation is solved by sines and cosines (which can be written as complex exponentials), and the Θ equation is solved by associated Legendre functions:

$$\Theta(\theta)\Phi(\phi) = P_\ell^{m_\ell}(\cos\theta)e^{im_\ell\phi} = Y_{\ell,m_\ell}(\theta,\phi), \tag{2.8}$$

where m_ℓ can range from $-\ell$ to ℓ.

The solution to the radial equation, (2.5), can be found in the standard way, the details of which are described in Appendix A. Using various substitutions, Eq. (2.5) can be written in the form of Laguerre's associated differential equation, and the full radial wave functions are

$$R_{n\ell}(\rho) = N_{n\ell}e^{-\rho/2}\rho^\ell L_{n+\ell}^{2\ell+1}(\rho), \tag{2.9}$$

where $\rho = r/a_0 n$ is written in terms of the Bohr radius, $a_0 = \hbar^2/me^2$, $L_{n+\ell}^{2\ell+1}(\rho)$ are the associated Laguerre polynomials, and $N_{n\ell}$ is a normalization factor given by

$$N_{n\ell} = \left[\frac{(n-\ell-1)!}{2n\left[(n+\ell)!\right]^3}\right]^{\frac{1}{2}}. \tag{2.10}$$

Along the way to this solution in Appendix A, to satisfy a boundary condition, we found that the energy levels are quantized,

$$|E_n| = \frac{m_e e^4}{4\hbar^2 n^2}, \tag{2.11}$$

where n is known as the principle quantum number. Further, ℓ, the azimuthal quantum

number, is constrained to lie between 0 and $n-1$, inclusive. Note that each Ps energy level is half of the corresponding hydrogen energy level; for instance, the Ps ground state energy, for $n = 1$, is $E_1 = -6.8$ eV $= -0.25$ atomic units (a.u.), compared to -13.6 eV for hydrogen.

The complete wave function can be written by combining Equations (2.8) and (2.9):

$$\psi_{nlm}(r,\theta,\phi) = -\left[\left(\frac{2}{(na_0)^3}\right)\frac{(n-\ell-1)!}{2n\left[(n+\ell)!\right]^3}\right]^{1/2} e^{-\rho/2}\rho^{\ell}L_{n+\ell}^{2\ell+1}(\rho)Y_{\ell,m_\ell}(\theta,\phi), \qquad (2.12)$$

where $\rho = r/a_0 n$.

Since our investigation is an equilibrium calculation based on the assumption that Ps thermalizes quickly after entering a material, and the temperatures we are considering are such that $kT \ll E_1 - E_0$, we are interested in the ground state wavefunction,

$$\psi_{100}(r,\theta,\phi) = \left(\frac{1}{8\pi a_0^3}\right)^{1/2} e^{-r/2a_0}. \qquad (2.13)$$

Note that this is identical to the ground state wavefunction for hydrogen except for the factor of two in the exponential; this means that the positron and electron forming Ps are most likely to be twice as far apart as the proton and electron forming hydrogen.

In our simulation, we calculate the radial probability density for Ps, $P(r)$, which gives the probability density of finding the positron and electron a distance r apart. $P(r)$ is formally obtained by multiplying the square of the wavefunction by a spherical shell volume element. For the $1S$ state that we are considering, Eq. (2.13) leads to

$$P(r) = \frac{r^2}{2a_0^3}e^{-r/a_0}. \qquad (2.14)$$

2.1.2 Ps Fine Structure Energy Corrections

The Hamiltonian given in Eq. (2.1) is not the whole story; various other effects, such as spin, result in small perturbations to this Hamiltonian, which cause small corrections in the quantized energy levels given by (2.11). The various perturbations to the Ps energy summarized below are described by Griffiths [21]. The origins of some of the energy corrections for Ps are similar to those for hydrogen, such as the relativistic

kinetic energy, spin-orbit coupling, and the Lamb shift, but they have different relative magnitudes in Ps. For example, the Lamb shift, caused by the quantization of the electromagnetic field, is of order $\alpha^5 mc^2$ in Ps, where $\alpha = e^2/\hbar c = 1/137$ is the fine structure constant, so we will ignore it in what follows. There are also new effects in Ps that cannot be compared to the fine or hyperfine structure of hydrogen. All of these are described below.

The Schrödinger equation uses a non-relativistic representation of the kinetic energy of each particle, $\hat{T} = \hat{p}^2/2m$. Expanding the relativistic expression for kinetic energy to the next order,

$$\hat{T}_{\rm rel} = \sqrt{\hat{p}^2c^2 + m^2c^4} - mc^2 = \frac{\hat{p}^2}{2m} + \frac{\hat{p}^4}{8m^3c^2} + \cdots . \tag{2.15}$$

The $\Delta E_{\rm rel} = \left\langle \psi \left| \hat{T} - \hat{T}_{\rm rel} \right| \psi \right\rangle$ induced by this change in the Hamiltonian is one-eighth the size of the perturbation for hydrogen: the fact that there are two relativistic particles doubles it, and the fact that $\langle \hat{p}^4 \rangle$ goes like $(mc^2)^4$ reduces it by a factor of 2^4.

Another perturbation is caused by the fact that each spinning particle is a dipole and sees the other charged particle orbiting around it and creating a magnetic field. In hydrogen, the electron spin combines with the relativistic correction to give the fine structure, and the proton spin, reduced by a factor of m_e/m_p, results in hyperfine splitting. For Ps, however, this factor becomes unity, which means the hyperfine and fine splittings are of the same order.

Some new corrections that are not relevant to hydrogen are due to the finite propagation time for the electromagnetic field and to the possibility of virtual annihilation. The former results from the fact that the positron, unlike the proton in hydrogen, cannot be considered to be stationary. The second results from the fact that an electron and positron can form a virtual photon, as illustrated in Figure 3. This correction is proportional to $|\psi(0)|^2$, since the two particles must be in the same place for "annihilation" to occur.

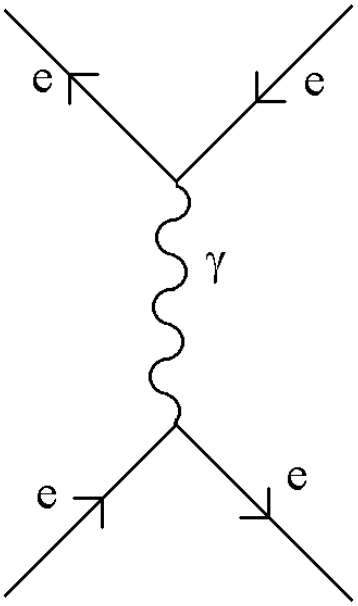

Figure 3: Feynman diagram of the virtual annihilation of an electron and positron. This causes the annihilation correction to Ps energy, which is proportional to $|\psi(0)|^2$.

These factors combine to give the total fine structure correction for Ps:

$$\Delta E_{\rm fs} = \frac{\alpha^4 mc^2}{2n^3}\left[\frac{11}{32n} - \frac{1+\frac{\epsilon}{2}}{2\ell+1} + \frac{\delta_{\ell 0}\delta_{s1}}{2}\right] \tag{2.16}$$

where ϵ is given by

$$\epsilon = \begin{cases} 0 & s=0 \\ -\frac{(3\ell+4)}{(\ell+1)(2\ell+3)} & j=\ell+1, s=1 \\ \frac{1}{\ell(\ell+1)} & j=\ell, s=1 \\ \frac{(3\ell-1)}{\ell(2\ell-1)} & j=\ell-1, s=1 \end{cases} \tag{2.17}$$

Note that the energy difference between o-Ps ($s=1$, $\ell=0$) and p-Ps ($s=0$, $\ell=0$) ground states can be written as

$$\Delta E_{\rm o-p} = \frac{7}{12}\alpha^4 mc^2. \tag{2.18}$$

2.2 Ps in Solids

2.2.1 Internal Contact Density κ

Equation (2.16) was derived for free Ps, and a close reading of reference [21] shows that, for the $\ell = 0$ states, the prefactor contains the unperturbed wavefunction at the origin,

$$\frac{\alpha^4 mc^2}{2n^3} = |\psi(0)|^2 \frac{\pi e^2 \hbar^2}{2m^2 c^2}. \tag{2.19}$$

Inside a solid, however, the surrounding environment can change the amount of overlap between the electron and positron, $|\psi(0)|^2$, by pulling the particles apart or squeezing them together. This change is described by κ, the internal (or relative) contact density of Ps:

$$\kappa = \frac{|\psi(0)|^2}{|\psi_{\text{free}}(0)|^2}. \tag{2.20}$$

Eq. (2.13) shows that the denominator is $1/8\pi a_0^3$. We can write the center-of-mass wavefunction at the origin in terms of the six-coordinate, two-particle wavefunction defined in Eq. (2.2) as

$$|\psi(0)|^2 = \int |\phi(\boldsymbol{r}_+, \boldsymbol{r}_-)|^2 \, \delta(\boldsymbol{r}_+ - \boldsymbol{r}_-) d^3\boldsymbol{r}_+ d^3\boldsymbol{r}_-, \tag{2.21}$$

where the Dirac delta function picks out only the overlap at the origin. The internal contact density can thus be rewritten as [10]

$$\kappa = 8\pi a_0^3 \int |\phi(\boldsymbol{r}_+, \boldsymbol{r}_-)|^2 \, \delta(\boldsymbol{r}_+ - \boldsymbol{r}_-) d^3\boldsymbol{r}_+ d^3\boldsymbol{r}_-. \tag{2.22}$$

For unperturbed Ps, $\kappa = 1$, and κ will decrease or increase along with $|\psi(0)|^2$. From Equations (2.19) and (2.22), we see that the energy splitting between o-Ps and p-Ps given in Eq. (2.18) is actually proportional to κ:

$$\Delta E_{\text{o-p}} = \frac{7}{12} \kappa \alpha^4 mc^2. \tag{2.23}$$

There is an additional contribution to the hyperfine splitting, due to the atoms in the crystal, that is not proportional to κ, but this contribution is at most 5% of $\Delta E_{\mathrm{o-p}}$ for free Ps, so it can be neglected [10].

The internal contact density κ modifies the annihilation rates of p-Ps and o-Ps in solids. As mentioned in the Introduction, the self-annihilation rate of Ps in vacuum is about $\Gamma_0^s = 8$ ns^{-1} for the singlet state and $\Gamma_0^t = 0.007$ ns^{-1} for the triplet state [3], and the annihilation rate is greater for the singlet state because the more common two-photon annihilation cannot occur in the triplet state. If Ps becomes squeezed inside a solid, however, the new self-annihilation rate is proportional to κ. There is also an additional contribution to the annihilation rate due to the other electrons in the solid; this is known as "pick-off" annihilation. As an elaboration of Eq. (1.1), we can say that the annihilation rates for Ps in solids are thus given by

$$\begin{aligned} \Gamma^s &= \kappa\Gamma_0^s + \Gamma_{\mathrm{p.o.}}, \\ \Gamma^t &= \kappa\Gamma_0^t + \Gamma_{\mathrm{p.o.}}, \end{aligned} \tag{2.24}$$

where $\Gamma_{\mathrm{p.o.}}$ is the rate of pick-off annihilation [10].

2.2.2 Solid Argon

We performed our simulation in solid argon because of the presence in the literature of both experimental and theoretical results for this solid. For instance, rare-gas solids like Ar have been studied for use as sources of slow positrons or of thermalized Ps [22]. As will be discussed in Section 3.2.2, calculating the pseudopotential felt by the electron is difficult, so to inaugurate our simulation method we used a system in which this potential had already been studied in detail.

Argon forms a face-centered cubic (fcc) crystal lattice, which means that there is an atom at each corner of the unit cell and at the center of each face, as illustrated in Figure 4. One side of the unit cell has length $a = 10.04$ a.u. [23], where one atomic unit of distance is 0.529177×10^{-10} m. The smallest structural unit of atoms used to build a crystal is known as the basis. For Ar, the locations of the basis atoms are $(0, 0, 0)$, $(a/2, a/2, 0)$, $(a/2, 0, a/2)$, and $(0, a/2, a/2)$ in Cartesian coordinates. An ideal

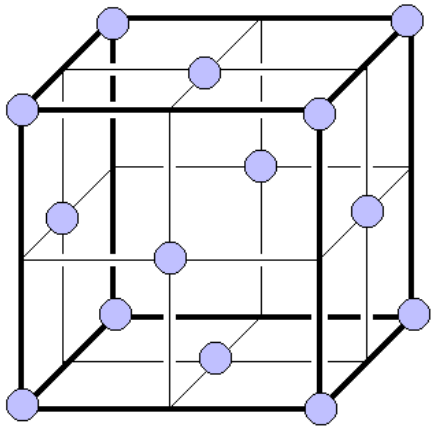

Figure 4: The face-centered cubic crystal lattice.

Ar crystal is generated when this basis is repeated an infinite number of times. It is often convenient to know the lattice vectors for the Ar crystal,

$$\boldsymbol{a}_1 = a(\hat{\boldsymbol{y}} + \hat{\boldsymbol{z}})/2, \tag{2.25}$$

$$\boldsymbol{a}_2 = a(\hat{\boldsymbol{x}} + \hat{\boldsymbol{z}})/2, \tag{2.26}$$

$$\boldsymbol{a}_3 = a(\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}})/2, \tag{2.27}$$

which allow us to work in either lattice vector coordinates or Cartesian coordinates. Since the unit cell, of volume a^3, contains eight corner atoms that are each shared among eight unit cells and six atoms on the faces that are each shared among two unit cells, the density of atoms in solid Ar is $\rho = 4/a^3$.

Until the 1960s, elements in the "noble gas" group of the periodic table, such as Ar, were considered to be inert gases due to their filled outermost (valence) shells. Ar forms a solid crystal only below its melting point of 83.81 K, which corresponds to $\beta = 1/kT = 3895$ a.u. [23]. Because Ar has a high ionization potential, the energy of an incident positron must be over about 10 eV for it to strip off one of argon's 18 electrons to form Ps. This is illustrated in Figure 5, which shows the results of an experiment in which positrons were sent through an Ar surface with different incident energies. Y_+ is the percentage that were reemitted as positrons, Y_{Ps} is the percentage

that were reemitted as Ps, and Y_S is the percentage that annihilated in the solid [24].

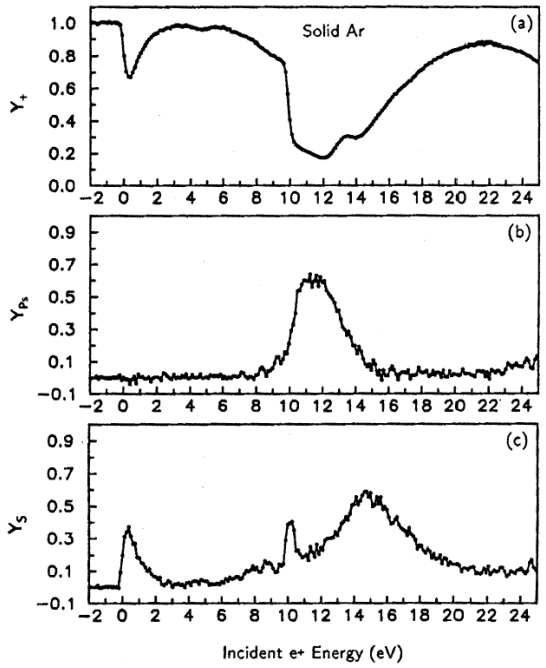

Figure 5: Positron behavior in solid Ar surface as a function of incident energy. The vertical axes represent the probability of (a) positron reemission, (b) Ps formation, and (c) positron annihilation in the solid. [24]

Experiments have also indicated that both p-Ps and o-Ps can form in Ar. As mentioned in the Introduction, the triplet state (o-Ps) cannot decay into two photons, so experimentalists detect its presence by looking at the ratio of 3γ decays to 2γ decays. This ratio was sufficiently high in Ar to confirm the presence of o-Ps [25].

Jean, Yu, and Zhou have studied positron annihilation in Ar, and after requiring one of their lifetimes to be 125 ps (due to p-Ps), they fit their data to two other lifetimes: 340–390 ps (which they attributed to the annihilation of single positrons) and 2.1–2.5 ns (which they attributed to o-Ps self-annihilation). Since 2.5 ns is a longer lifetime than anyone expects to see in a close-packed defect-free solid, they suggest that it is due to the self-trapping of Ps in voids [26]. Gullikson and Mills, however, point out that the long 2.5 ns lifetime is observed even at temperatures where there should be no voids [27]. Our own results in Section 4 show that Ps does not fall into an available

void in Ar, so if self-trapping occurs, it involves a more dramatic rearrangement of nuclei and electrons than is modeled by our potential together with the introduction of a monatomic vacancy.

In 1967, E. J. Woll developed a model for the positron wavefunction and lifetime in solid Ar by using the atomic sphere approximation (ASA). This allowed him to replace the unit cell with a sphere of radius 3.92 a.u., where he used $a = 9.94$ a.u. as the length of the Ar unit cell. He then numerically integrated the Schrödinger equation as a function of the distance r from the center of this sphere [28]. He compared his results to Liu and Robert's 1963 experimental measurement of the inverse lifetime of a single positron in Ar; they found $\Gamma = 2.3$ ns^{-1}, which corresponds to a lifetime of 435 ps [29]. He found a much better agreement with this lifetime when he included an attractive potential due to the virtual polarization of the electrons by the positron [28]. The potential seen by the positron with and without the polarization potential is seen in Figure 6. As will be seen in Section 4, the polarization potential causes the

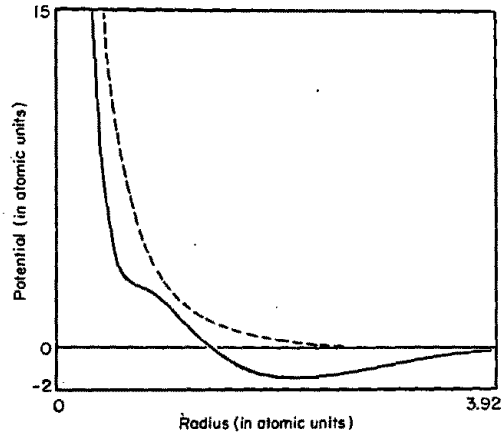

Figure 6: The potential seen by the positron in solid Ar, using ASA. The solid line includes the repulsion by the atomic cores, the Hartree attraction by the outer electrons, and the polarization potential. The dashed line shows the same potential without considering polarization. [28]

same effect in our simulation; there is a potential minimum around the Ar atoms, such that the Ps does not even become trapped in the cavity formed by a missing atom. Our calculation is different from Woll's, however, in several important ways. We use a

realistic solid structure, not ASA. Also, we are modeling Ps, not just a positron, and we have potentials for both the electron and the positron.

Unfortunately, we cannot compare everything we calculate in our simulation to results in the experimental literature yet. Most notably, we have found no experimental data on κ in Ar with which we can compare our own interesting result. Appendix D, however, demonstrates how such a measurement could be accomplished, as it has been for other materials. Future investigations will involve simulating Ps inside α-SiO_2, in which κ has been determined to be 0.31 ± 0.02 [30].

3 Computational Methods

3.1 Path Integral Monte Carlo (PIMC)

As described in the Introduction, path integral Monte Carlo (PIMC) is one of the many quantum Monte Carlo techniques that are used to simulate quantum systems using pseudorandom number generators.

