Open Source Code for Path Integral Monte Carlo

Bryan K. Clark and Kenneth P. Esler, Jr.
Department of Physics, University of Illinois at Urbana-Champaign

Supported by the National Science Foundation under Award Number DMR-03 25939 ITR,
via the Materials Computation Center at the University of Illinois at Urbana-Champaign

Motivation

Path Integral Monte Carlo (PIMC) is a numerical method to study finite-temperature systems at the quantum level. The imaginary-time path integral formalism, first introduced by Feynman, naturally includes correlation effects. Transforming this into a numerical algorithm yields exact results for bosons and very high accuracy results for fermions. It is a very general method applicable to such diverse systems as superfluid helium, exotic phases of hydrogen under extreme pressures, and the Wigner crystal. The PIMC method, and related quantum Monte Carlo methods, have provided the gold-standard against which other methods are judged.

The PIMC method is in a state of continual development as a whole range of new techniques are being created. To facilitate this, it is helpful to be able to quickly prototype new techniques and algorithms without sacrificing computational efficiency. Toward this end, we introduce here an object-oriented Path Integral Monte Carlo Code which we call PIMC++. This code has been designed with the goal of becoming a new standard code for the PIMC community. While modular and efficient, it is sufficiently general to be useful for a broad range of physical systems. To ensure accessibility and to engage a wide community, the code will be released as free and open-source software.

1 Path Integral Monte Carlo

1.1 Introduction

Path Integral Monte Carlo is a means of computing thermal expectation values of observables in many-body quantum systems, utilizing the thermal density matrix, $\rho(\mathbf{R},\mathbf{R}';\beta)$, defined by

$$\rho(\mathbf{R},\mathbf{R}';\beta) \equiv \left\langle \mathbf{R} \left| e^{-\beta\hat{H}} \right| \mathbf{R}' \right\rangle . \tag{1}$$

Any thermal expectation value may be computed as

$$\langle\hat{\mathcal{O}}\rangle_{\text{thermal}} = \frac{1}{Z}\int d\mathbf{R}\,\hat{\mathcal{O}}\rho(\mathbf{R},\mathbf{R}';\beta). \tag{2}$$

We can rewrite ρ as

$$\begin{aligned}\rho(\mathbf{R},\mathbf{R}';\beta) &= \int d\mathbf{R}_1\,d\mathbf{R}_2\,\ldots d\mathbf{R}_{M-1} \\ &\quad \times\rho(\mathbf{R},\mathbf{R}_1;\frac{\beta}{M})\rho(\mathbf{R}_1,\mathbf{R}_2;\frac{\beta}{M})\ldots\rho(\mathbf{R}_{M-1},\mathbf{R}';\frac{\beta}{M}).\end{aligned} \tag{3}$$

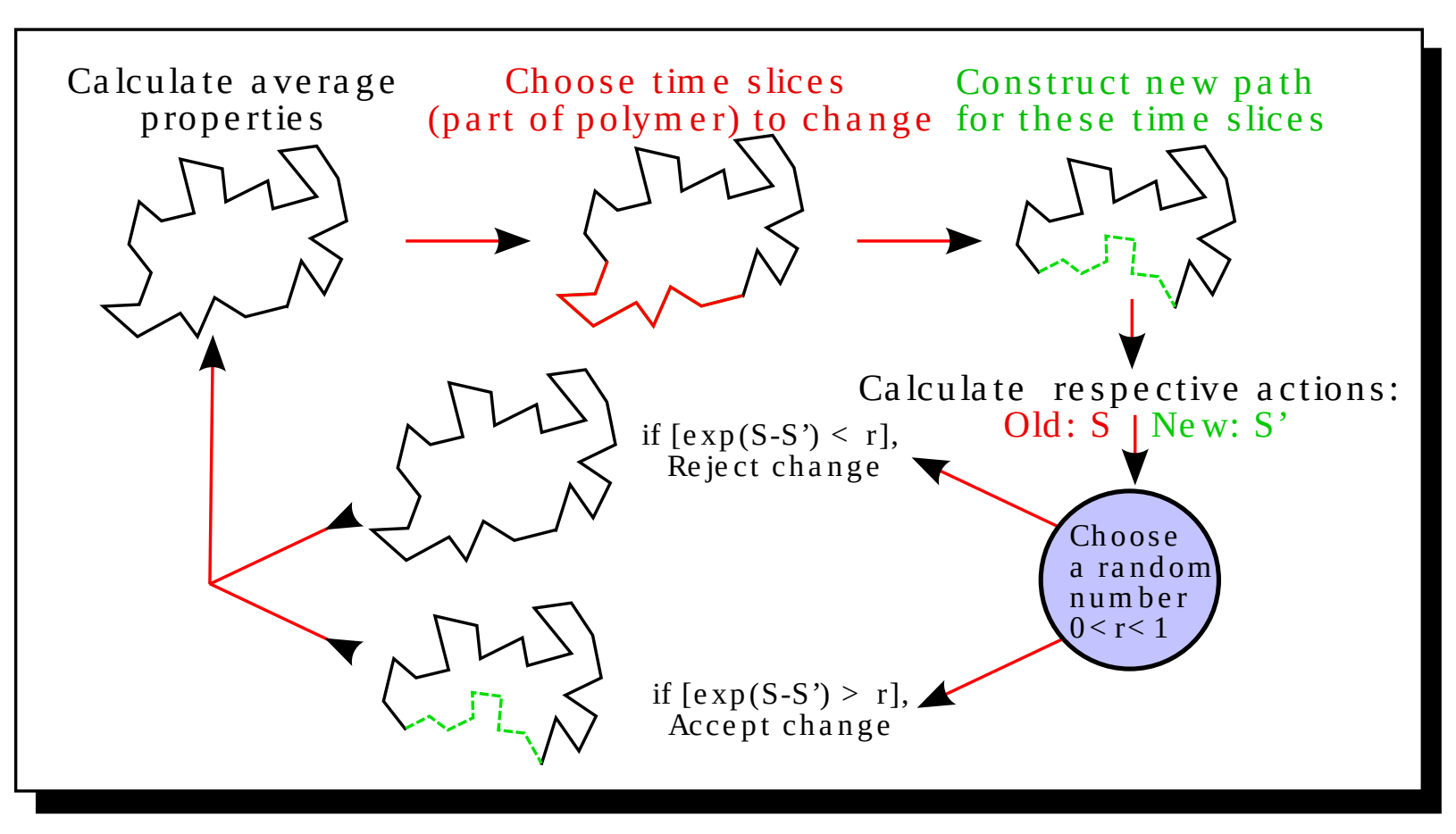
This expansion allows the use of a short-time approximation to the density matrix, which becomes exact as $M \to \infty$. We may then combine (2) and (3), and use Metropolis Monte Carlo to perform the high-dimensional integration.

1.2 Quantum-Classical Isomorphism

The integrand is a function of $3N$ spatial variables, at each of M points in "imaginary time." Taken together, these variables sweep out N discrete particle paths which return to themselves. Graphically these paths appear similar to classical ring polymers. Viewed this way, the PIMC simulation involves sampling many configurations of these ring polymers. Equilibrium averages of measurable quantities are then calculated by averages over these configurations. Properties of the quantum system then become describable in this classical language. Below we summarize this correspondence [1].

Physical quantity	Polymer isomorphism
bose condensation	delocalization of ends of open polymer
boson statistics	allowing polymers to hook up in any possible way
degeneracy temp.	a condition in which polymers are dense enough and extended enough that they touch and can exchange
density	the bead density
free energy	free energy of system of ring polymers
imaginary velocity	bond vector
kinetic energy	negative spring energy
momentum correlation	bond-bond correlation
momentum distribution	Fourier transform of end-end distribution
pair correlation	pair-correlation function between beads at the same "time"
particle	ring polymer
superfluid density	the mean-squared winding number
superfluid state	a state in which a finite fraction of polymers are hooked together in polymers of macroscopic size
temperature	inverse polymer length, inverse coupling constant for the inter-polymer potential, and spring constant between neighboring beads
thermal wavelength	polymer extension