In quantum statistical mechanics, the expectation value of the observable associated with the operator $\hat{A}$ is given by

$$\left\langle \hat{A} \right\rangle = \frac{1}{Z} \operatorname{Tr}[\exp(-\beta \hat{H}) \hat{A}], \tag{3.1}$$

where $\exp(-\beta \hat{H})$ is called $\hat{\rho}$, the density matrix. Z is the partition function, which can be written in the position basis as

$$\begin{aligned} Z &= \operatorname{Tr}[\exp(-\beta \hat{H})] \\ &= \int dx \left\langle x \left| \exp(-\beta \hat{H}) \right| x \right\rangle \\ &= \int dx_1 \cdots \int dx_P \left\langle x_1 \left| e^{-\frac{\beta \hat{H}}{P}} \right| x_2 \right\rangle \left\langle x_2 \left| \cdots \right| x_P \right\rangle \left\langle x_P \left| e^{-\frac{\beta \hat{H}}{P}} \right| x_1 \right\rangle, \end{aligned} \tag{3.2}$$

inserting $P - 1$ complete sets of states. Using the Hamiltonian for one particle in one dimension under a potential $V(x)$, and making the approximation that, for large P,

the kinetic and potential energy operators commute,

$$\begin{aligned}
\left\langle x \left| e^{-\frac{\beta\hat{H}}{P}} \right| x' \right\rangle &= \left\langle x \left| e^{-\frac{\hat{p}^2}{2m}\frac{\beta}{P} - \frac{\hat{V}\beta}{P}} \right| x' \right\rangle \\
&\approx \left\langle x \left| e^{-\frac{\beta\hat{V}}{2P}} e^{-\frac{\hat{p}^2}{2m}\frac{\beta}{P}} e^{-\frac{\beta\hat{V}}{2P}} \right| x' \right\rangle \\
&= e^{-\frac{\beta}{2P}(V(x)+V(x'))} \left\langle x \left| e^{-\frac{\hat{p}^2}{2m}\frac{\beta}{P}} \right| x' \right\rangle \\
&= e^{-\frac{\beta}{2P}(V(x)+V(x'))} \int \left\langle x \left| e^{-\frac{\hat{p}^2}{2m}\frac{\beta}{P}} \right| p \right\rangle \left\langle p \left| x' \right\rangle dp \right. \\
&= e^{-\frac{\beta}{2P}(V(x)+V(x'))} \int e^{-\frac{\hat{p}^2}{2m}\frac{\beta}{P}} \frac{e^{\frac{ipx}{\hbar}}}{\sqrt{2\pi\hbar}} \frac{e^{\frac{-ipx'}{\hbar}}}{\sqrt{2\pi\hbar}} dp \\
&= e^{-\frac{\beta}{2P}(V(x)+V(x'))} \frac{1}{2\pi\hbar} e^{-\frac{mP}{2\hbar^2\beta}(x-x')^2} \int_{-\infty}^{\infty} e^{-\frac{\beta}{2mP}\left(p - \frac{mPi(x-x')}{\beta\hbar}\right)^2} dp \\
&= \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{1/2} e^{-\frac{mP}{2\hbar^2\beta}(x-x')^2} e^{-\frac{\beta}{2P}(V(x)+V(x'))}. \qquad (3.3)
\end{aligned}$$

Substituting this result into Eq. (3.2) for the partition function gives

$$\begin{aligned}
Z &= \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{\frac{P}{2}} \int dx_1 \cdots dx_P \exp\left[\sum_{i=1}^{P} -\frac{mP}{2\hbar^2\beta}(x_i - x_{i+1})^2 - \frac{\beta}{2P}(V(x_i) + V(x_{i+1}))\right] \\
&= \left(\frac{mP}{2\pi\beta\hbar^2}\right)^{\frac{P}{2}} \int dx_1 \cdots dx_P \exp\left[-\beta\left(\frac{k}{2}\sum_{i=1}^{P}(x_i - x_{i+1})^2 + \frac{1}{P}\sum_{i=1}^{P} V(x_i)\right)\right], \qquad (3.4)
\end{aligned}$$

where $k = \frac{mP}{\beta^2\hbar^2}$ and $x_{P+1} = x_1$. As shown in Appendix B, this partition function (aside from the prefactor) is equivalent to the partition function for a classical ring polymer of P beads coupled by harmonic springs of constant k, where each bead feels a potential $\frac{V(x)}{P}$. This approximation is called the "primitive" approximation and is only valid for large P and high temperatures (low β); specifically, Binder claims,

$$\frac{P}{\beta} \gg \frac{\hbar^2}{m\sigma^2}, \qquad (3.5)$$

where σ is the characteristic distance over which the potential changes [16].

The above discussion considers a one-dimensional variable, x, which describes a single particle in one dimension. For our investigation, this result was extended to

a six-dimensional variable, $(\boldsymbol{r}_+, \boldsymbol{r}_-)$, which describes two particles moving in three dimensions. The Ps atom is thus modeled as two interacting cyclic polymer chains, each of length P. The two chains interact via a Coulombic potential, as will be described in Section 3.2. Models of a single positron and of Ps treated by this method are illustrated in Figure 7.

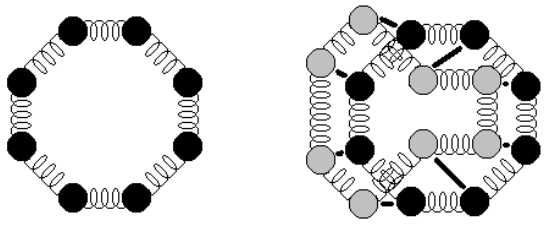

Figure 7: Models of quantum particles treated using PIMC. On the left is a single positron represented as a ring of beads that interact with their closest neighbors via a harmonic potential. On the right are a positron and an electron; each positron bead has a Coulombic interaction with the electron bead in the corresponding position, represented by the thick black lines. In our code, each particle is represented by thousands of beads, rather than the eight pictured here.

Equation (3.4) gives the path integral representation of the partition function. The "Monte Carlo" part of PIMC refers to the probabilistic method of treating this system. Obviously, all possible configurations of beads cannot be examined, so measurements are taken on a random selection. Instead of choosing equally from all possible states, and then weighting them by their Boltzmann factor ($\exp(-\beta U)$), it makes more sense to choose states with probability $\exp(-\beta U)$, and to then weight them equally. This is the standard Metropolis algorithm [31], which is known as an importance sampling technique. One pass through the Metropolis algorithm is described here, with the corresponding line numbers for our PIMC code (Appendix E).

1. A new configuration of beads is determined by moving a random portion of each of the old chains to a new position, using a Gaussian distribution. This is accomplished in the `tryboth` subroutine of our code, lines 773-817.

2. The potential energy difference of the new state relative to the present state, δU, is calculated. This is done in lines 591-643, and the energy difference times the

effective β is stored in the variable `vchange`.

3. If $\delta U \leq 0$, the new state is accepted. Otherwise, a random number $0 \leq \eta \leq 1$ is generated, and the new state is only accepted if $\exp(-\beta\delta U) > \eta$. This condition can be rewritten as $-\beta\delta U > \log \eta$, which is what we use in lines 644-651 of our code. The variable `det` stores the value of $-\beta\delta U$, and the variable `de` stores the value of $\log \eta$.

For a starting configuration, the beads in each chain are randomly placed in a Gaussian distribution about the origin, as illustrated in Figure 8. This is accomplished in the `init_beads` subroutine in lines 471-543 of the PIMC code. A Gaussian distribution is chosen because it is appropriate for a free quantum particle. We can see from Figure 9 that the final distribution of beads about their center of mass retains this same general shape, although each chain contracted due to the presence of the other.

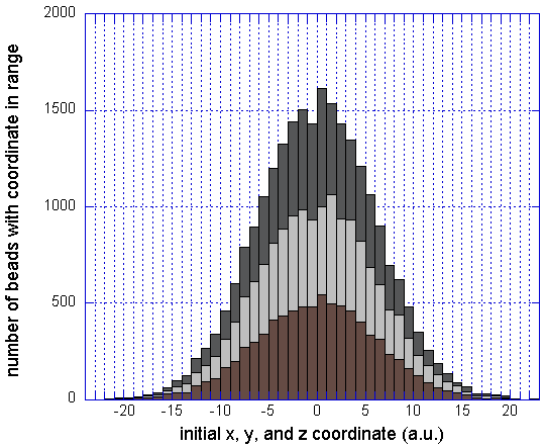

Figure 8: The beads representing the positron and electron are initially placed in a Gaussian distribution about the origin. This histogram shows the initial distribution of the x, y, and z coordinates of two 4000-bead chains. All of the x coordinates for both the positron and electron are shown in one shade of grey, and the y and z coordinates are the other shades.

When a section of a chain is moved during the first step of the Metropolis algorithm in our `tryboth` subroutine, the probability of moving a bead a certain distance from its original position is also given by a Gaussian distribution. The width of this distribution,

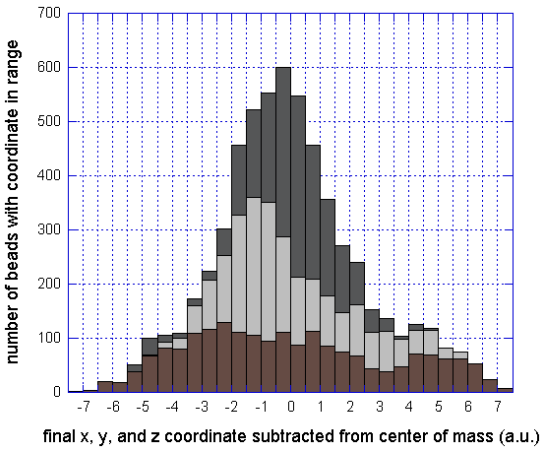

Figure 9: After several million passes through the Metropolis algorithm, the distribution of beads about their center of mass still maintains the general shape seen in Figure 8. The narrower width of the figure indicates that the beads have contracted from their starting configuration. This histogram shows the distribution of the x, y, and z coordinates of two 1000-bead chains, representing "free" Ps, in their final configuration. As in Figure 8, the different shades of grey represent the x, y, and z coordinates.

however, depends on the position of the bead in the chain, since the likelihood of a particular bead placement being favorable is conditional on the placements of its neighbors. We therefore decrease the Gaussian width as we move successive beads, using a recursive method developed by Levy [32].

In our code, the length of chain that is moved at each pass is adjusted every ten passes to keep the percentage of accepted moves around 50 percent. This typical portion of the chain that is redistributed at each pass is between 3 and 7 percent for the chain lengths used in this study. Also, every ten passes, instead of moving a portion of the current chain, a center-of-mass move is attempted using the same Metropolis algorithm; the entire chain is transposed by up to 0.1 a.u. in each of the Cartesian directions. Since the majority of these center-of-mass moves are accepted, this allows the beads to sample a greater region within the Ar unit cell.

After a large number of passes, the system will settle into the states with the lowest free energy. We can measure some property by recording its value at each pass and averaging these measurements, as Eq. (3.1) suggests. For example, we calculate the

radial distribution function, $P(r)$, which gives the likelihood of finding the electron and the positron a distance r apart, by forming a histogram of the relative distance between each pair of electron and positron beads at every pass. Section 3.2 will describe how we measure the energy of the particles, Section 3.3 will describe measuring the overlap between the two particles, and Section 3.4 will describe the pair correlation function, which is used to obtain information about where Ps is located in the crystal lattice.

3.2 Ps Energy

As seen in Section 3.1, the PIMC method depends on calculating the likelihood of a given Ps configuration. Calculating this, however, is one of the more subtle parts of the problem, since we need to determine the likelihood due to the Coulombic interaction of the Ps particles as well as the energy due to the surrounding atoms in the crystal. The former problem is dealt with using the Pollock propagator, which actually does not use the Coulomb energy, but takes a more fundamental approach. This propagator will be described in Section 3.2.1. To determine the potential felt by the positron due to the surrounding Ar atoms, we use a technique called density functional theory (DFT), as described in Section 3.2.3. The external potential felt by the electron is difficult to calculate, and we use a phenomenological form based on experimental data, as described in Section 3.2.2.

The δU used in the Metropolis algorithm is simply the potential energy difference between two different states. To model Ps accurately, therefore, we only need estimate the potential energy of the particles in a given configuration. The kinetic energy is also of interest, however, for comparison with theoretical and experimental results. Section 3.2.4 contains a discussion of kinetic energy estimators.

3.2.1 Electron-Positron Interaction: The Exact Coulombic Propagator

A real positron and electron interact via the Coulomb potential,

$$V_{\text{coul}} = -\frac{e^2}{r}, \tag{3.6}$$

where r is the relative distance between the two particles. For our computer simulation using the approximation of Eq. (3.4), however, this potential would result in the two chains of beads collapsed on top of each other, unable to escape the negative infinity at the origin.

One solution to this problem, used in our earlier work, is to use the Yukawa potential,

$$V_{\text{yuk}} = -\frac{e^2}{r}[1 - \exp(-r/a)], \tag{3.7}$$

and to extrapolate the results to the limit $a \to 0$. Müser and Berne have shown that as $a \to 0$ and $P \to \infty$, the Yukawa potential results in the correct path integral for the Coulomb system [33]. In earlier work, we have used this potential to generate the correct $1s$ state for free Ps, and thence to model Ps in a hard spherical well [20].

For this thesis, however, we used an approach developed by Roy Pollock, which takes advantage of the fact that free Ps can be solved exactly, as seen in Section 2.1. At a temperature β, the Coulomb pair density matrix is given by

$$\begin{aligned}
\rho(\boldsymbol{r}, \boldsymbol{r}'; \beta) &= \left\langle \boldsymbol{r} \left| e^{-\beta \hat{H}} \right| \boldsymbol{r}' \right\rangle \\
&= \langle \boldsymbol{r}| \, e^{-\beta \hat{H}} \sum_s |\psi_s\rangle \, \langle \psi_s| \, \boldsymbol{r}' \rangle \\
&= \sum_s e^{-\beta E_s} \, \langle \boldsymbol{r} \, |\psi_s\rangle \, \langle \psi_s| \, \boldsymbol{r}' \rangle \\
&= \sum_s e^{-\beta E_s} \psi(\boldsymbol{r}) \psi^*(\boldsymbol{r}'),
\end{aligned} \tag{3.8}$$

where $\boldsymbol{r}$ and $\boldsymbol{r}'$ represent the relative coordinates between the positron and the electron, and the sum is over the Coulomb potential energy eigenstates. This can also be written as

$$\rho(\boldsymbol{r}, \boldsymbol{r}'; \beta) = \frac{\exp\left(-(\boldsymbol{r} - \boldsymbol{r}')^2 / 2\frac{\hbar^2}{\mu}\beta\right)}{\left(2\pi\frac{\hbar^2}{\mu}\beta\right)^{3/2}} e^{-P(\boldsymbol{r}, \boldsymbol{r}'; \beta)}, \tag{3.9}$$

where this equation can be considered a definition for $P(\boldsymbol{r}, \boldsymbol{r}'; \beta)$, which is the part of

ρ that exists due to the Coulomb interaction [34, 35]. In our code, the current positron and electron separation is $\boldsymbol{r}$, and the separation of the next two beads along the chain is $\boldsymbol{r}'$. We are able to determine the $P(\boldsymbol{r}, \boldsymbol{r}'; \beta)$ associated with a particular configuration of our Ps chains by looking up each state in a table that contains the exact solution for P from Pollock's code [35, 36]. By using the exact Coulombic propagator, we assume that the wave function of Ps in a solid is a perturbation of the free Ps wave function. While this is a reasonable assumption in Ar, it remains to be seen whether it is appropriate in highly-polarizing solids [36].

3.2.2 Electron External Potential: Phenomenological Model

As described in Section 2.2.2, argon's 18 electrons completely fill its orbitals, so it is generally inert. An extra electron placed in the midst of Ar atoms will be attracted to the positive nucleus and repelled by the electrons. Due to Ar's polarizability, α, the Ar electrons will be pushed away from the extra electron, drawing it toward the nucleus. Because modeling this interaction between an electron and an atom is very complicated, we chose to begin with the Ar system because its details have been extensively studied.

Other researchers have examined the effect of placing an extra electron on an Ar atom and have developed a pseudopotential to describe this interaction [37]. This potential is a phenomenological form, which means that it is based on fitting parameters to experimental results, and we additionally include a polarization potential. By adding the polarization potential, which extends from the nucleus like $-\alpha/2r^4$, to the experimental results without polarization, which decrease from some constant value at the nucleus, we obtain a slight potential well. We can write the potential felt by an electron a distance r from a single Ar nucleus as

$$V_{-}(r) = -31.6712e^{-1.78984r} + 127.081e^{-2.2r} - \frac{5.43}{(r^2+0.7)^2}, \tag{3.10}$$

where the first two terms are phenomenological and the last is a method of calculating the polarization potential [37].

We assume that the potential due to many Ar atoms is approximately the superposition of the potential due to each Ar atom individually. The `clust_elec` subroutine in

our `cluster_elec` module creates a list of Ar atoms within 15 a.u. of the sphere that circumscribes the unit cell. These are assumed to be those atoms that would contribute to the potential of an electron within the unit cell. (For Ar without a monovacancy, this turns out to be 164 atoms.) The potential at each bead due to this cluster of atoms is determined by the `eval_pseudo` function in our `epot` module, which is not included in this thesis.

Figure 10 shows the potential felt by an electron in solid Ar using our method, and Figure 11 shows the potential felt by an electron in Ar with a monovacancy. The most energetically favorable regions for the electron are the lighter ones, and the dark circles are slices through the Ar atoms, which are prohibited to the electron. Note that the monovacancy is not as attractive to the electron as the areas closer to the crystal atoms, an effect due to the polarization potential.

3.2.3 Positron External Potential: Density Functional Theory (DFT)

The density of electrons and the potential felt by the positron can be found using density functional theory (DFT), a technique developed by Hohenberg, Kohn, and Sham in the 1960s for expressing the ground state properties a system as a function of the electron density. The basic theory behind this technique is described in Appendix C. As for the electron, there are competing effects due to the nuclei and the electrons of the Ar atoms, as well as a polarization potential of the form $-\alpha/2r^4$, which result in a potential due to a single Ar atom that is qualitatively similar to Woll's potential in Figure 6. The potential felt by a positron a distance r from a single Ar atom is given by

$$V_+(r) = \frac{18e^2}{r} - \int \frac{e^2\rho_-(\boldsymbol{r}')}{|\boldsymbol{r}' - \boldsymbol{r}|} d^3\boldsymbol{r}' - V_p(r), \tag{3.11}$$

where $V_p(r)$ is a polarization term based on a calculation by Gibson, which insures that $V_+(r)$ approaches $-\alpha/2r^4$ as $r \to \infty$ [38].

For this investigation, we began with a DFT-generated file containing the potential at each point in a $40 \times 40 \times 40$ spatial grid over the unit cell. To determine the potential at the location of a given bead in our PIMC code, we use a cubic spline fit to the data

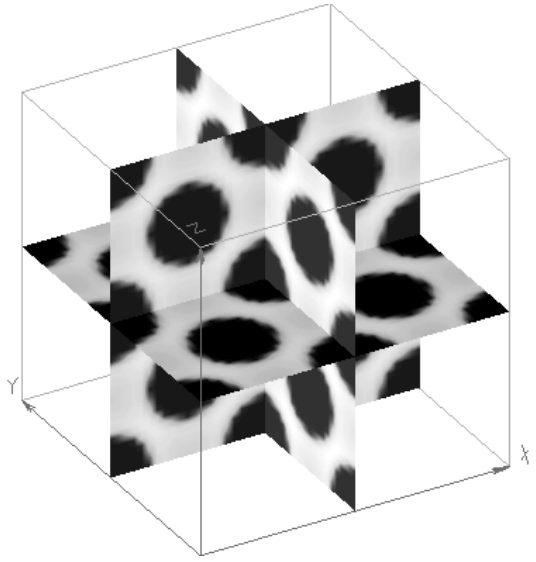

Figure 10: Slices through the potential felt by an electron in solid Ar. Eight unit cells are shown here. Light shading represents a region preferred by the electron, and dark shading represents an energetically unfavorable region. The black circles are slices through the Ar atoms, which are prohibited to the electron.

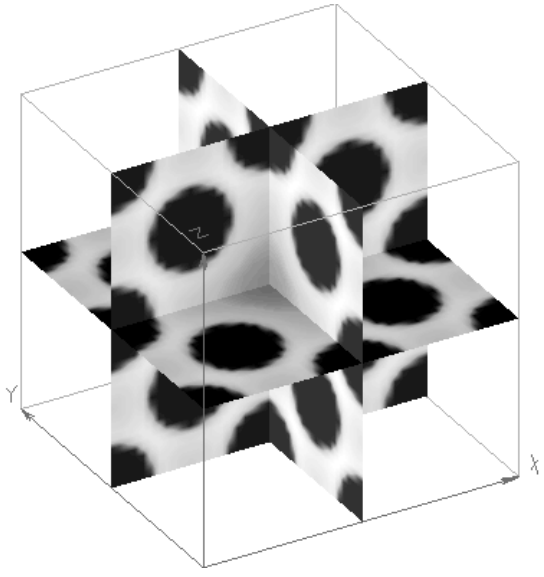

Figure 11: Slices through the potential felt by an electron in Ar with a monovacancy in the middle of an eight-unit-cell block. The monovacancy is not as attractive to the electron as areas closer to the crystal atoms.

stored in this file. This is accomplished using the `evalVcg` function in the module `vcGrid`, which is called at line 618 in Appendix E.