1.3 Simplified Algorithm


Calculate average properties
Choose time slices (part of polymer) to change
Construct new path for these time slices
Calculate respective actions:
Old: S
New: S'
Choose a random number 0 < r < 1
if [exp(S-S') < r], Reject change
if [exp(S-S') > r], Accept change

1.4 The Pair Approximation

In order to perform the PIMC simulation, we need an approximation for $\rho(\mathbf{R},\mathbf{R}';\tau)$, where $\tau = \beta/M$. We use the pair-product approximation,

$$\rho(\mathbf{R},\mathbf{R}';\tau) = \rho_0(\mathbf{R},\mathbf{R}';\tau)\prod_{i<j}\rho_I(\mathbf{r}_i-\mathbf{r}_j,\mathbf{r}'_i-\mathbf{r}'_j;\tau), \tag{4}$$

where ρ_0 is the free-particle density matrix, and ρ_I is the interaction part of the exact density matrix for the pair of particles i and j. This approximation includes all two-body correlation effects exactly, leaving three-body and higher effects to be included through the PIMC simulation. The approximation is very well-controlled, having a time-step error of $\mathcal{O}(\tau^3)$.
The two-body density matrix for a pair of particles interacting with a central potential can be calculated exactly with squarer++ and tabulated for use in PIMC simulation.

2 PIMC++

2.1 General Features

- modular, object-oriented design for easy addition of new moves, observables, and actions
- free and periodic boundary conditions as well as twist-averaged boundary conditions for reducing finite-size effects
- Bose, Fermi, and distiguishable-particle statistics
- high-accuracy pair actions
- optimized long-range/short-range action breakup
- embedded conjugate-gradient plane-wave pseudohamiltonian code for calculation of ground-state nodes
- specialized moves to enhance ergodicity
- calculation of properties involving off-diagonal terms of the density matrix

2.2 Code Design

PIMC++ is written in C++, with some calls to C and F77 libraries. It makes extensive use of object-oriented design principles. The main code is broken into four broad object types:

1. Path representation: specifies the location of all the particles at each time slice (i.e., the classical polymers). Also stores important quantities, such as the permutations and species information.
2. Moves: responsible for attempting random changes to the paths.
3. Actions: give the relative probability for a given path. This is used to decide whether to accept or reject a proposed move.
4. Observables: measure the equilibrium averages of physical quantities. Comprising the output of the simulation, they include total energies, correlation functions, structure factors, superfluid fractions, etc.

All these objects make use of inheritance to be independent and interchangeable, allowing new types to be created and inserted with little effort.

2.3 Input and Output

In order to facilitate the construction of object-oriented design, PIMC++ includes a novel hierarchical I/O library. Using a structured ASCII format for simple data, and NCSA's cross-platform HDF5 binary format for heavy data, the library abstractly accesses data in either format transparently at the application level. By segregating data through hierarchy, it allows application objects to read and write their own data in a modular fashion.

2.4 Parallelization

In order to tackle cutting-edge problems, PIMC++ must be able to scale to utilize modern supercomputers. In order to scale efficiently to hundreds of processors, it implements two independent modes of parallelism, which may be used in conjuction.

1. Natural parallelism: Monte Carlo is *naturally* parallel. Simple cloning of simulation runs with different random seeds yields better statistics with no communication overhead.
2. Time slice partitioning: Very large systems may need to be divided due to storage and equilibration time limitations. Time-slice decomposition allocates contiguous chunks of paths onto different processors, requiring only occasional communication.

These two modes, when used together, allow efficient scaling on very large clusters and supercomputers.

2.5 Pseudohamiltonians

In the pseudopotential approximation, the scattering effects of the atomic nucleus and core electrons upon the valence electrons are mimicked by an effective potential, yielding the single-electron Hamiltonian of the form

$$h(\mathbf{r}) = -\frac{1}{2}\nabla^2 + V(r). \tag{5}$$

The pseudohamiltonian approximation adds additional flexibility by allowing the mass to vary as a function of position and direction,

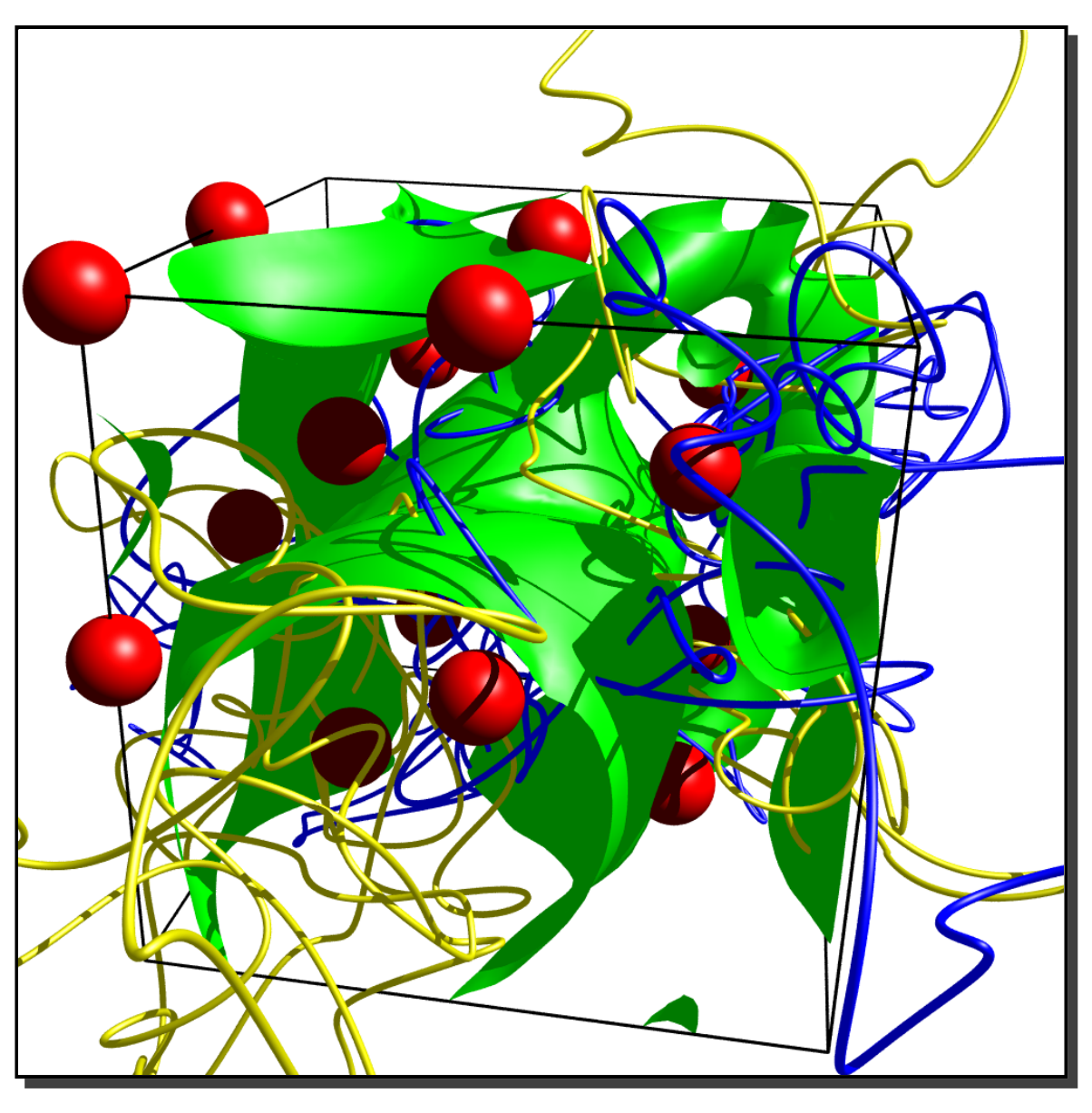
$$h^{\text{PH}}(\mathbf{r}) = -\frac{1}{2}\nabla[1+a(r)]\cdot\nabla + \frac{b(r)\hat{L}^2}{2r^2} + V(r), \tag{6}$$

where $a(r)$ and $b(r)$ are the radial and tangential inverse masses, respectively. In order for the eigenspectrum to be bounded from below, we must have that

$$\begin{aligned} A(r) &\equiv 1+a(r) > 0 \qquad (7)\\ B(r) &\equiv 1+a(r)+b(r) > 0. \qquad (8)\end{aligned}$$