Figure 12 shows the potential felt by a positron in solid Ar, and Figure 13 shows the potential felt by a positron in Ar with a monovacancy. Note the similarities to Figures 10 and 11. The external potential felt by the electron and positron are not identical: for instance, the positron potential goes to infinity at the origin, whereas the electron potential goes to a finite number. In both cases, however, the monovacancy is less energetically attractive than the surrounding area, due to the polarization potential.

3.2.4 Kinetic Energy Estimators

For this investigation, we have used the "kinetic" or Barker estimate for the Kinetic energy, given by [39, 20]

$$\langle T_{\text{kin}} \rangle = \left\langle \frac{3P}{\beta} - \frac{mP}{2} \sum_{*=+,-} \sum_{i=1}^{P} \frac{\left(\boldsymbol{r}_{i-1}^{*} - \boldsymbol{r}_{i}^{*}\right)^2}{\hbar^2 \beta^2} \right\rangle . \tag{3.12}$$

Because we are subtracting two large numbers in the hope of obtaining approximately 0.25 a.u., we should not be surprised by the errors that result. Developing better methods of estimating the kinetic energy of Ps in a solid is an area for future investigation.

In our earlier work, we found that another estimator of the kinetic energy, the virial estimator, resulted in smaller errors for our Ps systems. The virial estimator uses the relationship between the average kinetic energy and the average potential energy to indirectly determine the kinetic energy, reminiscent of orbital motion problems in classical mechanics. Like the classical statement of the virial theorem, the virial estimator relates the kinetic energy to the gradient of the potential [40]:

$$\langle T_{\text{vir}} \rangle = \left\langle \frac{1}{2P} \sum_{*=+,-} \sum_{i=1}^{P} \boldsymbol{r}_i \cdot \nabla_i V(\boldsymbol{r}_i) \right\rangle . \tag{3.13}$$

Despite the virial estimator's success in the case of Ps in a hard cavity [20], we switched back to the kinetic estimator when we began modeling Ps in solids due to the greater difficulty of calculating the numerical derivative of the potential. The `vcGrid` module

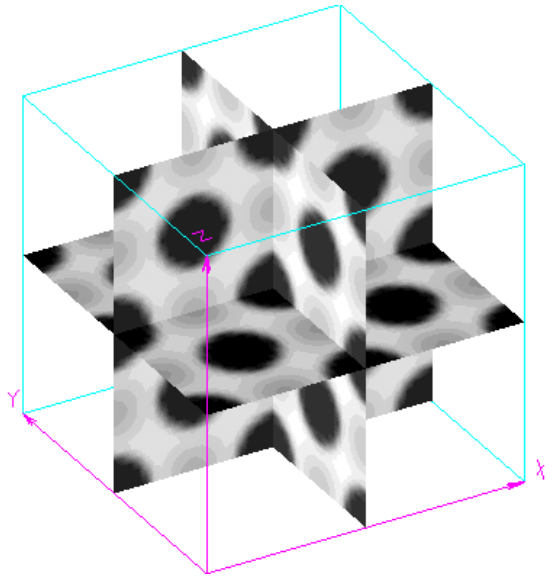

Figure 12: Slices through the potential felt by a positron in solid Ar. Eight unit cells are shown here.

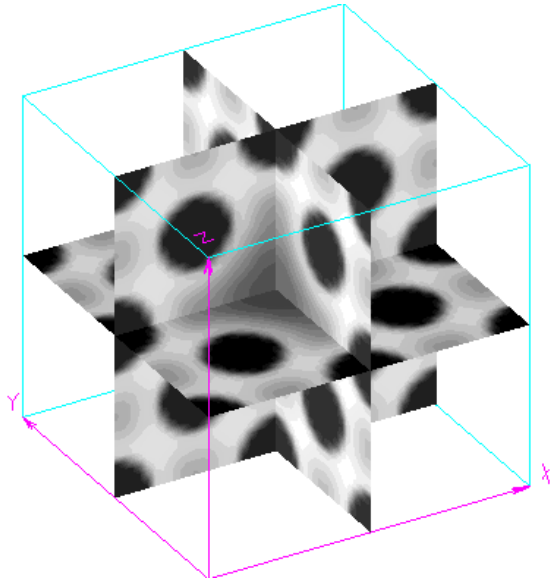

Figure 13: Slices through the potential felt by a positron in Ar with a monovacancy in the middle of an eight-unit-cell block.

for calculating the external potential felt by the positron currently contains a subroutine that can find the gradient of the potential, but we did not yet have an opportunity to test this procedure.

Another idea for estimating the kinetic energy uses the fact that by using the Pollock propagator, we know the Ps density matrix. From quantum statistical mechanics, we know that we can write the average energy of an ensemble as

$$\langle E \rangle = -\frac{1}{\beta}\frac{\partial \ln Z}{\partial \beta}. \tag{3.14}$$

The partition function, as we saw in Eq. (3.2), is simply the trace of the thermal density matrix, and there are tools in Roy Pollock's `Table` module that will allow us to calculate its derivative with respect to β [35]. This will hopefully be utilized in some future work.

3.3 Ps Lifetime: Enhancement Factor γ

The lifetime of Ps in a solid is easily measured experimentally and can reveal information about the solid (as in Figure 2), so it is important to determine accurately the Ps lifetime in our simulation. In general, the annihilation rate of Ps due to the surrounding solid (the pick-off annihilation rate) can be written as

$$\Gamma_{\text{p.o.}} = \tau^{-1} = \pi r_e^2 c \int d\boldsymbol{r}_- d\boldsymbol{r}_+ \rho_+(\boldsymbol{r}_+)\rho_-(\boldsymbol{r}_-)\gamma[\rho_-(\boldsymbol{r}_-)]\delta(\boldsymbol{r}_- - \boldsymbol{r}_+), \tag{3.15}$$

where $r_e = e^2/m_e c^2$ is the classical electron radius, $\rho_+(\boldsymbol{r}_+)$ is the positron density, $\rho_-(\boldsymbol{r}_-)$ is the density of Ar's electrons, and γ is the so-called enhancement factor, which is in general a functional of the electron density [41]. Setting $\gamma = 1$ gives the independent-particle model (IPM), which ignores exchange-correlation effects, resulting in lifetimes that are much longer than those observed in experiments.

M. J. Puska's research group in Finland has be working to develop better models for γ that more accurately reproduce experimental results. Their insulator model (IM),

which describes Ps in insulators such as Ar, gives the annihilation rate as

$$\Gamma_{\mathrm{IM}} = \Gamma_{\mathrm{IPM}} \left(1 + A + B\Omega \frac{(\epsilon - 1)}{(\epsilon + 2)} \right), \tag{3.16}$$

where $\epsilon = 1.66$ for Ar, $\Omega = 10.04^3/4$ is the unit cell volume, and $A = 0.684$ and $B = 0.0240$ are experimentally determined parameters [42]. For Ar, the term in parentheses in Eq. (3.16) is a correction factor of magnitude 2.78.

In our code, we calculate $\int \rho_+ \rho_- d^3r$ and multiply this by $\pi r_e^2 c = 50.469$ to obtain Γ_{IPM}. We then multiply Γ_{IPM} by 2.78 to obtain Γ_{IM}, which is the pick-off annihilation rate for Ps in Ar. If κ is also known for Ps in Ar, we can then calculate the total singlet and triplet annihilation rates using Eq. (2.24).

3.4 Pair Correlation Function $g(r)$

A complete description of the location of a positron in a system containing N crystal atoms requires a probability function of $3(N + 1)$ variables, $P(\boldsymbol{r}_+, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$. The theory of fluids offers a reduced description of this quantity, the pair correlation function $g(r)$, which gives the probability of observing any crystal atom a distance r from the positron. $g(r)$ is normalized by the number of atoms in the crystal, [43]

$$\int_0^\infty \rho g(r) 4\pi r^2 dr = N, \tag{3.17}$$

where $\rho = 4/a^3$ is the density of atoms in the fcc crystal lattice. Thus, $g(r)$ levels off at unity at distances far from the origin. We see from Eq. (3.17) that $N(r) \equiv \rho g(r) 4\pi r^2 dr$ is the number of atoms between r and $r + dr$ from the positron [43]. By measuring the distance between each bead and each crystal atom during every pass and placing these measurements in a histogram, a process known as "binning," we are able to graph this quantity, $N(r)$. The bead-atom pair correlation function is then

$$g(r) = \frac{N(r)}{\rho 4\pi r^2 \Delta r}, \tag{3.18}$$

where $4\pi r^2 \Delta r$ is the volume of each bin.

3.5 Practical Details: Hardware and Software

Our PIMC code was written in FORTRAN 90 and was run on a Linux workstation at LLNL as well as on Swarthmore's AppleSeed parallel supercomputing cluster. The AppleSeed cluster consists of about ten networked Macintosh G4 computers that share information using the Message Passing Interface (MPI) library [44]. Because our algorithm involves averaging many independent chain configurations, we parallelized our code by simply having each networked computer perform its own calculation, and then averaging the statistics. The computers then spend little time passing data or waiting for another computer to finish a calculation.

As seen in the code included in Appendix E, the modifications required for this parallelization scheme are minimal, making it easy to transfer to a Linux workstation. In the lines 1108-1112 of the `Initialize` subroutine, the "master" node connects to the "slave" computers to tell them to start running the PIMC program. After all computers have completed the main `MC_passes` loop of the program, the data is sent from the slave computers to the master node using the `MPI_SEND` and `MPI_RECV` commands, as seen in lines 285-435. Quantities that are best averaged, such as the radial distribution function, the Cartesian binning of beads in the lattice, and the pair correlation function $g(r)$ are added together by the master node. Other results, such as the averaged energy at each step and the annihilation rate, are left as individual estimates to give a better approximation for the error.

4 Results

We have examined Ps in defect-free fcc Ar and in Ar with a monovacancy, using the potentials that were illustrated in Figures 10-13. We have also simulated free Ps in order to compare these results with the theoretical energy from Eq. (2.11) and radial distribution function from Eq. (2.14). This thesis represents the first study, to our knowledge, of free Ps or Ps within a solid where the positron and electron are simulated as having independent degrees of freedom, using their exact Coulombic propagator. All of the Ps results presented here were collected from one million passes of our algorithm, after equilibrating for at least one million passes. Equilibration is necessary because the initial position of the beads is extremely unlikely to be a thermodynamically favorable one, and it would thus be incorrect to include measurements of these states in the statistical averages.

4.1 A Single Positron in Ar

Before examining the Ps results, we should consider how to determine the input parameters β and P. As seen in Eq. (3.5), P must be significantly larger than β for the PIMC approximations to be valid. In practice, the meaning of "significantly larger" is determined by experimenting with a system with known results, such as free Ps, to find the largest ratio of β/P that still yields accurate results. This ratio is known as the effective inverse temperature, β_{eff}, and we generally used $\frac{1}{8} \leq \beta_{\text{eff}} \leq \frac{1}{12}$.

Ar is only in the solid form below its melting temperature, which corresponds to values of β above 3895 a.u., as noted in Section 2.2.2. This high value of β, however, results in an undesirably long computation time because of the necessarily large values of P. To determine whether accurate simulations could be performed at higher temperatures, we calculated results at a variety of temperatures for a single positron in solid Ar. The computation time for a positron is significantly less than for Ps; the single chain means that the Pollock propagator and the electron external potential need not be calculated.

Statistics for the simulated positrons were collected over 100 to 200 thousand passes on each of seven AppleSeed computers, after allowing each system to equilibrate for 50

to 200 thousand passes. These seven runs were then used to obtain average measurements and uncertainties. For example, the lifetime results for the run at $\beta = 4000$ a.u., or $T = 82$ K, is shown in Figure 14.

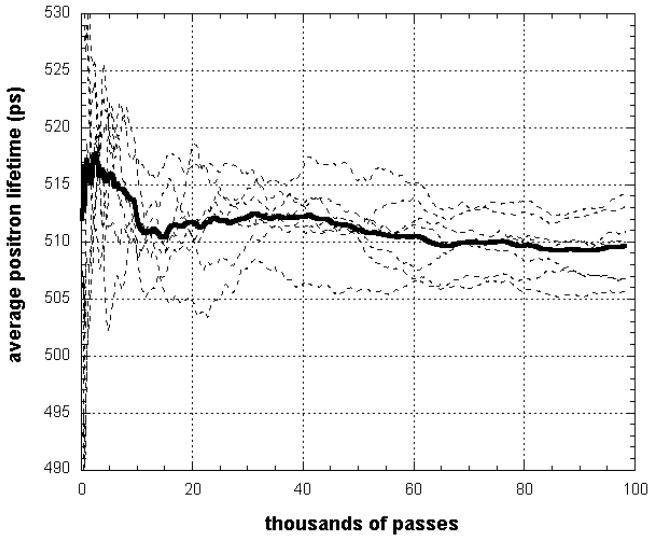

Figure 14: Cumulative average of positron lifetime from 100k passes of statistics, after 200k passes of equilibration. The thin dashed lines show results from the seven AppleSeed computers, and the thick solid line is their average. From this graph, we conclude that the lifetime is 510 ± 5 ps.

Table 1 shows the energy and lifetime from our positron simulation at three different temperatures, one of which is below the Ar melting point. It is clear that decreasing the

Table 1: Energy and Lifetime of a Positron in Solid Ar

P	β (a.u.)	Potential E (a.u.)	Lifetime (ps)
2k	400	-0.0850 ± 0.0005	509 ± 5
12k	2400	-0.0850 ± 0.0005	511 ± 5
20k	4000	-0.0850 ± 0.0005	510 ± 5

temperature (increasing β) has no significant impact on the important results of our simulations. Changing the temperature does have effects on the system (for example, the de Broglie wavelength of each particle scales as $1/\sqrt{T}$), but it seems that the particles sample the same regions of the crystal as we increase the temperature. We therefore chose β around 100 to 400 a.u. for our Ps measurements.

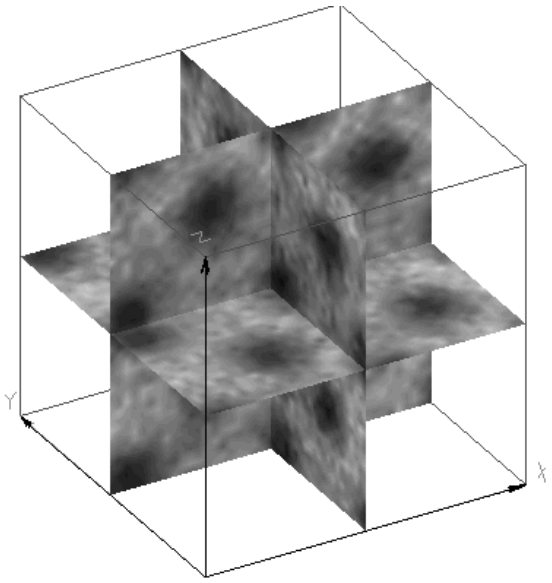

Figure 15: Positron density in fcc Ar at $\beta = 400$ a.u. Light areas represent high positron density. The dark areas correspond to the locations of the Ar atoms, where the positron density is expected to be zero.

Since we are modeling a single positron that is not bound to any particular electron, it will have to annihilate via "pick-off" annihilation. We can thus use the pick-off annihilation rate calculated by our code to find the lifetime of a single positron in Ar, as shown in Table 1. These results are somewhat higher than Liu and Robert's experimental lifetime of 435 ps [29] or Jean, Yu, and Zhou's experimental lifetime of 340-390 ps.

We can also compare the potential felt by our positron in Ar, $-0.0850 = 2.31$ eV, to the calculations in the literature. Our result is in good agreement with two results which come from positron DFT with slightly different potential models for the polarization of Ar: 2.13 eV and 3.32 eV [22]. All of these theoretical results are slightly higher than the experimental measurement of 1.55 ± 0.05 eV [45].

Figure 15 shows the positron distribution throughout the Ar lattice at $\beta = 400$ a.u. It is clearly sampling the entire available area, while avoiding the Ar atoms. Comparing this to Figure 12 shows that the simulated positron prefers areas of lowest potential, as expected. Because a single positron samples much more space inside the Ar lattice than

a Ps atom, it is hard to obtain such clear results from binning the three-dimensional Ps distribution. For this reason, we calculate the pair correlation function $g(r)$, as will be discussed in Section 4.2.2.

4.2 Ps in Ar with and without a Monovacancy

4.2.1 Radial Distribution and κ

Figure 16 shows the radial probability density for the relative coordinate between the electron and positron, $P(r)$, which is the likelihood of finding the particles separated by a distance r. The solid line gives the $1S$ exact theory from Eq. (2.14), which closely matches the results from our simulation of free Ps, shown by the diamonds. Interestingly, the crosses and circles, which represent Ps in Ar with and without a monovacancy, respectively, are indistinguishable.

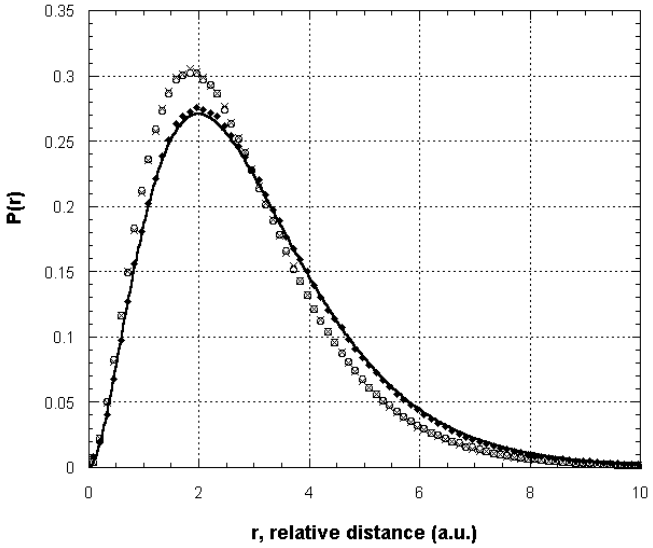

Figure 16: Radial distribution function, $P(r)$, for Ps in Ar. r is the relative coordinate between the positron and electron. Solid line: $1S$ exact theory for free Ps. Diamonds: Free Ps simulated by our code. Circles: Ps in fcc Ar. Crosses: Ps in Argon with a monovacancy.

Figure 17 shows $P(r)/r^2 = 4\pi|\psi(r)|^2$ for our different simulations. From Eq. (2.14),

we see that the $1S$ theory for free Ps is

$$\frac{P(r)}{r^2} = \frac{1}{2}e^{-r} \tag{4.1}$$

in atomic units; this is shown by the solid line in Fig. 17. Our simulation of free Ps

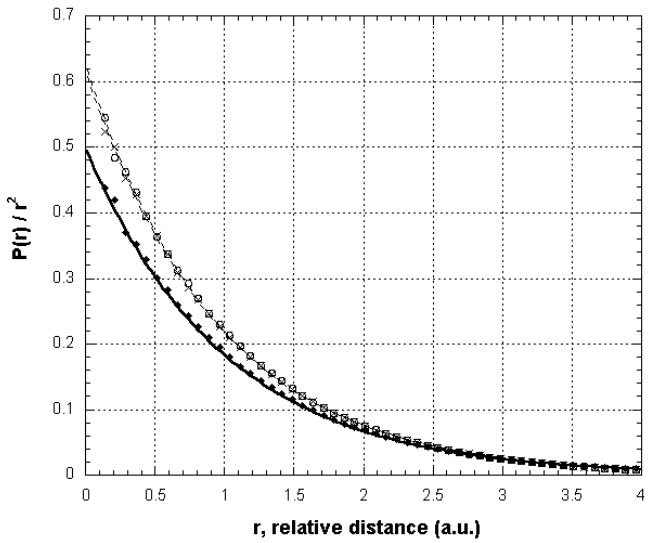

Figure 17: Radial distribution function, $P(r)$, divided by r^2. Solid line: $1S$ exact theory. Diamonds: Free Ps simulation. Circles: Ps in fcc Ar. Crosses: Ps in Ar with a monovacancy. Solid and dashed lines are curve fits: results for free Ps were fit to $0.50e^{-1.00r}$, the results for Ps in fcc Ar were fit to $0.63e^{-1.05r}$, and the results for Ps in Ar with a monovacancy were fit to $0.62e^{-1.05r}$, where all uncertainties are in the last digit.

exactly matches this theory; the best fit to our results was $P(r)/r^2 = 0.50e^{-1.00r}$. The wavefunction for Ps in Ar is squeezed, just as for Ps in a hard spherical cavity [20]; fitting the results for fcc Ar and Ar with a monovacancy gave $P(r)/r^2 = 0.63e^{-1.05r}$ and $P(r)/r^2 = 0.62e^{-1.05r}$, respectively. From these curve fit parameters we calculate the internal contact density of Ps in Ar to be approximately

$$\kappa_{\mathrm{Ar}} = \frac{|\psi_{\mathrm{Ar}}(0)|^2}{|\psi_{\mathrm{free}}(0)|^2} = \frac{0.625}{0.50} = 1.25. \tag{4.2}$$

This value of κ can be used in Eq. (2.24) to find the total annihilation rate of o-Ps in Ar, as will be see in Section 4.2.3. The effect of κ on the annihilation rate may not be

experimentally observable, since the pick-off annihilation rate $\Gamma_{\text{p.o.}}$ is much larger than the Γ_0^t. κ can be measured in a clever way, however, from the hyperfine interaction, as Section 2.2.1 and Appendix D show.

In the materials in which κ has thus far been measured, it has always been found to be less than one [10, 30]. Given our potential, it makes sense that the Ps wavefunction should be compressed, and that κ should be slightly greater than one. Using the magnetic quenching method described in Appendix D, experiments could be performed to check our result. We also plan to check our methods by simulating Ps in α-SiO_2, in which κ has already been measured [30].

4.2.2 Pair Correlation Function

Figure 18 shows the bead-atom correlation function for the positron in Ps, in fcc Ar and Ar with a monovacancy, calculated using Eq. (3.18). As is to be expected, there is no probability of finding a positron in the same location as an atom, and $g(r)$ levels off to unity far from the origin, as described in Section 3.4. Although the Ps atom is delocalized, just like an atom in a fluid, it has certain preferred spots that can be ascertained from the peaks in $g(r)$. We see, for example, that the positron is very likely to be 4 a.u. or 10 a.u. away from an Ar atom, but less likely to be 7 a.u. away.

From the similarity of the $g(r)$ function for Ps in Ar with and without a monovacancy, we see that the Ps atom is likely to be found in similar locations in these two crystals, as suggested by the slices through the potential files seen in Figures 12 and 13. The two $g(r)$ functions are not, however, identical. In Ar with a monovacancy, the bead-atom $g(r)$ function (shown by the crosses in Figure 18) has a higher first peak around 4 a.u. than the $g(r)$ function in fcc Ar, and then it levels off more quickly. This suggests that in Ar with a monovacancy, Ps does not move around quite as much as it does in fcc Ar, perhaps because it is able to rest in the potential well due to a single Ar atom rather than being pulled between two neighboring atoms.

To better understand what our $g(r)$ function tells us about the location of Ps in Ar, we can compare it to what we would see if a point Ps were fixed to a specific location in the lattice. Since the Ar atoms are already fixed in a periodic lattice, fixing Ps would

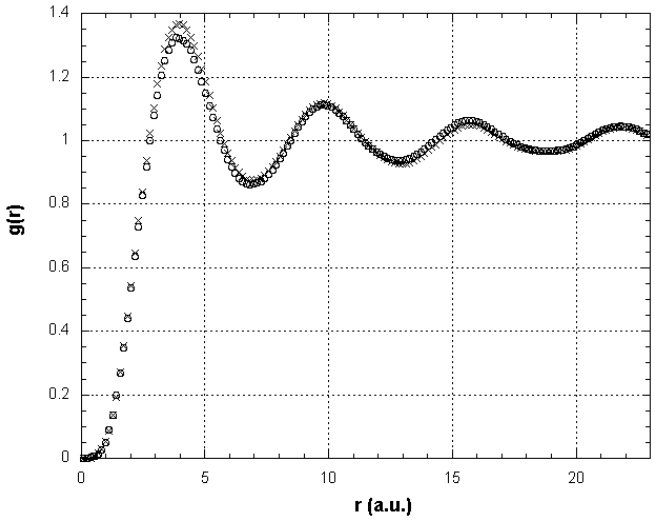

Figure 18: Pair correlation function $g(r)$ for Ps in Ar. Circles: Ps in fcc Ar. Crosses: Ps in Argon with a monovacancy.

cause $g(r)$ to have sharp peaks instead of the gentle bumps seen in Figure 18. Figure 19 shows what $g(r)$ would look like if Ps were fixed at one of the potential minima, which turn out to be directly between two adjacent Ar atoms. For example, one potential minimum is located at $(10.04/4, 10.04/4, 0)$. The sharp peak at the beginning of this pair correlation function, just before 4 a.u., is due to the two closest atoms, which are each $\sqrt{2 \times (10.04/4)^2} = 3.55$ a.u. away. From the first peak in the calculated $g(r)$ in Figure 18, we know that Ps is very likely to be found about 4 a.u. away from an Ar atom, so this potential minimum is a probable location.

Figure 20 shows what $g(r)$ would look like if a point Ps were fixed at the monovacancy in Ar. The first sharp peak is due to the atoms on the closest faces, which are each $\sqrt{2 \times (10.04/2)^2} = 7.10$ a.u. away. If the center of the monovacancy were the preferred location of Ps, we would not see the first peak in Figure 18 at 4 a.u. We conclude that Ps avoids the monovacancy.

It is likely that the monovacancy is not attractive to Ps because of the polarization potential, which causes it to remain close to the Ar atoms. This is reminiscent of the findings of Woll over thirty years ago; although his model is somewhat simplistic, he correctly found that the positron is not simply repelled from the Ar atoms, collecting

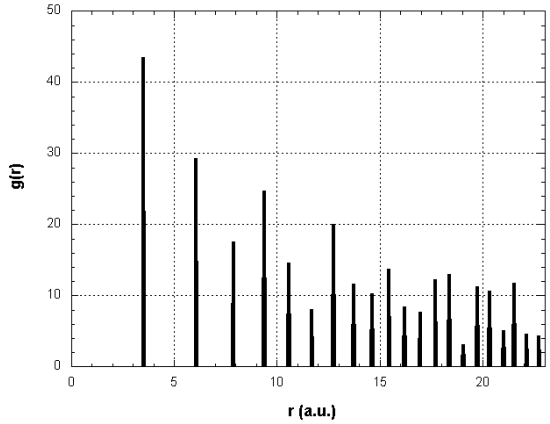

Figure 19: This is what $g(r)$ would look like if a point-sized Ps atom were fixed to the potential minimum at (2.51 a.u., 2.51 a.u., 0.0). Because the first sharp peak occurs around the same location as the first peak in the actual $g(r)$ for Ps in Ar with or without a monovacancy, Ps probably spends a lot of time at this location in the lattice.

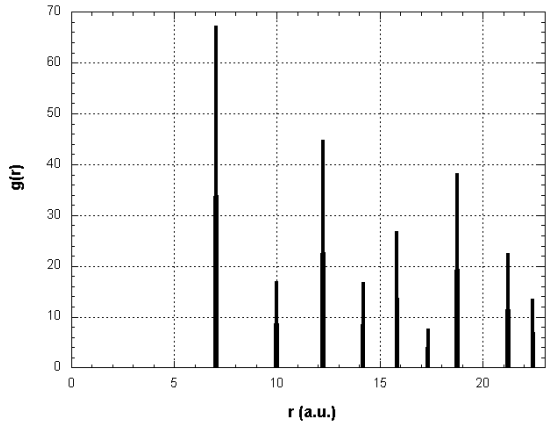

Figure 20: This is what $g(r)$ would look like if the a point Ps atom were pinned at the monovacancy in Ar. We see that the closest Ar atom is 7 a.u. away. Since the first peak in Figure 18 is around 4 a.u., we again see that the monovacancy is not attractive to Ps.

in voids or defects. Rather, there is a potential well around each Ar atom for both the electron and the positron, and, in a lattice of this size, there is no force driving Ps into the monovacancy.

4.2.3 Energy and Lifetimes

Table 2 presents several results for the energy of free Ps. Although the Pollock propagator was used for determining whether to accept a new configuration in the PIMC algorithm, the Yukawa potential, with the parameter a from Eq. (3.7) set to 0.1, was used for calculating the potential energies shown here. This will be corrected in future investigations, but it does not seem to limit the accuracy of the results; the potential energy measurements for free Ps are very close to the expected -0.5 a.u. The kinetic energies, however, are systematically high by about 0.05 a.u., since we expect to add the kinetic and potential energies to obtain a total energy of -0.25 a.u., as seen from Eq. (2.11).

Table 2: Energy of free Ps

P	β (a.u.)	Energy (a.u.)	Potential E (a.u.)
1k	100	-0.208	-0.500
1k	100	-0.209	-0.503
1k	100	-0.212	-0.500

Table 3 gives the energy and inverse lifetime for several simulations of Ps in fcc Ar, and Table 4 gives the same quantities for Ps in Ar with a monovacancy. Though it is necessary to generate more results, it seems that Ps in Ar with a monovacancy has a slightly longer lifetime than Ps in fcc Ar.

Table 3: Energy and Lifetime of Ps in FCC Ar

P	β (a.u.)	Energy (a.u.)	Potential E (a.u.)	$\Gamma_{\text{p.o.}}$ (ns^{-1})
4k	400	-0.322	-0.699	1.45
4k	400	-0.265	-0.694	1.45
4k	400	-0.306	-0.697	1.42
1k	100	-0.316	-0.702	1.40
500	50	-0.288	-0.710	1.38

As mentioned in Section 2.2.2, Gullikson and Mills noted that the long o-Ps lifetime in solid Ar measured by Jean, Yu, and Zhou is present even when there should be no vacancies in the sample [27]. We also see little difference in our calculated lifetimes

Table 4: Energy and Lifetime of Ps in Ar with a Monovacancy

P	β (a.u.)	Energy (a.u.)	Potential E (a.u.)	$\Gamma_{\text{p.o.}}$ (ns^{-1})
4k	400	-0.328	-0.679	1.27
4k	400	-0.311	-0.686	1.31
4k	400	-0.316	-0.674	1.23
1k	100	-0.305	-0.697	1.38
500	50	-0.276	-0.701	1.33

for Ps in Ar with or without a monovacancy, although the numbers we obtain are somewhat different from experiment, as will be discussed presently.

By putting the value of κ from Eq. (4.2), the accepted experimental result of 0.007 ns^{-1} for Γ_0^t, and our calculated pick-off annihilation rate $\Gamma_{\text{p.o.}}$ into Eq. (2.24), we can now determine the annihilation rate of o-Ps in Ar. Averaging the results for $\Gamma_{\text{p.o.}}$ in fcc Ar, we obtain

$$\begin{aligned}\Gamma^t =& \kappa\Gamma_0^t + \Gamma_{\text{p.o.}} \\ =& 1.25 \times 0.007\,\text{ns}^{-1} + 1.42\,\text{ns}^{-1} \\ =& 1.43\,\text{ns}^{-1},\end{aligned} \tag{4.3}$$

which corresponds to a lifetime of 700 ps; the standard deviation from the different trials gives an error of ± 15 ps. A similar analysis results in a lifetime of 760 ± 30 ps for Ar with a monovacancy.

We can also use Eq. (2.24) to determine the p-Ps annihilation rate in Ar. Since $\Gamma_0^s = 8\ \text{ns}^{-1}$ is much larger than Γ_0^t, the modified κ in Ar will significantly change the total annihilation rate for the singlet state in a way it did not for the triplet state. We obtain

$$\begin{aligned}\Gamma^s =& \kappa\Gamma_0^s + \Gamma_{\text{p.o.}} \\ =& 1.25 \times 8\,\text{ns}^{-1} + 1.42\,\text{ns}^{-1} \\ =& 11.02\,\text{ns}^{-1},\end{aligned} \tag{4.4}$$

which corresponds to a lifetime of 86 ± 1 ps. Performing the same analysis in Ar with

a monovacancy, we obtain 88 ± 1 ps.

We can compare our results of 510 ps for positron annihilation and 700–760 ps for o-Ps annihilation to the experimental measurements of Jean, Yu, and Zhou. They report three different signals: $\tau_1 = 125$ ps, $\tau_2 \approx 340$–390 ps, and $\tau_3 \approx 2.1$–2.5 ns [26]. They assumed that the p-Ps lifetime would not change significantly in Ar, so τ_1 was fixed by them at 125 ps while they fit their data to find τ_2 and τ_3. Our results suggest, however, that p-Ps does have a shorter lifetime in Ar, so it would be interesting to look for a shorter lifetime in their data. The second lifetime, τ_2, is lower than either of the lifetime ranges we found; they claim that it is due entirely to the annihilation of single positrons. The third is higher than could be expected to exist in defect-free Ar (as well as higher than our own calculations), so they claim that it is due to o-Ps self-trapping in voids. This explanation has been rejected, however, both by Gullikson and Mills [27] and by ourselves, and no one has offered a better interpretation for these long lifetimes.

Although Jean, Yu, and Zhou claim that o-Ps does not exist in a regular Ar lattice, we have found that there is sufficient space for o-Ps to sit inside solid Ar. It is possible that their τ_2 lifetime is actually due to a mixture of positrons and o-Ps. Still, our lifetime measurements of around 700 ps exceed this value. We may find that our potential models need to be adjusted, or that we must perform a more self-consistent calculation by allowing the electric charge or Ar nuclei to slightly adjust their positions around Ps. Also, since the insulator model we are using for γ results in longer lifetimes than any other model (except the IPM), any adjustments to our model for γ would reduce our lifetime measurements, perhaps resulting in better agreement with experiment. Developing a better model for γ in an insulator, as well as performing simulations in other materials, will help resolve these issues in future work.

While our lifetime calculations are somewhat different from those found in experiments, it is noteworthy that we find that the pick-off annihilation rate for a single positron (1.95 ns^{-1}) is higher than for Ps (1.42 ns^{-1}) in solid Ar. Thus, o-Ps has a longer lifetime than a bare positron in a material. It is reasonable to expect that a positron in a bound state with an electron will avoid the other electrons in the solid more than a bare positron, which only feels an attractive force toward electrons. This

result also supports the assertions of experimentalists, who always tend to attribute the longest lifetime in their positron lifetime spectra to o-Ps decay.

5 Conclusions and Future Directions

We have demonstrated that a path integral Monte Carlo simulation can effectively model Ps in an insulating solid. The Pollock propagator for simulating the Coulomb potential is efficient and accurately reproduces the theoretical results for ground state free Ps. Using data generated by code based on density functional theory, we have modeled the external potential caused by solid Ar and found that the Ps wavefunction is squeezed inside this solid, with an internal contact density of $\kappa_{\mathrm{Ar}} = 1.25 \pm 0.02$. This means that the hyperfine splitting energy of Ps in Ar is 1.25 times greater than that of free Ps. We have also demonstrated that when the Ar lattice has a monovacancy (which we accomplished by creating a missing atom in the middle of each cube of eight unit cells), the Ps does not fall into the vacancy. Rather, due to the polarization potential, Ps remains near the Ar atoms.

The lifetime of o-Ps in Ar with a monovacancy was calculated to be 760 ± 30 ps, a slight increase from the calculation of 700 ± 15 ps for solid Ar. In support of the experimental work of Rice-Evans *et al.* [25], we found that there is sufficient room for o-Ps to exist in fcc Ar, contradicting the claim of Jean, Yu, and Zhou [26] and suggesting that the shorter observed lifetimes may encompass a mixturee of o-Ps and single positron decay. We also find the lifetime of a single positron in solid Ar to be 510 ± 5 ps, supporting the prediction that a single positron is more likely to be picked off by an electron from the crystal than a bound positron is. Contrary to the assumption that the p-Ps lifetime remains constant at 125 ps in Ar [26], we find that it is decreased to under 90 ps.

Future investigations will involve improving our estimation of the kinetic energy and simulating Ps inside other many other materials, including silica, sodalite, and other zeolites. Determining the potential felt by the electron is difficult, but we have already been able to model a single positron inside α-SiO_2. Once we have accurately simulated the external potential felt by an electron in this material, we will be able to measure the hyperfine splitting of Ps to compare with experimental results [30].

A Radial Schrödinger Equation for Coulombic Potential

In Eq. (2.5), separation of variables for the Schrödinger equation leads to the radial equation

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) = \left[\frac{\ell(\ell+1)}{r^2} - \frac{m}{\hbar^2}\left(\frac{e^2}{r}+E\right)\right]R. \tag{A.1}$$

This equation is more easily solved by defining a function $U(r) = rR(r)$, in which case it becomes

$$\left[\frac{d^2}{dr^2} + \frac{m}{\hbar^2}\left(\frac{e^2}{r}+E\right) - \frac{\ell(\ell+1)}{r^2}\right]U(r) = 0. \tag{A.2}$$

Replacing r with the dimensionless variable ρ,

$$\rho = \frac{2\sqrt{m_e|E|}}{\hbar}r, \tag{A.3}$$

and defining

$$\lambda = \frac{e^2}{2\hbar}\sqrt{\frac{m}{|E|}}, \tag{A.4}$$

Eq. (A.2) becomes

$$\left[\frac{d^2}{d\rho^2} + \frac{\lambda}{\rho} - \frac{1}{4} - \frac{\ell(\ell+1)}{\rho^2}\right]U(\rho) = 0, \tag{A.5}$$

where we are interested in bound state energies $E = -|E|$.

As $\rho \to \infty$, this becomes

$$\left[\frac{d^2}{d\rho^2} - \frac{1}{4}\right]U(\rho) = 0, \tag{A.6}$$

which has the general solution $Ae^{-\rho/2} + Be^{\rho/2}$. Because the equation must be normalizable, B must be zero. This suggests a solution of the form $U(\rho) = e^{-\rho/2}F(\rho)$, for

which Eq. (A.5) becomes

$$\left[\frac{d^2}{d\rho^2} - \frac{d}{d\rho} + \frac{\lambda}{\rho} - \frac{\ell(\ell+1)}{\rho^2}\right] F(\rho) = 0. \tag{A.7}$$

Substituting in the Frobenius series $F(\rho) = \rho^s \sum_{k=0}^{\infty} c_k \rho^k$ gives

$$\sum_{k=0}^{\infty} c_k \rho^{k+s-2} \left[(k+s)(k+s-1) - \ell(\ell+1)\right] + \sum_{k=0}^{\infty} c_k \rho^{k+s-1} \left[\lambda - k - s\right]. \tag{A.8}$$

The coefficient of each power of ρ must separately equal zero, and the coefficient of the lowest power of ρ is $s(s-1) - \ell(\ell+1)$, which is zero for $s = \ell + 1$ or $s = -\ell$. Since R must be finite at the origin, the latter solution is rejected, and s is replaced by $\ell + 1$. Forcing the other coefficients to be zero leads to the recursion relation

$$\frac{c_{k+1}}{c_k} = \frac{k + \ell + 1 - \lambda}{(k+1)(k+2\ell+2)}. \tag{A.9}$$

If the series does not terminate, c_{k+1}/c_k goes to $1/k$ for large k, so $U(\rho)$ goes to $\rho^{\ell+1+\rho} e^{\rho} e^{-\rho/2}$ for large ρ, which is not normalizable. Thus, to satisfy the boundary conditions, the series must terminate: λ must equal $k + \ell + 1$ for some k. This is the origin of the principle quantum number,

$$n = \lambda = 1, 2, 3, 4, \ldots . \tag{A.10}$$

Then ℓ, the azimuthal quantum number, is constrained to be between 0 and $n-1$. The energy levels, as defined in Eq. (A.4) are thus quantized:

$$|E_n| = \frac{m_e e^4}{4\hbar^2 n^2}. \tag{A.11}$$

We can now write ρ as

$$\rho = r\left(\frac{4m_e\,|E|}{\hbar^2}\right)^{1/2} = \frac{r m_e e^2}{\hbar^2 n} = \frac{r}{a_0 n}, \tag{A.12}$$

in terms of the Bohr radius, $a_0 = \hbar^2/m_e e^2$.