2.6 Fermions

Because fermions are antisymmetric, the integral over their partition function can not be naively interpreted as a probability distribution, inhibiting using Metropolis for these systems. The canonical methods for dealing with this problem are the restricted path or restricted phase methods. These methods work by restricting the integration to the region in which a trial density matrix is positive. While approximate, these methods have proven to be very accurate for many systems. PIMC++ implements both of these methods in advanced ways.
The following represents the nodal surface for a single electron at one time slice in a 16-atom BCC sodium simulation.



3 Supporting Tools

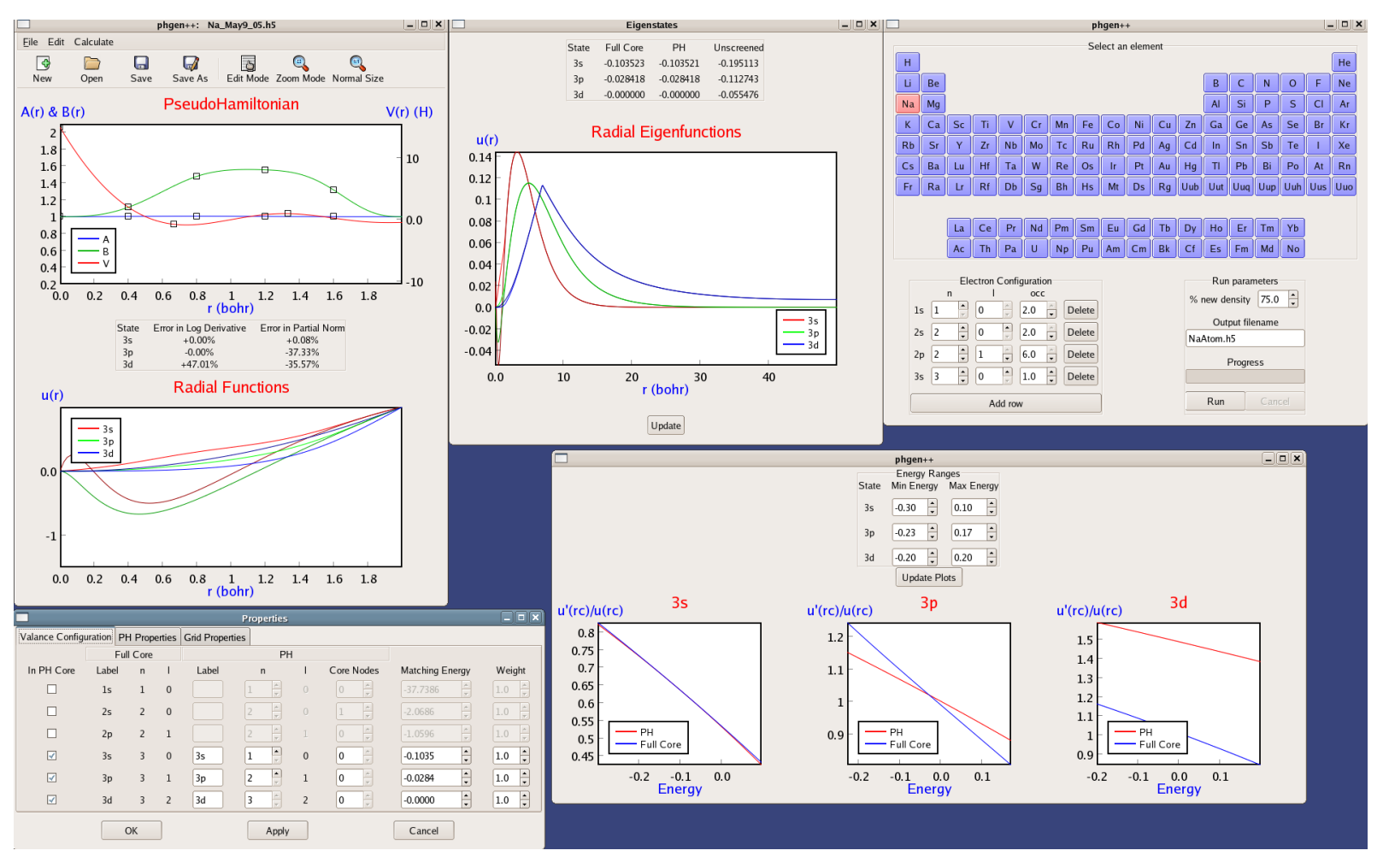
We have developed a complete tool suite supplementing PIMC++, which facilitates preparing, running, and analyzing the output of the code.