To find the radial wave functions, substitute $F(\rho) = \rho^{\ell+1}G(\rho)$ into Eq. (A.7) to obtain

$$\left[\rho\frac{d^2}{d\rho^2} + (2\ell + 2 - \rho)\frac{d}{d\rho} + (n - \ell - 1)\right] G(\rho) = 0. \tag{A.13}$$

This is Laguerre's associated differential equation, and its solutions are the associated Laguerre polynomials, $L_{n+\ell}^{2\ell+1}(\rho)$ [46]. The full radial wave functions for positronium are thus

$$R_{n\ell}(\rho) = N_{n\ell}e^{-\rho/2}\rho^{\ell}L_{n+\ell}^{2\ell+1}(\rho), \tag{A.14}$$

where $N_{n\ell}$ is a normalization factor given by [46]

$$\left(\frac{1}{N_{n\ell}}\right)^{1/2} = \int_0^\infty e^{-\rho}\rho^{2\ell}\left[L_{n+\ell}^{2\ell+1}(\rho)\right]^2\rho^2 d\rho = \frac{2n\left[(n+\ell)!\right]^3}{(n-\ell-1)!}. \tag{A.15}$$

B Partition Function for Classical Ring Polymer

Consider the situation in Figure 21 where we have P beads, connected in a ring by springs of constant k. Suppose that these beads are constrained to move in one dimension (along a circle, for instance) and that each bead additionally feels a potential $\frac{V(x)}{P}$. The energy of this ensemble consists of the kinetic energy of the beads, the potential

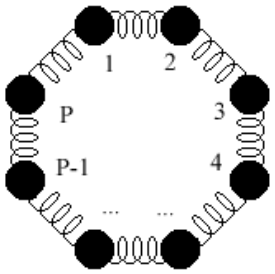

Figure 21: Model of a classical ring polymer of P beads coupled by harmonic springs of constant k, where each bead feels a potential $\frac{V(x)}{P}$.

energy stored in the springs, and the energy due to the external potential V. We can thus write down the energy as

$$\begin{aligned} E &= \left[\frac{p_1^2}{2m} + \cdots + \frac{p_P^2}{2m}\right] + \left[\frac{k}{2}(x_1 - x_2)^2 + \cdots + \frac{k}{2}(x_P - x_1)^2\right] + \left[\frac{V(x_1)}{P} + \cdots + \frac{V(x_P)}{P}\right] \\ &= \left[\sum_{i=1}^{P} \frac{p_i^2}{2m}\right] + \left[\frac{k}{2}\sum_{i=1}^{P}(x_i - x_{i+1})^2 + \frac{1}{P}\sum_{i=1}^{P} V(x_i)\right], \end{aligned} \tag{B.1}$$

where we define $x_{P+1} \equiv x_1$. The semiclassical partition function (using Planck's constant h as the dimension of a cube of classical phase space), can be written as

$$Z = \int \cdots \int \exp(-\beta E) \frac{dp_1 \cdots dp_P dx_1 \cdots dx_P}{h^P}, \tag{B.2}$$

where $\beta = \frac{1}{kT}$ is the inverse temperature. Since the exponential of a sum is the product of exponentials, we can pull out the kinetic energy terms as a product of P idential

integrals, each equal to

$$\int_{-\infty}^{\infty} \exp\left(-\frac{\beta p^2}{2m}\right) dp = \left(\frac{2\pi m}{\beta}\right)^{\frac{1}{2}} = \left(\frac{m}{2\pi\beta\hbar}\right)^{\frac{1}{2}}. \tag{B.3}$$

The partition function in Eq. (B.2) thus becomes

$$Z = \left(\frac{m}{2\pi\beta\hbar^2}\right)^{\frac{P}{2}} \int dx_1 \cdots dx_P \exp\left[-\beta\left(\frac{k}{2}\sum_{i=1}^{P}(x_i - x_{i+1})^2 + \frac{1}{P}\sum_{i=1}^{P} V(x_i)\right)\right]. \tag{B.4}$$

Aside from a difference in prefactor, this is equivalent to the partition function for a single particle under the potential $V(x)$, seen in Eq. (3.4), if we set $k = \frac{mP}{\beta^2\hbar^2}$. It is this map from a quantum system to a classical system that is at the heart of PIMC.

C The Basics of Density Functional Theory (DFT)

If we are considering a system of N electrons, this means that we are reducing an expression of $3N$ variables to an expression of only three variables, which is a significant simplification. The justification for this will be discussed, but first let us consider how to write the energy functional.

The energy of a system can be broken into three parts: the kinetic energy, the electrostatic interactions of the electrons and nuclei, and the external potential energy,

$$E[\rho(\boldsymbol{r})] = T[\rho(\boldsymbol{r})] + E_{\mathrm{es}}[\rho(\boldsymbol{r})] + E_{\mathrm{ext}}[\rho(\boldsymbol{r})], \tag{C.5}$$

where the external potential is of the form $E_{\mathrm{ext}} = \int \hat{V}_{\mathrm{ext}}(\boldsymbol{r})\rho(\boldsymbol{r})d^3\boldsymbol{r}$ and the electrostatic term includes the Coulombic electron-electron and electron-nuclei interactions. Earlier density functional theories developed by Thomas and Fermi in the 1920s had failed due to their approximation of the kinetic energy based on a homogeneous electron gas known as jellium. Kohn and Sham succeeded because of the introduction of Kohn-Sham orbitals, which are the orthonormal single-electron wavefunctions that would be exact if the electrons did not interact. The electron density can be expressed as a linear combination of them,

$$\rho(\boldsymbol{r}) = \sum_{n=1}^{N} a_n |\phi(\boldsymbol{r})|^2, \tag{C.6}$$

and an approximation to the kinetic energy is then [48]

$$T_{\mathrm{s}}[\rho(\boldsymbol{r})] = -\frac{\hbar^2}{2m}\sum_{n=1}^{N} a_n \phi_n^*(\boldsymbol{r})\nabla^2\phi_n(\boldsymbol{r})d^3\boldsymbol{r}. \tag{C.7}$$

Similarly, we cannot write down the exact electronic part of $E_{\mathrm{es}}[\rho(\boldsymbol{r})]$, but we can use the approximation

$$E_{\mathrm{e-e}}[\rho(\boldsymbol{r})] \approx J[\rho(\boldsymbol{r})] \equiv \frac{1}{2}\int \frac{\rho(\boldsymbol{r})\rho(\boldsymbol{r'})}{|\boldsymbol{r}-\boldsymbol{r'}|}d^3\boldsymbol{r}d^3\boldsymbol{r'}. \tag{C.8}$$

This is the direct Coulomb interaction, neglecting the Pauli exclusion principle which

states that two electrons cannot occupy the same position [47]. The above approximations are grouped in a new term, the so-called exchange correlation energy:

$$E_{\mathrm{xc}}[\rho(\boldsymbol{r})] = (T[\rho(\boldsymbol{r})] - T_{\mathrm{s}}[\rho(\boldsymbol{r})]) + (E_{\mathrm{e-e}}[\rho(\boldsymbol{r})] - J[\rho(\boldsymbol{r})])\,. \tag{C.9}$$

The exchange correlation energy is calculated using the local density approximation (LDA), for which we assume that the response of an electron at a given point can be approximated by the response in a jellium of the same electron density [48]. With these changes, Eq. (C.5) becomes

$$E[\rho(\boldsymbol{r})] = T_{\mathrm{s}}[\rho(\boldsymbol{r})] + E_{\mathrm{e-n}}[\rho(\boldsymbol{r})] + J[\rho(\boldsymbol{r})] + E_{\mathrm{xc}}[\rho(\boldsymbol{r})] + E_{\mathrm{ext}}[\rho(\boldsymbol{r})]. \tag{C.10}$$

Writing the energy as a functional of the electronic density is justified by one of the Hohenberg-Kohn theorems, which states that given an external potential $\hat{V}_{\mathrm{ext}}(\boldsymbol{r})$, the ground state electron density $\rho(\boldsymbol{r})$ is a unique function of $\hat{H}_0 + \hat{V}_{\mathrm{ext}}$ [47]. To prove this, suppose not, so that two potentials, $\hat{V}_{\mathrm{ext}}$ and $\hat{V}'_{\mathrm{ext}}$ both give the same ground state electronic density. Since they are different potentials, they have different ground states and ground-state energies, given by

$$\begin{aligned} \left(\hat{H}_0 + \hat{V}_{\mathrm{ext}}\right) |\,\psi\rangle &= E\,|\,\psi\rangle \\ \left(\hat{H}_0 + \hat{V}'_{\mathrm{ext}}\right) |\,\psi'\rangle &= E'\,|\,\psi'\rangle\,. \end{aligned} \tag{C.11}$$

Since $|\,\psi'\rangle$ is the ground state of $\hat{H}_0 + \hat{V}'_{\mathrm{ext}}$, $E' = \left\langle \psi' \middle| \hat{H}_0 + \hat{V}'_{\mathrm{ext}} \middle| \psi' \right\rangle$ is strictly less than the energy of any other state under this Hamiltonian. Specifically, E' is less than

$$\begin{aligned} \left\langle \psi \middle| \hat{H}_0 + \hat{V'}_{\mathrm{ext}} \middle| \psi \right\rangle &= \left\langle \psi \middle| \hat{H}_0 + \hat{V}_{\mathrm{ext}} - \hat{V}_{\mathrm{ext}} + \hat{V'}_{\mathrm{ext}} \middle| \psi \right\rangle \\ &= E + \left\langle \psi \middle| \hat{V'}_{\mathrm{ext}} - \hat{V}_{\mathrm{ext}} \middle| \psi \right\rangle \\ &= E + \int \rho(\boldsymbol{r}) \left[\hat{V'}_{\mathrm{ext}} - \hat{V}_{\mathrm{ext}}\right] d^3\boldsymbol{r}, \end{aligned} \tag{C.12}$$

so we have

$$E' < E + \int \rho(\boldsymbol{r}) \left[\hat{V'}_{\text{ext}} - \hat{V}_{\text{ext}} \right] d^3\boldsymbol{r}. \tag{C.13}$$

Performing the same operations for the unprimed Hamiltonian and ground state energy, we see that

$$E < E' + \int \rho(\boldsymbol{r}) \left[\hat{V}_{\text{ext}} - \hat{V'}_{\text{ext}} \right] d^3\boldsymbol{r}. \tag{C.14}$$

Adding Equations (C.13) and (C.14) gives

$$E + E' < E + E' + \int \rho(\boldsymbol{r}) \left[\hat{V'}_{\text{ext}} - \hat{V}_{\text{ext}} \right] d^3\boldsymbol{r} + \int \rho(\boldsymbol{r}) \left[\hat{V}_{\text{ext}} - \hat{V'}_{\text{ext}} \right] d^3\boldsymbol{r}. \tag{C.15}$$

But since the electron density is the same, the last two terms on the right cancel, giving $E + E' < E + E'$, a contradiction. We thus see that the ground state electron density is uniquely determined by the external potential, and thus the external potential can be determined from a given ground state electron density.

By minimizing the energy in Eq. (C.10) subject to the constraint that $\int \rho(\boldsymbol{r}) d^3\boldsymbol{r} = N$ for a system of N electrons, one can obtain the Kohn-Shan effective Shrödinger equation,

$$\left[-\frac{\nabla^2}{2m} + V_{\text{eff}}(\boldsymbol{r}) \right] \phi_n(\boldsymbol{r}) = \epsilon_n \phi_n(\boldsymbol{r}). \tag{C.16}$$

This effective potential is given by

$$V_{\text{eff}}(\boldsymbol{r}) = -\sum_a \frac{Z_a}{|\boldsymbol{r} - \boldsymbol{r'}|} + \int \frac{\rho(\boldsymbol{r'})}{|\boldsymbol{r} - \boldsymbol{r'}|} d^3\boldsymbol{r'} + \mu_{\text{xc}}[\rho(\boldsymbol{r})] + V_{\text{ext}}(\boldsymbol{r}) \tag{C.17}$$

where $\mu_{\text{xc}}[\rho(\boldsymbol{r})] = \delta E_{xc}[\rho]/\delta\rho$ is a functional derivative.

This is the basis of finding the electronic density, $\rho[\boldsymbol{r}]$. An extension of this to a system that also has positron density will yield V_{eff} for a positron, as shown by Sterne and Kaiser [49], but this calculation is beyond the scope of this thesis.

D Measuring κ with Magnetic Quenching

Although the internal contact density of Ps has not been measured in Ar, it has been experimentally determined in other materials through a technique known as magnetic quenching. For this reason, κ can also be referred to as the "quenching rate constant."

When Ps is placed in a uniform magnetic field of magnitude B, the perturbing Hamiltonian is given by

$$\hat{H}_1 = -\left(\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-\right) \cdot \boldsymbol{B}. \tag{D.1}$$

The $|\uparrow\rangle$ and $|\downarrow\rangle$ spin states for the electron and the positron are eigenvectors of these operators:

$$\boldsymbol{\mu}_+ |\uparrow\rangle_+ = \mu |\uparrow\rangle_+ , \tag{D.2}$$
$$\boldsymbol{\mu}_+ |\downarrow\rangle_+ = -\mu |\downarrow\rangle_+ , \tag{D.3}$$
$$\boldsymbol{\mu}_- |\uparrow\rangle_- = -\mu |\uparrow\rangle_- , \tag{D.4}$$
$$\boldsymbol{\mu}_- |\downarrow\rangle_- = \mu |\downarrow\rangle_- , \tag{D.5}$$

where μ is the scalar magnetic moment of the electron. Each state of Ps can be written as a direct product of these eigenstates:

$$|0,0\rangle = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle_+ |\downarrow\rangle_- - |\downarrow\rangle_+ |\uparrow\rangle_- \right), \tag{D.6}$$
$$|1,0\rangle = \frac{1}{\sqrt{2}} \left(|\uparrow\rangle_+ |\downarrow\rangle_- + |\downarrow\rangle_+ |\uparrow\rangle_- \right), \tag{D.7}$$
$$|1,1\rangle = |\uparrow\rangle_+ |\uparrow\rangle_- , \tag{D.8}$$
$$|1,-1\rangle = |\downarrow\rangle_+ |\downarrow\rangle_- . \tag{D.9}$$

The $|1,1\rangle$ and $|1,-1\rangle$ states are already eigenstates of $\hat{H}_1$, so we need only consider

the 2×2 array for the $|0,0\rangle$ p-Ps and $|1,0\rangle$ o-Ps states,

$$\begin{bmatrix} \left\langle 0,0 \middle| \hat{H}_1 \middle| 0,0 \right\rangle & \left\langle 0,0 \middle| \hat{H}_1 \middle| 1,0 \right\rangle \\ \left\langle 1,0 \middle| \hat{H}_1 \middle| 0,0 \right\rangle & \left\langle 1,0 \middle| \hat{H}_1 \middle| 1,0 \right\rangle \end{bmatrix} = \begin{bmatrix} 0 & -2\mu B \\ -2\mu B & 0 \end{bmatrix}. \tag{D.10}$$

If we write the hyperfine splitting between the o-Ps and p-Ps states in vacuum as $\hbar\omega_0$, then we need to add the factor $\Delta E_{\mathrm{o-p}} = \hbar\omega$, where $\hbar\omega = \kappa\hbar\omega_0$. This factor makes the matrix for $\hat{H}$ in the spin basis become

$$\hat{H} = \begin{bmatrix} 0 & -2\mu B \\ -2\mu B & \hbar\omega \end{bmatrix} = -2\mu B \begin{bmatrix} 0 & 1 \\ 1 & -\frac{2}{x} \end{bmatrix}, \tag{D.11}$$

where $x \equiv 4\mu B/\hbar\omega$. Diagonalizing the matrix in Eq. (D.11) gives the eigenvalues $-[1 \pm \sqrt{1+x^2}]/x$, resulting in the eigenstates

$$|+\rangle = \frac{1}{\sqrt{1+y^2}} |1,0\rangle - \frac{y}{\sqrt{1+y^2}} |0,0\rangle , \tag{D.12}$$

$$|-\rangle = \frac{y}{\sqrt{1+y^2}} |1,0\rangle + \frac{1}{\sqrt{1+y^2}} |0,0\rangle , \tag{D.13}$$

where $y = x/[1+\sqrt{1+x^2}]$.

Using Eq. (D.13), we can write the self-annihilation rate of the $|-\rangle$ state, Γ^-, in terms of Γ^t and Γ^s, the self-annihilation rates of o-Ps and p-Ps without a magnetic field:

$$\Gamma^- = \frac{y^2}{1+y^2}\Gamma^t + \frac{1}{1+y^2}\Gamma^s. \tag{D.14}$$

Using Eq. (2.24), which gives the modified annihilation rates for Ps in a solid in terms of κ, we can write:

$$\Gamma^- - \Gamma^s = \frac{y^2}{1+y^2}(\Gamma^t - \Gamma^s) = \frac{\kappa y^2}{1+y^2}(\Gamma^t_0 - \Gamma^s_0). \tag{D.15}$$

When $B = 0$, x and y are also both zero, so $\Gamma^s = \Gamma^-(B=0)$. Using the accepted

values of 0.007 ns^{-1} and 2 ns^{-1} for Γ_0^t and Γ_0^s, respectively, we obtain

$$\Gamma^-(B) - \Gamma^-(B=0) = -\frac{(1.993\text{ns}^{-1})\kappa y^2}{1+y^2}. \tag{D.16}$$

Since y is just a function of x, and $x = 4\mu B/\kappa\hbar\omega_0 = B/(\kappa{\times}36\text{ kGauss})$, the annihilation rate of the $|-\rangle$ state is simply a function of the magnetic field strength B (which can easily be controlled) and the internal contact density κ [50].

Γ^- can be measured by fitting the spectra for varying B to find the different annihilation rates and looking for one that shifts in the magnetic field. The self-annihilation rate Γ^- will not be hidden by pick-off effects because the $|-\rangle$ state is a correction to the p-Ps $|0,0\rangle$ state, whose self-annihilation rate dominates any pick-off annihilation.

By measuring annihilation rates as a function of magnetic field strength, a number of experimentalists have measured κ inside various solids. For example, Nagai, Nagashima, and Hyodo have determined that $\kappa = 0.31 \pm 0.02$ in α-SiO_2 [30]. This magnetic quenching method could be used to measure κ in Ar to compare with our result of $\kappa = 1.25 \pm 0.02$.

E PIMC Program Code

```
1   PROGRAM PIMC  ! Path Integral Monte Carlo
2
3   ! ***** The HISTORY of this program: *****
4   !
5   ! This code originated from sphere4.2.f90 (Summer 1999), which was a
6   ! staging program for a single particle using the image potential due
7   ! to a hard sphere. It evolved to sphere_esV5.f90 (March 2000), which
8   ! was modified to use the pseudopotential from the electronic structure
9   ! calculation and to read discrete potentials with periodic boundary
10  ! conditions. A cubic spline interpolation routine due to J.E. Pask was
11  ! added to interpolate between potential values.
12  !
13  ! periodic1.f90 was tailored for periodic potentials and used routines
14  ! collected by Philip Sterne (sterne1@llnl.gov) in a wrapper vcg.f90,
15  ! which calculated positron potentials by interpolation from the output
16  ! file of another program called fepot. This became a two-chain model
17  ! in periodic2.f90, in which the chains interacted via a Yukawa
18  ! potential.
19  !
20  ! In Mac_PEZ.f90, the chains could also interact via the Coulombic
21  ! thermal density propagator and the bead correlation function was
22  ! calculated. This code could also run on a Mac as well as Oxford.
23  ! Mac_PPEZnew.f90 is a parallel version of the code designed to run on
24  ! the Appleseed cluster. The pair correlation function, g(r), was also
25  ! added.
26
27  ! ***** Separate Modules *****
28
29  use vcGrid       ! deals with vcg files: potential V and charge density
30                   !   C on a grid G (Phil Sterne)
31                   !   300 lines
32  use types        ! sets double precision type (J. E. Pask, January 1997)
33                   !   13 lines
34  use epot         ! produces pseudopotential for electron in lattice
35                   !   (Amy Bug, April 2000)
36                   !   308 lines
37  use cluster_elec ! generates cluster of atoms that contribute to
38                   !   electron pseudopotential (Amy Bug, April 2000,
39                   !   adopted from subroutine clust by Phil Sterne)
40                   !   524 lines
41  use cluster_gofr ! generates cluster of atoms that contribute to
42                   !   g(r) function (Lisa Larrimore, July 2001, adopted
43                   !   from cluster_elec by Amy Bug)
44                   !   225 lines
45  use Table        ! calculates look-up table for the Ps thermal density
```

```
46                  !   matrix (Roy Pollock)
47                  !   662 lines
48
49  ! ***** Variable Declarations *****
50
51  implicit none
52
53  ! MPI variables used for parallelization
54  include 'mpif.h'
55  integer :: rank, size, tag, count, steps, ierr
56  integer :: status(MPI_STATUS_SIZE)
57  real (dp), allocatable :: R_Corr_Temp(:), radbin_temp(:)
58  real (dp), allocatable :: tempbin(:,:,:), grbeadbin_temp(:)
59  real (dp), allocatable :: grcmbin_temp(:)
60
61  integer, parameter :: nbinmax=10000 ! limiting number of radial bins
62  integer, parameter :: limlow=-20,limhi=20 ! limits of potl array points
63  integer, parameter :: nx=41,ny=41,nz=41 ! should agree w/ limits above
64  real, parameter :: delta = .502 ! spacing between grid points;
65                                  ! this*limlow = u.c. diam
66  real, parameter :: pi = 3.14159
67
68  integer :: nbins                ! number of bins for radial binning
69  real :: rbinmax                 ! limiting radius for binning
70
71  integer :: i,j,k,s,ib,id,ic,irun,ibin   ! counters
72  integer :: nequil       ! after nequil MC passes, begin data taking
73  integer :: nevalu = 10  ! update acceptance rate every nevalu MC passes
74  integer :: ninit        ! initialize from old position or not?
75  integer :: nb, mb, npass, jump  ! see read(11,*)'s for explanation
76  integer :: npts(3)              ! replaces nx, ny, nz
77  integer :: CorrelationCounts    ! number of corr calcs to do per cycle
78  integer :: deg                  ! degree of spline
79
80  logical :: UsePollock           ! use Pollock? (else, use Yukawa)
81  logical :: UseExternal          ! use external potentials? (else, free Ps)
82  logical, allocatable :: x_changed(:,:)  ! Flag for beads that are moved
83
84  real, dimension(:,:,:), allocatable :: x,xn  ! position of Ps beads
85  real, allocatable :: xEnergyOld(:,:)    ! potl E of each pair of beads
86  real, allocatable :: xEnergyNew(:,:)    ! potl E of each new pair
87  real (dp), allocatable :: xOverlap(:)   ! overlap coefficients
88  real (dp), allocatable :: R_Correlation(:)  ! correlation between beads
89  real (dp) :: R_Dist                  ! distance between a pair of beads
90  real :: DP_New, DP_Old = 0.0            ! density potentials
91  real :: xc1(3), xc2(3), xc(3)           ! centroids (e-, e+, Ps)
92  double precision :: xper, yper, zper
93  real :: elecbin(limlow:limhi, limlow:limhi, limlow:limhi)
```

```
94                                          ! cartesian bins for electron
95   real :: posibin(limlow:limhi, limlow:limhi, limlow:limhi)
96                                          ! cartesian bins for positron
97   real (dp) :: radbin(nbinmax)    ! radial bins for relative coordinate
98
99   ! Energy variables:
100  real (dp) :: energy_ave         ! kin energy
101  real (dp) :: energyV_ave        ! pot'l energy
102  real (dp) :: energy2_ave        ! ave kin energy squared
103  real (dp) :: energyV2_ave       ! ave pot'l energy squared
104  integer :: energy_count         ! counter for averaging
105
106  ! g(r) variables:
107  real (dp) :: gr_bead_bin(nbinmax) ! array for binning bead-atom g(r)
108  real (dp) :: gr_cm_bin(nbinmax)  ! array for binning cm-atom g(r)
109  real (dp) :: gr_pt(3)            ! point from which g(r) is measured
110  real (dp) :: gr_dist             ! distance over which g(r) is measured
111  real (dp) :: gr_rmax
112          ! gr_dist + rcell (radius of sphere enclosing unit cell)
113  real (dp), pointer :: gr_clus(:,:)
114          ! indices of atoms within gr_dist of unit cell boundary
115  integer :: gr_nclus              ! number of atoms in gr_nclus
116  integer :: gr_bead_count ! # of times bead-atom distances are binned
117  integer :: gr_cm_count   ! number of times cm-atom distances are binned
118  real (dp) :: tmat(3,3)   ! transformation from real to lattice vector
119  real (dp) :: temp1, temp2
120
121  real ::  amass, beta, hbar, rcav, aep ! see read(11,*)'s below
122  real ::  wave                         ! deB wavelength of free particle
123  real (dp) :: c_overlap  ! charge overlap integral with gamma correction
124  real (dp) :: c_overlap_zero     ! charge overlap integral alone
125  integer :: c_overlap_count    ! counts times c_overlap is calculated
126  real :: ac, acsum      ! acceptance rate and its average for bead moves
127  real :: accm, accmsum   ! acceptance rate and its average for cm moves
128  real :: sigelec, sigposi, sigelec_self, sigposi_self    ! width of path
129  real :: rrel                    ! relative coordinate distance
130  real :: rforbin(nbinmax)        ! relative coord distance as bin radius
131
132  integer, parameter :: fileNameLength=70 ! length of string for filename
133  integer   :: iui                    ! logical unit number for input
134  integer   :: iuo                    ! logical unit number for output
135
136  real (dp) :: rv_(3,3)        ! lattice vectors in atomic units
137                               ! rv_(i,j) is jth component of ith vector
138  real (dp) :: gv_(3,3)        ! inverse lattice vectors
139  real (dp) :: rv(3,3)         ! transpose of lattice vectors
140                               ! rv(i,j) is ith comp of jth vector
141  real (dp) :: gv(3,3)         ! transpose of inverse lattice vectors
```

```
142   real (dp) :: tolcs           ! tolerance for cluster size
143   logical :: lprint_e          ! verbose print electron cluster info?
144   real (dp) :: dummyvec(3,3)   ! dummy; same as lattice vectors
145   integer   :: dummynpts(3)    ! dummy; same as number of lattice pts
146   real (dp) :: rpoint (3)      ! point in cartesian coordinates
147   real (dp) :: pot             ! potential
148   real (dp) :: cdt             ! total charge density
149   type(vcgData) :: potVcg      ! potential on spline grid
150   type(vcgData) :: cdVcg       ! total charge density on spline grid
151   type(vcgData) :: gamVcg      ! contact corr function on  spline grid
152   logical :: periodic(3)       ! t = lattice vector direc is periodic
153   character(fileNameLength) :: vcgFile ! filename for vcg file
154   character(fileNameLength) :: eptFile ! filename for epot input
155   real (dp) :: gam                     ! enhancement factor
156   integer   :: error                   ! error flag
157
158   real :: starttime, endtime      ! time variables
159   character(fileNameLength) :: TempFort   ! for renaming Fortran files
160
161   ! ***** Executable *****
162
163   starttime = MPI_WTIME()   ! Using a function from the Message
164                             ! Passing Interface (MPI) library, we
165                             ! can time our calculation.
166   call ReadInput()          ! Get input parameters.
167   call Initialize()         ! Initialize variables.
168
169   ! Loop through the PIMC routine for a total of 'npass' times.
170   MC_passes:  do irun = 0, npass-1
171
172     ! Do a Monte Carlo move.
173     call move(ac)
174
175     ! Update the centers of mass.
176     do i=1,3
177       xc1(i) = sum(x(:,1,i))/float(nb)
178       xc2(i) = sum(x(:,2,i))/float(nb)
179     end do
180     xc = (xc1+xc2)/2.0
181
182     ! Print what percent of the calculation has been completed,
183     ! using the magic of integer arithmetic.
184     if (irun*100/npass > (irun-1)*100/npass) write(*,*) &
185       (irun*100/npass), "percent done"
186
187     ! Now we update the acceptance rate by adding 'ac' to 'acsum'.
188     ! ac was set to 1 by move(ac) if a successful move was made.
189     acsum = acsum + ac
```

```
190
191     ! Every nevalu-th move, we look at acsum, and see how many moves are
192     ! being accepted. If the rate is too low or two high, we adjust the
193     ! number of beads moved in each step.
194     if (mod(irun,nevalu) == nevalu-1) then
195       ac = acsum/(nevalu-1)    ! Determine fraction of accepted moves.
196                           ! (single bead moves only occur nevalu-1 times)
197       write(6,100) irun,ac,mb  ! a simple diagnostic
198       100 format ("",I8,"       ",F6.2,"          ",I4)
199       ! adjust mb so the acceptance rate is roughly 50%
200       if(ac >  0.5) mb=mb+1
201       if(ac < 0.5) mb=mb-1
202       if(mb < 1) mb=1
203       if(mb > nb) mb=nb
204       acsum = 0
205     end if
206
207     ! Every jump-th move, we make calculations regarding the centroids of
208     ! the beads, their dispersion, and their correlation function.
209
210     if (mod(irun,jump)==0) then    ! When irun is a multiple of jump...
211
212       ! if we have completed the specified number of equilibration
213       ! steps, then...
214       if (irun.gt.nequil) then
215
216         ! ...we bin the cartesian coordinates of the beads...
217         call makexyzbin(elecbin,posibin)
218
219         ! ...and the relative, radial, seperation of the electron and
220         ! positron beads...
221         CALL RadialBin()
222
223         ! ...and calculate the overlap integral...
224         CALL ChargeOverlap()
225         c_overlap_count = c_overlap_count + 1
226         write(30,*) c_overlap/float(c_overlap_count)/float(nb), &
227                     c_overlap_zero/float(c_overlap_count)/float(nb)
228
229         ! ...and calculate the pair correlation function.
230         CALL GofR_Bin()
231
232       end if
233
234       ! Sometimes, we find the bead that is farthest from the
235       ! center of mass, in the x-direction:
236       ! i=1
237       ! do j=2,nb
```

```
238       !   rrel=abs(x(i,2,1)-xc2(1))
239       !   if (abs(x(j,2,1)-xc2(1)).gt.rrel) i=j
240       ! enddo
241       ! if (i.eq.nb) rrel=abs(x(i,2,1)-xc2(1))
242       ! write(39,*) rrel
243
244       ! Write the position of the electron and positron centroids:
245       write(31,"(6f12.5)") xc1, xc2
246
247       ! Calculate electron and positron bead dispersions:
248       sigelec = 0.0
249       sigposi = 0.0
250       sigelec_self=0.0
251       sigposi_self=0.0
252       do ib = 1,nb
253         sigelec = sigelec + (x(ib,1,1)-xc(1))**2 + (x(ib,1,2)-xc(2))**2 &
254           + (x(ib,1,3) - xc(3))**2
255         sigposi = sigposi + (x(ib,2,1)-xc(1))**2 + (x(ib,2,2)-xc(2))**2 &
256           + (x(ib,2,3) - xc(3))**2
257         sigelec_self = sigelec_self + (x(ib,1,1)-xc1(1))**2 &
258           + (x(ib,1,2)-xc1(2))**2 + (x(ib,1,3) - xc1(3))**2
259         sigposi_self = sigposi_self + (x(ib,2,1)-xc2(1))**2 &
260           + (x(ib,2,2)-xc2(2))**2 + (x(ib,2,3) - xc2(3))**2
261       end do
262       sigelec = (sigelec/float(nb)) ** .5
263       sigposi = (sigposi/float(nb)) ** .5
264       sigelec_self = (sigelec_self/float(nb)) ** .5
265       sigposi_self = (sigposi_self/float(nb)) ** .5
266       ! Write these values out to the 'fort.12' file.
267       write(12,"(i6, 1x, 4f12.5)") irun, sigposi, sigelec, &
268         sigelec_self, sigposi_self
269
270       ! Calculate the bead correlation function:
271       ! CALL DoCorrelation()
272
273     end if
274
275   end do MC_passes
276
277   write(*,*) "C_Overlap:", c_overlap / float(c_overlap_count) / float(nb)
278   write(*,*) "C_Overlap_Zero:", c_overlap_zero / float(c_overlap_count) &
279                                 / float(nb)
280
281   ! Write the bead correlation function to fort.18
282   ! When running on the AppleSeed cluster, use these lines to
283   ! make all "slave" computers send their final results to the
284   ! "master" node.
285   allocate(R_Corr_Temp(nb) )
```

```
286   if(rank == 0) then
287     do i = 1,size-1
288       call MPI_RECV(R_Corr_Temp,nb,MPI_DOUBLE_PRECISION,i,5, &
289                     MPI_COMM_WORLD,status,ierr)
290       R_Correlation(:) = R_Correlation(:) + R_Corr_Temp(:)
291     enddo
292   else
293     call MPI_SEND(R_Correlation,nb,MPI_DOUBLE_PRECISION,0,5, &
294                   MPI_COMM_WORLD,ierr)
295   endif
296   deallocate(R_Corr_Temp)
297
298   TempFort='for18_correlation'
299   open(unit=18,file=TempFort,status='replace',action='write')
300   write(18,*) "Bead Correlation Function of Positronium: The square of"
301   write(18,*) "the ave dist between a bead and its Nth neighbor."
302   write(18,*) "Data taken from PPEZnew. Some constants:"
303   write(18,*) "Beta = ", beta
304   write(18,*) "# of beads =", nb
305   write(18,*) "# of runs  =", npass
306   do i = 1,nb
307     write(18,*) i, R_Correlation(i)!/R_Correlation(1)
308   enddo
309
310   ! Write the radial distribution function to fort.20
311   allocate(radbin_temp(nbinmax) )
312   if(rank == 0) then
313     do i = 1,size-1
314       call MPI_RECV(radbin_temp,nbinmax,MPI_DOUBLE_PRECISION,i,6, &
315                     MPI_COMM_WORLD,status,ierr)
316       radbin(:) = radbin(:) + radbin_temp(:)
317     enddo
318   else
319     call MPI_SEND(radbin,nbinmax,MPI_DOUBLE_PRECISION,0,6, &
320                   MPI_COMM_WORLD,ierr)
321   endif
322   deallocate(radbin_temp)
323
324   TempFort='for20_rad_dist'
325   open(unit=20,file=TempFort,status='replace',action='write')
326   write(20,*) "Dist'n fcn of rel coord, e+ dist from (0,0,0)"
327   write(20,*) "Beta = ", beta
328   write(20,*) "# of beads =", nb
329   write(20,*) "# of runs  =", npass
330   do ibin = 1, nbins
331     write(20,*) rforbin(ibin), radbin(ibin)
332   end do
333
```

```
334   ! Write the electron distribution to fort.21
335   allocate(tempbin(limlow:limhi, limlow:limhi, limlow:limhi) )
336   if(rank == 0) then
337     do i = 1,size-1
338       call MPI_RECV(tempbin,(-limlow+limhi+1)**3,MPI_DOUBLE_PRECISION, &
339                     i,7,MPI_COMM_WORLD,status,ierr)
340       elecbin(:,:,:) = elecbin(:,:,:) + tempbin(:,:,:)
341     enddo
342   else
343     call MPI_SEND(elecbin,(-limlow+limhi+1)**3,MPI_DOUBLE_PRECISION, &
344                   0,7,MPI_COMM_WORLD,ierr)
345   endif
346   deallocate(tempbin)
347
348   TempFort='for21_cart_elec'
349   open(unit=21,file=TempFort,status='replace',action='write')
350   write(21,*) "Cartesian Electron Distribution"
351   write(21,*) "Some constants:"
352   write(21,*) "Beta = ", beta
353   write(21,*) "# of beads =", nb
354   write(21,*) "# of runs  =", npass
355   do i = limlow, limhi
356     do j = limlow, limhi
357       do k = limlow, limhi
358        write(21,*) elecbin(i,j,k)
359       end do
360     end do
361   end do
362
363   ! Write the positron distribution to fort.22
364   allocate(tempbin(limlow:limhi, limlow:limhi, limlow:limhi) )
365   if(rank == 0) then
366     do i = 1,size-1
367       call MPI_RECV(tempbin,(-limlow+limhi+1)**3,MPI_DOUBLE_PRECISION, &
368                     i,8,MPI_COMM_WORLD,status,ierr)
369       posibin(:,:,:) = posibin(:,:,:) + tempbin(:,:,:)
370     enddo
371   else
372     call MPI_SEND(posibin,(-limlow+limhi+1)**3,MPI_DOUBLE_PRECISION, &
373                   0,8,MPI_COMM_WORLD,ierr)
374   endif
375   deallocate(tempbin)
376
377   TempFort='for22_cart_posi'
378   open(unit=22,file=TempFort,status='replace',action='write')
379   write(22,*) "Cartesian Positron Distribution"
380   write(22,*) "Some constants:"
381   write(22,*) "Beta = ", beta
```

```
382   write(22,*) "# of beads =", nb
383   write(22,*) "# of runs  =", npass
384   do i = limlow, limhi
385     do j = limlow, limhi
386       do k = limlow, limhi
387        write(22,*) posibin(i,j,k)
388       end do
389     end do
390   end do
391
392   ! Write the g(r) raw data
393   ! N(r) = gr_bin/gr_count
394   ! g(r) = N(r)/(atom density * volume of bin at r)
395   !      atom density = 4/a^3 for fcc crystal lattice
396   !      volume of bin at r = 4 * pi * r^2 * bin width
397   allocate(grcmbin_temp(nbinmax) )
398   if (rank.eq.0) then
399     do i = 1,size-1
400       call MPI_RECV(grcmbin_temp,nbinmax,MPI_DOUBLE_PRECISION,i,9, &
401                     MPI_COMM_WORLD,status,ierr)
402       gr_cm_bin(:) = gr_cm_bin(:) + grcmbin_temp(:)
403     enddo
404   else
405     call MPI_SEND(gr_cm_bin,nbinmax,MPI_DOUBLE_PRECISION,0,9, &
406                   MPI_COMM_WORLD,ierr)
407   endif
408   deallocate(grcmbin_temp)
409
410   TempFort='for23_cm_gofr'
411   open(unit=23,file=TempFort,status='replace',action='write')
412   write(23,*) "cm-atom pair correlation function g(r)"
413   write(23,*) "beta, nb, npass, nequil:", beta, nb, npass, nequil
414   write(23,*) "gr_dist, bin width (gr_dist/nbins):", gr_dist, &
415     gr_dist/float(nbins)
416   write(23,*) "distances binned how many times? on how many computers?",&
417     gr_cm_count, size
418   do i=1,nbins
419     temp1 = gr_dist/float(nbins)*(i-1)            !r
420     temp2 = gr_cm_bin(i)/float(gr_cm_count*size)  ! N(r)
421     write(23,*) temp1, temp2
422   enddo
423
424   allocate(grbeadbin_temp(nbinmax) )
425   if (rank.eq.0) then
426     do i=1,size-1
427       call MPI_RECV(grbeadbin_temp,nbinmax,MPI_DOUBLE_PRECISION,i,2, &
428                     MPI_COMM_WORLD,status,ierr)
429       gr_bead_bin(:) = gr_bead_bin(:) + grbeadbin_temp(:)
```

```
430     enddo
431   else
432     call MPI_SEND(gr_bead_bin,nbinmax,MPI_DOUBLE_PRECISION,0,2, &
433                   MPI_COMM_WORLD,ierr)
434   endif
435   deallocate(grbeadbin_temp)
436
437   TempFort='for24_bead_gofr'
438   open(unit=24,file=TempFort,status='replace',action='write')
439   write(24,*) "bead-atom pair correlation function g(r)"
440   write(24,*) "beta, nb, npass, nequil:", beta, nb, npass, nequil
441   write(24,*) "gr_dist, bin width (gr_dist/nbins):", gr_dist, &
442     gr_dist/float(nbins)
443   write(24,*) "distances binned how many times? on how many computers?",&
444     gr_bead_count, size
445   do i=1,nbins
446     temp1 = gr_dist/float(nbins)*(i-1)                    ! r
447     temp2 = gr_bead_bin(i)/float(gr_bead_count*size)      ! N(r)
448     write(24,*) temp1, temp2
449   enddo
450
451   ! Write the final positions of the beads to fort.17
452   TempFort='for17_final_bead'
453   open(unit=17,file=TempFort,status='replace',action='write')
454   do ic = 1, 2
455     do ib = 1, nb
456       write(17,*) x(ib,ic,:)
457     end do
458   end do
459   print*, "these data came from Mac_PPEZnew.f90"
460
461   endtime = MPI_WTIME()
462   print*, "Total time (in seconds):", endtime-starttime
463   print*, "           (in hours):", (endtime-starttime)/3600.0
464
465   call MPI_FINALIZE(ierr)
466
467   contains
468
469   !__________INIT_BEADS SUBROUTINE  _________________!
470
471   subroutine init_beads(ni)
472   implicit none
473   integer :: is = 1     ! (rightnow, a dummy) variable for gaussian RNG
474   integer :: id         ! counter for dimension
475   integer :: ierror
476   integer :: ibeadcount ! number of beads in reading file
477   integer :: ni         ! type of initialization: 0 de novo or 1 from file
```

```
478   integer :: ic         ! charge (electron = 1 or positron = 2)
479   real :: xsum,xshift
480   double precision :: gg
481
482   ! If starting from scratch (ni flag is 0 in PEZ.in), place the
483   ! beads in a Gaussian distribution about the origin.
484   if(ni == 0) then
485   gg = min(wave*wave,rcav*rcav/12.0)
486   ! start gg smaller if you wish
487   !   gg = 0.1
488     dim: do id = 1,3
489       xsum = 0.0
490       charge1: do ic = 1,2
491         bead1: do ib = 1, nb
492           x(ib,ic, id) = gauss(gg,is)
493           xsum=xsum+x(ib,ic,id)
494         end do bead1
495         x(nb+1,ic,id) = x(1,ic,id)
496       end do charge1
497       xsum = xsum/2.0/float(nb)
498       xshift = xc(id)-xsum
499       charge2: do ic = 1,2
500         bead2: do ib = 1,nb+1
501           x(ib,ic,id)=x(ib,ic,id)+xshift
502         end do bead2
503       end do charge2
504     end do dim
505   end if
506
507   ! If the ni flag is set to 1, read the initial positions from the
508   ! file for17_final_bead.
509   if (ni == 1) then
510     ibeadcount = 0
511     open (unit = 16, file = 'for17_final_bead', status = 'old', &
512          action = 'read', iostat = ierror)
513     if (ierror /= 0) then
514       write(*,*) 'An error occured opening for17_final_bead'
515       write(*,*) 'Consider changing ninit in the PEZ.in file to 0.'
516       STOP
517     end if
518     charge3: do ic = 1, 2
519       bead3: do ib = 1, nb
520         read(16,*,iostat = ierror) x(ib,ic,1), x(ib,ic,2), x(ib,ic,3)
521         ibeadcount = ibeadcount+1
522         if (ierror /= 0) STOP
523       end do bead3
524     end do charge3
525
```

```
526     !Close off the chain
527     x(nb+1,:,:) = x(1,:,:)
528
529     if(ibeadcount /= 2*nb) then
530       write(*,*) 'too few or too many beads in for17_final_bead file'
531       STOP
532     end if
533   end if
534
535   if (ni > 1) then
536     write(*,*) 'erroneous flag for reading beads'
537     STOP
538   end if
539
540   ! Initially, the "new" positions in xn are the same as the old.
541   xn = x
542
543   end subroutine init_beads
544
545   !_______MC MOVE SUBROUTINE_______!
546
547   subroutine move(ac)
548   use vcGrid
549   use types
550   implicit none
551   !real, DIMENSION(:,:,:), POINTER :: xtemp
552   real, intent(out) :: ac
553   real :: vsum, vsumnew, vchange, de, det, gsum, gsumnew,gchange,gtest
554   real :: vee, energy, yuk_e_poo
555   real :: effBeta        ! beta / number of beads
556   real (dp) :: pot1      ! electron pseudopotl
557   real (dp) :: relec(3)  ! electron coordinates
558   real :: dis1(3), dis2(3), dis12(3)
559   real :: rdis1, rdis2, rdis12 !sep's for electron-positron interaction
560   real :: qep(2) ! charges on electron and positron
561   qep(1) = -1.0; qep(2) = 1.0
562
563   effBeta = beta / float(nb)
564   ac = 0.00 !set acceptance rate for this step equal to zero
565
566   ! Do a staging move on the beads (re-pick from a gaussian distribution).
567   ! Do electron and positron moves serially. Every 10 passes, try a
568   ! center of mass move instead.
569   if (mod(irun,10).eq.0) then
570     call move_cm(xn)
571   else
572     call tryboth(xn)
573   end if
```

```
574
575   ! DP_* are variables for the thermal density propagator 'potential'.
576   DP_Old = 0.0
577   DP_New = 0.0
578
579   do i = 1,nb
580     if ( x_changed(i,1) .OR. x_changed(i,2) ) then
581       ! If a bead has been moved, recalculate its energies...
582       dis1 = x(i,1,:) - x(i,2,:)
583       dis2 = x(i+1,1,:) - x(i+1,2,:)
584       dis12 = dis1 - dis2
585       rdis1  = sqrt(sum(dis1**2))
586       rdis2  = sqrt(sum(dis2**2))
587       rdis12 = sqrt(sum(dis12**2))
588
589       ! Calculate the energy associated with the positron-electron
590       ! interaction using the Pollock propagator or the Yukawa potl.
591       if (UsePollock) then
592         DP_Old = DP_Old - LookUpTable(rdis1, rdis2, rdis12, effBeta)
593       else
594         DP_Old = DP_Old + vfun1(rdis1,qep,aep) * effBeta
595       endif
596
597       xper = xn(i,2,1)
598       yper = xn(i,2,2)
599       zper = xn(i,2,3)
600       relec = (/xn(i,1,1), xn(i,1,2), xn(i,1,3)/)
601       dis1 = xn(i,1,:) - xn(i,2,:)
602       dis2 = xn(i+1,1,:) - xn(i+1,2,:)
603       dis12 = dis1 - dis2
604       rdis1  = sqrt(sum(dis1**2))
605       rdis2  = sqrt(sum(dis2**2))
606       rdis12 = sqrt(sum(dis12**2))
607
608       if (UsePollock) then
609         DP_New = DP_New - LookUpTable(rdis1, rdis2, rdis12, effBeta)
610       else
611         DP_New = DP_New + vfun1(rdis1, qep, aep) * effBeta
612       endif
613
614       ! The energy of each positron is calculated using the evalVcg
615       ! function in the vcGrid module, which uses a spline
616       ! interpolation of a vcg file.
617       if (x_changed(i,2) .and. UseExternal) then
618         xEnergyNew(i,2) = evalVcg(xper, yper, zper, potVcg)
619       endif
620
621       ! The energy of each electron is calculated using the eval_pseudo
```

```
622       ! function in the epot module.
623       if (x_changed(i,1) .and. UseExternal) then
624         call eval_pseudo(rv, gv_, relec, pot1)
625         xEnergyNew(i,1) = pot1 !electron-solid energy
626       endif
627
628     endif
629   end do
630
631   ! Now we can sum up old and new potentials
632   vsum = 0.0
633   vsumnew = 0.0
634   if (UseExternal) then
635     do i = 1,nb
636       vsum = vsum + sum(xEnergyOld(i,:))
637       vsumnew = vsumnew + sum(xEnergyNew(i,:))
638     enddo
639   end if
640
641   ! The energy difference between the old and new state, times an
642   ! effective beta (deltaE*beta), is stored as "vchange"
643   vchange = (vsumnew - vsum)*effBeta + DP_New - DP_Old
644   det = -vchange
645   de = dlog(ran1(rank) + 1.0d-10)  ! de
646   ac = 0.0d0
647   accm = 0.0d0
648   ! We accept the move only if exp(-beta*deltaE) > eta, or
649   ! -beta*deltaE > log(eta). In these variables, this means
650   ! that we accept if "det" > "de".
651   if (det > de ) then
652     gsum = 1.0  !dispense with cavity
653     if (mod(irun,10).eq.0) then
654       accm = 1.0d0
655       write(*,*) "CM MOVE ACCEPTED!"
656     else
657       ac=1.0d0    ! Flag the fact that a move has been accepted
658     end if
659     vee = vsumnew
660     do i = 1,nb
661       do j = 1,2
662         if( x_changed(i,j) ) then
663           x(i,j,:) = xn(i,j,:)
664           if(i==1) x(nb+1,j,:) = xn(nb+1,j,:)
665           xEnergyOld(i,j) = xEnergyNew(i,j)
666         endif
667       enddo
668     enddo
669   else
```

```
670     vee = vsum
671     do i = 1,nb
672       do j = 1,2
673         if( x_changed(i,j) ) then
674           xn(i,j,:) = x(i,j,:)
675           if(i==1) xn(nb+1,j,:) = x(nb+1,j,:)
676           xEnergyNew(i,j) = xEnergyOld(i,j)
677         endif
678       enddo
679     enddo
680   end if
681
682   yuk_e_poo = 0.0
683   ! Add up the Coulombic energy in the beads, using Yukawa potential.
684   ! We currently use the Yukawa potential to actually calculate the
685   ! energy, even when we were using the Pollock propagator to
686   ! determine which moves to accept. This will be modified in future
687   ! investigations.
688   do i = 1,nb
689     rdis1 = sqrt(sum( (xn(i,1,:)-xn(i,2,:))**2) )
690     yuk_e_poo = yuk_e_poo + vfun1(rdis1,qep,aep)
691   enddo
692   vee = (vee + yuk_e_poo) / float(nb)
693
694   ! Do some energy calculations...
695
696   if (irun.gt.nequil) then
697
698     energy = 1.5/beta*nb*2 - vquant()*nb*amass/2/beta**2 + vee
699
700     ! There used to be a subroutine named "virial" that would
701     ! calculate the kinetic energy of the beads using the
702     ! virial estimator. This needs to be reimplemented with the
703     ! Pollock propagator.
704     !vir = virial()
705     !energyV = vir
706
707     energy_count = energy_count + 1
708     energy_ave = energy_ave + energy
709     energyV_ave = energyV_ave + vee
710     energy2_ave = energy2_ave + energy**2
711     !energyV2_ave = energyV2_ave + energyV**2
712
713     if (mod(irun,jump)==0) then
714       ! no virials now
715       write(13,*) energy2_ave/energy_count
716       write(14,*) energy, vee
717       write(15,*) energy_ave/energy_count, energyV_ave/energy_count
```

```
718
719       ! alternate versions of the above with virials
720       !write(13,*) energy2_ave/energy_count, energyV2_ave/energy_count
721       !write(14,*) energy, energyV
722       !write(15,*) energy_ave/energy_count, energyV_ave/energy_count
723     end if
724   end if
725
726   !uncomment to analyze with acf.f90:
727   !if ((90000<=irun) .and. (irun<100000)) write(30,*) energy, energyV
728   !if ((190000<=irun) .and. (irun<200000)) write(31,*) energy, energyV
729
730   end subroutine move
731
732   !______ VFUN1 FUNCTION _______!
733
734   real function vfun1(r,q,a)
735   implicit none
736   real :: r,a
737   real :: q(2)
738
739   ! This gives the potential energy due to the electron-positron
740   ! interaction, using the Yukawa potential (an approximation to
741   ! the Coulomb potential).
742
743   vfun1 = q(1) * q(2) * (1-exp(-r/a))/(r)
744   end function vfun1
745
746
747   !_______VQUANT FUNCTION_______!
748
749   real function vquant()
750   implicit none
751   real :: dx,dy,dz,dr2
752
753   ! Quantum Potential Function
754   ! vquant simply returns the sum of the distances between beads
755   ! as you count around the chain, which is used in calculating
756   ! the energy.
757
758   vquant = 0.0
759   do ic = 1, 2
760     do ib = 1, nb
761       dx = x(ib,ic,1) - x(ib+1,ic,1)
762       dy = x(ib,ic,2) - x(ib+1,ic,2)
763       dz = x(ib,ic,3) - x(ib+1,ic,3)
764       dr2 = dx**2 + dy**2 + dz**2
765       vquant = vquant + dr2
```

```
766     end do
767   end do
768   return
769   end function vquant
770
771   !_______Trial staging move SUBROUTINE_______!
772
773   subroutine tryboth(xnew)
774   implicit none
775   integer :: is = 1 !(rightnow, a dummy) variable for gaussian RNG
776   real, intent(INOUT), DIMENSION(:,:,:) :: xnew
777   double precision :: const, g
778
779   const=2.0d0*wave*wave/dfloat(nb)
780
781   x_changed(:,:) = .FALSE.
782
783   ! We pick the new bead positions according to a Gaussian
784   ! Distribution:
785   charge: do ic=1,2  ! loop over the electron (1) and positron (2)
786     dim: do id=1,3   ! "id" is the axis direction
787
788       ! We go from the j bead to the j+mb bead
789       ! (j is selected at random)
790       j=int(nb*ran1(rank))+1
791
792       beads:  do i=1,mb
793
794         ib=j+mb-i+1
795
796         ! Account for periodicity in the chain:
797         if (ib .GT. nb) ib = ib-nb
798
799         ! The gaussian width depends on which bead we are at,
800         ! using an interpolation formula due to Levy.
801         g=const*dfloat(mb-i+1)/dfloat(mb-i+2)
802         xnew(ib,ic,id) = (xnew(ib+1,ic,id)*(mb-i+1)+xnew(j,ic,id)) &
803           /float(mb-i+2) + gauss(g,is)
804
805         ! Flag the fact that this bead has been moved:
806         x_changed(ib,ic) = .TRUE.
807
808         ! Close the chain if we have moved the 1st bead:
809         if(ib == 1) then
810           xnew(nb+1,ic,id) = xnew(1,ic,id)
811         endif
812
813       end do beads
```

```
814     end do dim
815   end do charge
816
817   end subroutine tryboth
818
819   !______CM MOVES SUBROUTINE____________!
820
821   subroutine move_cm(xnew)
822   implicit none
823   real (dp) :: d(3)
824   real, intent(INOUT), DIMENSION(:,:,:) :: xnew
825
826   ! Every 10 passes, we attempt a center of mass move, where the
827   ! entire chain is relocated by up to 0.1 a.u.
828
829   d(1) = 0.2*ran1(rank) - 0.1
830   d(2) = 0.2*ran1(rank) - 0.1
831   d(3) = 0.2*ran1(rank) - 0.1
832   do ic = 1,2
833     do ib = 1,nb+1
834       xnew(ib,ic,:) = xnew(ib,ic,:) + d
835     end do
836   end do
837   x_changed(:,:) = .true.
838
839   end subroutine move_cm
840
841
842   ! _______BINIT SUBROUTINE _______ !
843
844   subroutine binit(nbns, rbmax, rforbin)
845   implicit none
846   integer :: nbns
847   integer :: ibn
848   real :: rbmax
849   real, intent(out) :: rforbin(nbns) ! radius corresponding to bin
850
851   ! The array "rforbin" contains the r value (in a.u.) corresponding
852   ! to each bin (for radial binning)
853
854   do ibin = 1, nbns
855     rforbin(ibin) = (ibin - .5)*rbmax / float(nbns)
856   end do
857   end subroutine binit
858
859   ! _______RADIAL BINNING SUBROUTINE ___ !
860
861   subroutine RadialBin()
```

```
862
863   ! The separation of each electron-positron pair is calculated
864   ! and binned in the array "radbin"
865
866   do ib = 1, nb
867     rrel = (x(ib,1,1) - x(ib,2,1))**2 + &
868       (x(ib,1,2) - x(ib,2,2))**2 +  (x(ib,1,3) - x(ib,2,3))**2
869     rrel = rrel ** .5
870     ibin = int(rrel/rbinmax*float(nbins))
871     radbin(ibin+1) = radbin(ibin+1)+1
872   end do
873   end subroutine RadialBin
874
875   ! _______G(r) BINNING SUBROUTINE ___ !
876
877   subroutine GofR_Bin()
878     real (dp) :: rrel
879     integer   :: ibin
880
881     ! The pair correlation function g(r) is calculated.
882     ! First, we bin the distances between each positron bead
883     ! and each solid atom:
884
885     do ib = 1, nb   ! Measure g(r) from each positron bead:
886       gr_pt = x(ib,2,:)
887       gr_pt = matmul(tmat,gr_pt)  ! Map bead position into
888                                   ! lattice vector units.
889       gr_pt = gr_pt - int(minval(gr_pt)) + 1  ! Add enough integers to
890                                               ! make them all +ve.
891       gr_pt = gr_pt - int(gr_pt)  ! Put it back into 1st zone.
892       gr_pt = matmul(rv,gr_pt)    ! Map back into cartesian coordinates.
893       do i = 1, gr_nclus
894         rrel = norm(gr_clus(:,i)-gr_pt)
895         ibin = int(rrel/gr_dist*float(nbins))
896         gr_bead_bin(ibin+1) = gr_bead_bin(ibin+1)+1 ! +1 prevents prob's
897                                                     ! when ibin=0
898       end do
899       gr_bead_count = gr_bead_count + 1
900     end do
901
902     ! Next, we bin the distances from the positron center of mass
903     ! to each solid atom:
904
905     gr_pt = xc2   ! Measure g(r) from positron cm:
906     gr_pt = matmul(tmat,gr_pt)    ! Map bead position into lattice
907                                   ! vector units.
908     gr_pt = gr_pt - int(minval(gr_pt)) + 1  ! Add enough integers to
909                                             ! make them all +ve.
```

```
910     gr_pt = gr_pt - int(gr_pt)    ! Put it back into 1st zone.
911     gr_pt = matmul(rv,gr_pt)      ! Map back into cartesian coordinates.
912     ! gr_pt = (/2.92, 0.0, 0.0/)  ! (Diagnostic for calculating g(r)
913                                   ! from a fixed point.)
914     ibin = int(gr_pt(1)/15.0*float(nbins))
915     do i = 1, gr_nclus
916       rrel = norm(gr_clus(:,i)-gr_pt)
917       ibin = int(rrel/gr_dist*float(nbins))
918       gr_cm_bin(ibin+1) = gr_cm_bin(ibin+1)+1
919     end do
920     gr_cm_count = gr_cm_count + 1
921
922   end subroutine GofR_Bin
923
924
925   ! _______CARGE OVERLAP SUBROUTINE ___ !
926
927   subroutine ChargeOverlap()
928   implicit none
929
930   integer   :: ib
931   real (dp) :: cdt
932
933   ! c_overlap and c_overlap_zero are used in different models for
934   ! finding the positron lifetime.
935   ! * c_overlap is calculated based on one of 5 or 6 models that
936   !   we adjust (e.g. Boronski-Nieminnen model, which is appropriate
937   !   for electrons in metals). Currently set for semiconductor.
938   ! * c_overlap_zero is Gamma_IPM, the annihilation rate in the
939   !   independent particle model. (This ignores exchange-correlation
940   !   effects.)
941
942   do ib = 1, nb
943     if( x_changed(ib,2) .OR. (irun == nequil+1)) then
944       ! positron is replaced in periodic unit cell in evalVcg routine
945       xper = x(ib,2,1)
946       yper = x(ib,2,2)
947       zper = x(ib,2,3)
948       cdt = evalVcg(xper, yper, zper, cdVcg)
949       if (cdt < 0) then
950         write(*,*) 'cdt is negative : ', cdt
951         ! spline causes (small) unphys osc in cdt; enforce >0
952         if (cdt < 0.0) cdt = 0.0
953       end if
954       xOverlap(ib) = cdt
955     else
956       cdt = xOverlap(ib)
957     endif
```

```
958
959     gam = gammaV(xper,yper,zper,gamVcg,cdt)
960
961     c_overlap = c_overlap + cdt * gam
962     c_overlap_zero = c_overlap_zero + cdt
963   end do
964
965   end subroutine ChargeOverlap
966
967   ! _______CORRELATION SUBROUTINE____ !
968
969   subroutine DoCorrelation()
970     implicit none
971     integer :: i,j,k,s
972     real (dp) :: R_Dist_Sqr
973
974     do k = 1, CorrelationCounts
975       i = int(nb*ran1(rank))+1
976       do s = 1, nb
977         if (i==s) CYCLE  ! Don't bother calculating the distance
978                          ! between a bead and itself.
979         ! Calculate the square of the distance between the i-th and
980         ! s-th bead:
981         R_Dist_Sqr = sum((x(i,1,:)+x(i,2,:)-x(s,1,:)-x(s,2,:))**2) / 4.0
982         j = s-i+1
983         if(j<1) j = j + nb
984         R_Correlation(j) = R_Correlation(j) + R_Dist_Sqr
985       enddo
986       ! R_Correlation(1) will always be zero, so we can use it to count
987       ! the number of iterations:
988       R_Correlation(1) = R_Correlation(1) + 1.0
989     enddo
990   end subroutine DoCorrelation
991
992   ! _______XYZBIN SUBROUTINE_________ !
993
994   subroutine makexyzbin(elecb,posib)
995   implicit none
996   integer :: il, jl, kl
997   integer :: ic
998   real :: side,halfside
999   real :: xper, yper, zper  ! bead within periodic box
1000  real :: elecb(limlow:limhi, limlow:limhi, limlow:limhi)
1001  real :: posib(limlow:limhi, limlow:limhi, limlow:limhi)
1002  real :: epsil = 0.01  ! extra bit for comparison of periodic locations
1003
1004  ! This assumes that half side of box is |limlow|*delta = limhi*delta
1005  side = (2*limhi + 1)*delta
```

```
1006   halfside = side/2.0
1007   charge: do ic = 1, 2
1008   bead:   do ib = 1, nb
1009    xper = x(ib,ic,1) - anint(x(ib,ic,1)/side) * side
1010    yper = x(ib,ic,2) - anint(x(ib,ic,2)/side) * side
1011    zper = x(ib,ic,3) - anint(x(ib,ic,3)/side) * side
1012    if(xper*xper > halfside*halfside + epsil) then
1013      write(*,*) 'over'
1014      write(*,*) 'ib, x ', ib, x(ib,ic,1),xper
1015      STOP
1016    end if
1017    if(yper*yper > halfside*halfside + epsil) then
1018      write(6,*) 'over'
1019      write(*,*) 'ib, y ', ib, x(ib,ic,2), yper
1020      STOP
1021    end if
1022    if(zper*zper > halfside*halfside + epsil) then
1023      write(6,*) 'over'
1024      write(*,*) 'ib, z ', ib, x(ib,ic,3), zper
1025      STOP
1026    end if
1027
1028    il = anint(xper/delta)
1029    jl = anint(yper/delta)
1030    kl = anint(zper/delta)
1031
1032    if(ic.eq.1) elecb(il,jl,kl) = elecb(il,jl,kl)+1.0
1033    if(ic.eq.2) posib(il,jl,kl) = posib(il,jl,kl)+1.0
1034    end do bead
1035    end do charge
1036
1037   end subroutine makexyzbin
1038
1039   !_______GAUSSIAN FUNCTION_______!
1040
1041   double precision function gauss(g,ix)
1042   implicit double precision (a-h, o-z)
1043   double precision :: rr, ss
1044   integer :: ix
1045
1046   rr = (-dlog(ran1(rank)+1.0d-10)*g ) ** 0.5
1047   ss = 6.283185307d0*ran1(rank)
1048   gauss = rr*dcos(ss)
1049
1050   end function gauss
1051
1052   !______READ INPUT FILE___________!
1053
```

```
1054  subroutine ReadInput()
1055    implicit none
1056    integer :: i,j
1057
1058    open(unit=11,file='PEZ.in',status='old',action='read')
1059    read(11,*);read(11,*) nb      ! # beads for the particle
1060    read(11,*);read(11,*) mb      ! #beads moved per staging pass
1061    read(11,*);read(11,*) npass   ! # staging passes
1062    read(11,*);read(11,*) amass   ! mass of a single quantum particle
1063    read(11,*);read(11,*) beta    ! beta = 1/kT (in au where hbar=1)
1064    read(11,*);read(11,*) hbar    ! making hbar smaller reduces quantum
1065                                  ! effects; hbar=1 in au)
1066    read(11,*);read(11,*) jump    ! number of passes between printing
1067    read(11,*);read(11,*) rcav    ! radius of spherical cavity, now just
1068                                  ! a length scale
1069    read(11,*);read(11,*) aep     ! Yukawa radius
1070    read(11,*);read(11,*) ninit   ! flag for initializing beads from file
1071    read(11,*);read(11,*) nequil  ! equilibration steps
1072    read(11,*);read(11,*) CorrelationCounts ! # of corr calculations
1073    read(11,*);read(11,*) deg
1074    read(11,*);read(11,*) vcgFile
1075    read(11,*);read(11,*) eptFile
1076
1077    ! Read Parameters
1078    read(11,*);read(11,*) i,j
1079    if( i == 1 ) then
1080          UsePollock = .TRUE.
1081    else
1082          UsePollock = .FALSE.
1083    end if
1084    if( j == 1 ) then
1085          UseExternal = .TRUE.
1086    else
1087          UseExternal = .FALSE.
1088    end if
1089
1090    close(11)
1091
1092    if(npass <= nequil) then
1093      write(*,*) 'npass is not greater than nequil :('
1094      STOP
1095    end if
1096
1097    ! The free particle deB wavelength is a useful bit of trivia:
1098    wave = (beta*hbar*hbar/amass)**0.5
1099    print*, 'Free particle deB wavelength:', wave
1100
1101  end subroutine ReadInput
```

```
1102
1103  !______INITIALIZE___________!
1104
1105  subroutine Initialize()
1106    implicit none
1107
1108    call MPI_INIT(ierr)
1109    write(*,*) "Initialized MPI, error =", ierr
1110    call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
1111    call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
1112    write(*,*) 'Process ', rank, ' of', size, ' is alive!'
1113
1114    ! We need to initialize everything:
1115    ! Starting with allocating memory for beads arrays and energies
1116
1117    allocate(x(nb+1,2,3),  stat = error)
1118    if (error .ne. 0) then
1119      write(*,*) "Unable to allocate memory for the array: x(:,:,:)"
1120      stop
1121    endif
1122    allocate(xn(nb+1,2,3), stat = error)
1123    if (error .ne. 0) then
1124      write(*,*) "Unable to allocate memory for the array: xn(:,:,:)"
1125      stop
1126    endif
1127    allocate(xEnergyOld(nb+1,2), stat = error)
1128    if (error .ne. 0) then
1129      write(*,*) "Unable to allocate memory for array: xEnergyOld(:,:)"
1130      stop
1131    endif
1132    allocate(xEnergyNew(nb+1,2), stat = error)
1133    if (error .ne. 0) then
1134      write(*,*) "Unable to allocate memory for array: xEnergyNew(:,:)"
1135      stop
1136    endif
1137    allocate(xOverlap(nb), stat = error)
1138    if (error .ne. 0) then
1139      write(*,*) "Unable to allocate memory for the array: xOverlap(:)"
1140      stop
1141    endif
1142    allocate(x_changed(nb,2), stat = error)
1143    if (error .ne. 0) then
1144      write(*,*) "Unable to allocate memory for the array: x_changed(:)"
1145      stop
1146    endif
1147    allocate(R_Correlation(nb), stat = error)
1148    if (error .ne. 0) then
1149      write(*,*) "Unable to allocate memory for the array: R_Correlation"
```

```
1150        stop
1151      endif
1152
1153      TempFort='for31_centroids'
1154      open(unit=31,file=TempFort,status='replace',action='write')
1155
1156      TempFort='for12_bead_disp'
1157      open(unit=12,file=TempFort,status='replace',action='write')
1158
1159      TempFort='for13_aveE2'
1160      open(unit=13,file=TempFort,status='replace',action='write')
1161
1162      TempFort='for14_E_V'
1163      open(unit=14,file=TempFort,status='replace',action='write')
1164
1165      TempFort='for15_aveE_aveV'
1166      open(unit=15,file=TempFort,status='replace',action='write')
1167
1168      TempFort='for30_c_overlap'
1169      open(unit=30,file=TempFort,status='replace',action='write')
1170
1171      ! And initializing the thermal density propagator table:
1172      call CreateTable(beta/float(nb))
1173
1174      ! Setting the cartesian bins to zero
1175      elecbin(:,:,:) = 0
1176      posibin(:,:,:) = 0
1177
1178      ! Initializing radial bin for e+ and e- separation
1179      rbinmax = 10.0
1180      radbin(:) = 0.0
1181      call binit(nbins, rbinmax, rforbin)
1182
1183      ! Zeroing the centroid positions
1184      xc1(:) = 0.0
1185      xc2(:) = 0.0
1186      xc(:) = 0.0
1187
1188      ! Initializing variables for g(r) calculation
1189      gr_dist = 20.0
1190      gr_pt(:) = 0.0
1191      gr_bead_count = 0
1192      gr_cm_count = 0
1193      gr_bead_bin(:) = 0.0
1194      gr_cm_bin(:) = 0.0
1195
1196      ! Initializing energy variables
1197      energy_ave = 0.0d0
```

```
1198      energyV_ave = 0.0d0
1199      energy2_ave = 0.0d0
1200      energyV2_ave = 0.0d0
1201      energy_count = 0
1202      x_changed(:,:) = .TRUE.
1203      xEnergyOld(:,:) = 0.0
1204      xEnergyNew(:,:) = 0.0
1205
1206      ! Zero the bin for the bead correlation function:
1207      R_Correlation(:) = 0.0
1208
1209      ! Initialize positron overlap with charge density to zero
1210      c_overlap = 0.0
1211      c_overlap_zero = 0.0
1212      c_overlap_count = 0
1213
1214      ! Create an initial bead configuration:
1215      ! A gaussian distribution (i.e., free particle)
1216      call init_beads(ninit)
1217
1218      ! Initialize information for potential, charge density and gamma
1219      ! set the input and output unit numbers and read the main input file
1220      iui = 5
1221      iuo = 6
1222
1223      write(*,*) "Program PEZ: PIMC with Atomic Potl and Density Matrices"
1224      write(iuo,*) '***** positron-potential information *****'
1225      write(iuo,*) 'vcg datafile produced by fepot:'
1226      write(iuo,*) vcgFile
1227
1228      ! read in vcg file generated by fepot
1229      ! initialize cubic spline fit for potential and charge density
1230      call initVcg(vcgFile, potVcg, cdVcg, gamVcg, rv_, deg)
1231      tmat = transpose(inverse(rv_))
1232
1233      ! find inverse lattice vectors
1234      gv_ = potVcg%tmat
1235
1236      ! transpose
1237      rv = transpose(rv_)
1238      gv = transpose(gv_)
1239
1240      ! initialize routine to find electronic pseudopotential
1241      call init_pseudo(rv_, eptFile)
1242
1243      tolcs = 1.0d-12 ! cluster-size tolerance in the electron
1244                      ! psuedo-potl calculation
1245      lprint_e = 0    ! logical, can have clust_elec print more
```

```
1246                    ! information about the electron potential
1247
1248      periodic = .true.
1249
1250      ! find cluster contributing to electron pseudopotential
1251      call clust_elec(rv, gv, periodic, basis, itype, rmax_e, tolcs, &
1252                      iuo, lprint_e, ncvec, clus, ictyp, lclus)
1253      call clust_gofr(rv, gv, basis, itype, tolcs, iuo, lprint_e, ncvec, &
1254                      gr_clus, gr_nclus, ictyp, lclus, gr_dist, gr_rmax)
1255
1256      acsum = 0.0d0  ! measure of acceptance rate
1257
1258    end subroutine Initialize
1259
1260    !______RANDOM NUMBER GENERATING FUNCTION____!
1261
1262    double precision function ran1(idum)
1263    implicit none
1264    double precision ::  r(97)
1265    integer, intent(IN) :: idum
1266    save
1267    integer, parameter :: M1=259200,IA1=7141,IC1=54773
1268    real, parameter :: RM1=1.0d0/M1
1269    integer, parameter :: M2=134456,IA2=8121,IC2=28411
1270    real, parameter :: RM2=1.0d0/M2
1271    integer, parameter :: M3=243000,IA3=4561,IC3=51349
1272    integer :: IX1, IX2, IX3, jjj
1273    integer ::  iff=0
1274    if (idum < 0 .or. iff == 0) then
1275      iff = 1
1276      IX1 = mod(IC1-idum,M1)
1277      IX1 = mod(IA1*IX1+IC1,M1)
1278      IX2 = mod(IX1,M2)
1279      IX1 = mod(IA1*IX1+IC1,M1)
1280      IX3 = mod(IX1,M3)
1281      do jjj = 1,97
1282        IX1 = mod(IA1*IX1+IC1,M1)
1283        IX2 = mod(IA2*IX2+IC2,M2)
1284        r(jjj) = (dfloat(IX1)+dfloat(IX2)*RM2)*RM1
1285      end do
1286    end if
1287    IX1 = mod(IA1*IX1+IC1,M1)
1288    IX2 = mod(IA2*IX2+IC2,M2)
1289    IX3 = mod(IA3*IX3+IC3,M3)
1290    jjj = 1+(97*IX3)/M3
1291    if (jjj > 97 .or. jjj < 1) PAUSE
1292    ran1 = r(jjj)
1293    r(jjj) = (dfloat(IX1)+dfloat(IX2)*RM2)*RM1
```

```
1294    end function ran1
1295
1296    end PROGRAM PIMC
```

Acknowledgements

I am sincerely grateful to Professor Amy L. R. Bug, my thesis advisor, for her constant support and assistance. Ever since my first semester when she taught me about the twin paradox and EPR, piquing my interest in physics, she has encouraged me in my classwork and has enabled me to be involved in exciting research as an undergraduate.

I also thank Phillip A. Sterne of Lawrence Livermore National Laboratory (LLNL) for his advice and for the use of his computing resources. Being able to spend a week at LLNL and to meet scientists using similar methods to ours was invaluable to this thesis.

I am also grateful to my Swarthmore physics professors. Their dedication to their students has given me an excellent undergraduate physics education; not only have my classes been challenging and exciting, but it has been wonderful to have my professors constantly available to answer my questions and give me new questions to think about.

Finally, I thank my parents, for helping me pursue my dreams and for trying to understand them.

References

[1] Y. Ne'eman and Y. Kirsh, *The Particle Hunters* (University Press, Cambridge, 1996), p. 65.

[2] A. Rich, Rev. Mod. Phys. **53**, 127 (1981).

[3] S. J. Wang, B. Wang, J. Zhu, Z. Wang, Y. Q. Dai and C. Q. He, Mater. Sci. Forum 363-365, 219 (2001).

[4] O. Halpern, Phys. Rev. **88**, 164 (1952).

[5] R. H. Howell, Sci. and Tech. Rev. (December 1998). Retrieved October 30, 2001, from LLNL, Positron Facility Website:
http://www.llnl.gov/str/Howell.html

[6] C. G. Fischer, S. H. Connell, P. G. Coleman, F. Malik, D. T. Britton, J. P. F. Sellschop, Appl. Surf. Sci. **149**, 221 (1999).

[7] D. J. Griffiths, *Introduction to Electrodynamics* (Prentice Hall, New Jersey, 1999), p. 464.

[8] D. Knapp. Retrieved October 30, 2001, from LLNL, Electron-Positron Beam Facility Website:
http://www-phys.llnl.gov/H_Div/Positrons/PositronFacility.html

[9] J. Nissilä, K. Saarinen, and P. Hautojärvi, Phys. Rev. B **63**, 165202 (2001).

[10] A. Dupasquier, "Quasipositronium in Liquids and Solids," in *Positron Solid-State Physics*, edited by W. Brandt and A. Dupasquier (North Holland, New York, 1983).

[11] M. J. Puska, S. Mäkinen, M. Manninen, and R. M. Nieminen, Phys. Rev. B **39**, 7666 (1988).

[12] J. H. Hartley, R. H. Howell, P. Asoka-Kumar, P. A. Sterne, D. Akers, A. Denison, Appl. Surf. Sci. **149**, 204 (1999).

[13] H. Nakanishi, S. J. Wang, and Y. C. Jean, in *Positron Annihilation Studies of Fluids*, edited by S. C. Sharma (World Scientific, Singapore, 1988).

[14] Y. C. Jean, in *Positron Spectroscopy of Solids*, edited by A. Dupasquier and A. P. Mills, Jr. (IOS, Washington, D.C., 1995).

[15] J. M. Thijssen, *Computational Physics* (Cambridge University Press, New York, 1999), Chapters 10, 12.

[16] K. Binder, in *Encyclopedia of Applied Physics*, edited by G. Trigg (VCH Publishers, New York, 1994).

[17] R. Shankar, *Principles of Quantum Mechanics* (Plenum Press, New York, 1994), Chap. 21.

[18] T. Reese and B. N. Miller, Phys. Rev. E **47**, 2581 (1993).

[19] H. Schmitz and F. Müller-Plathe, J. Chem. Phys. **112**, 1040 (2000).

[20] L. Larrimore, R. McFarland, P. Sterne, and A. Bug, J. Chem. Phys. **113**, 10642 (2000).

[21] D. J. Griffiths, *Introduction to Elementary Particles* (Harper and Row, New York, 1987), Chap. 5.

[22] M. J. Puska and R. M. Nieminen, Phys. Rev. B **46**, 1278 (1992).

[23] C. Kittel. *Introduction to Solid State Physics* (John Wiley & Sons, New York, 1996).

[24] E. M. Gullikson, Phys. Rev. B **39**, 6128 (1989).

[25] P. Rice-Evans, M. Moussavi-Madani, K. U. Rao, D. T. Britton, and B. P. Cowan, Phys. Rev. B **34**, 6117 (1986).

[26] Y. C. Jean, C. Yu, and D.-M. Zhou, Phys. Rev. B **32** 4213 (1985).

[27] E. M. Gullikson and A. P. Mills, Phys. Rev. B **39**, 6121 (1989).

[28] E. J. Woll, in *Positron Annihilation*, edited by A. T. Stewart and L. O. Roellig, (Academic Press, New York, 1967).

[29] D. C. Liu and W. K. Roberts, Phys. Rev. **132**, 1633 (1963).

[30] Y. Nagai, Y. Nagashima, and T. Hyodo, Phys. Rev. B **60**, 7677 (1999).

[31] K. Ohno, K. Esfarjani, and Y. Kawazoe, *Computational Materials Science* (Springer, New York, 1999).

[32] D. F. Coker, B. J. Berne, and D. Thirumalai, J. Chem. Phys. **86**, 5689 (1987).

[33] M. H. Müser and B. J. Berne, J. Chem. Phys. **107**, 571 (1997).

[34] E. L. Pollock, Computational Physics Communications **52**, 49 (1988).

[35] E. L. Pollock (personal communication).

[36] P. Hastings, *Positronium Annihilation in Insulating Solids*, Unpublished Undergraduate Thesis, Swarthmore College, 2001.

[37] B. Space, D. F. Coker, Z. H. Liu, B. J. Berne, and G. Martyna, J. Chem. Phys. **97**, 2002 (1992).

[38] T. Gibson, J. Phys. B **23**, 767 (1990).

[39] J. A. Barker, J. Chem. Phys. **70**, 2914 (1979).

[40] D. M. Ceperley, Rev. Mod. Phys. **67** 279 (1995).

[41] M. J. Puska and R. M. Nieminen, Rev. Mod. Phys. **66**, 841 (1994).

[42] M. J. Puska, S. Mäkinen, M. Manninen, and R. M. Nieminen, Phys. Rev. B **39**, 7666 (1989).

[43] D. A. McQuarrie, *Statistical Mechanics* (Harper and Row, New York, 1976).

[44] V. K. Decyk and D. E. Dauger. Retrieved January 9, 2001 from UCLA Website: http://exodus.physics.ucla.edu/appleseed/appleseed_report01.pdf

[45] E. M. Gullikson, A. P. Mills, and E. G. McRae, Phys. Rev. B **37**, 588 (1988).

[46] M. R. Spiegel and J. Liu, *Schaum's Outlines: Mathematical Handbook of Formulas and Tables* (McGraw-Hill, New York, 1999), Part A Sect. 7.

[47] A. L. R. Bug (personal communication).

[48] E. Wimmer. Retrieved August 4, 2000 from Molecular Simulations Inc. Website: http://www.msi.com/materials/tech/qm/dft.html

[49] P. A. Sterne and J. H. Kaiser, Phys. Rev. B **43**, 13892 (1991).

[50] A. Dupasquier, "Positroniumlike Systems in Solids," in *Positron Solid-State Physics*, edited by W. Brandt and A. Dupasquier (North Holland, New York, 1983).