3.1 atom++

This is a simple self-consistent DFT code for atoms using the Vosko-Wilk-Nusair exchange-correlation potential. It uses scalar-relativistic corrections and allows restricted LDA calculations. This high-precision code has been verified against the NIST Atomic Reference Data[6]. This provides the all-electron properties which phgen++ aims to reproduce with effective potentials.

3.2 phgen++

We have developed an easy-to-use GUI interface for interactively developing pseudohamiltonians. The user simply opens the output from atom++, selects the states to keep in the valence shell and a core radius, then modifies the $A(r)$, $B(r)$, and $V(r)$ functions interactively until the desired properties are achieved. With each modification, the radial equations are integrated and plotted in real time, and the errors in the logarithmic derivatives and partial norms are displayed. Once a suitable PH is found, it is unscreened and saved to file.



3.3 squarer++ / fitter++

For each pair of particle types, we must calculate a corresponding pair density matrix. This is the function of squarer++. Its features include:

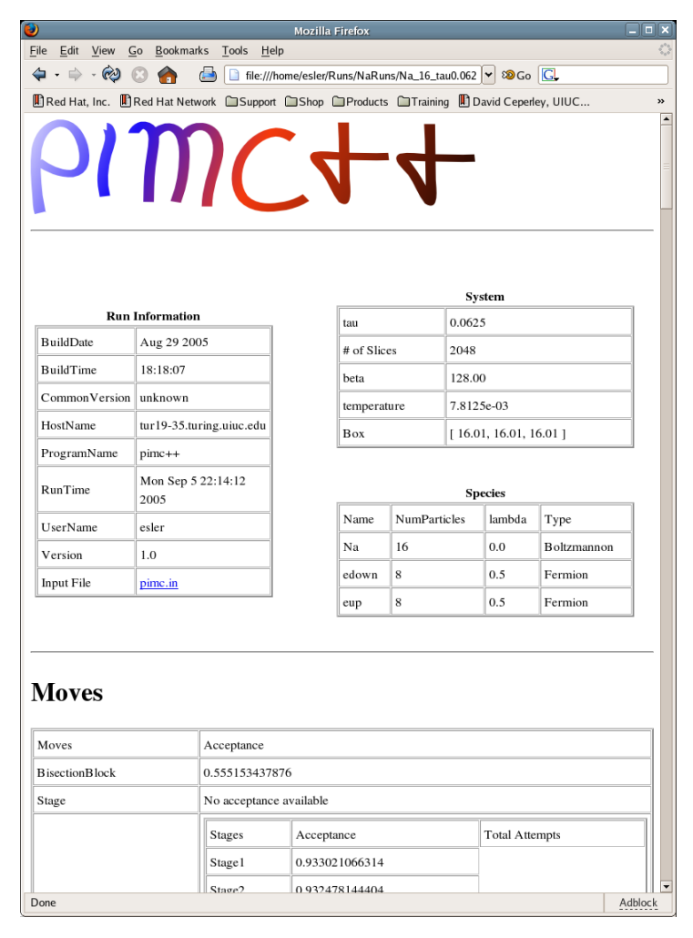
- adaptive integration techniques, logarithmic representation, bicubic spline interpolation, partial-wave extrapolation, and optimized grids to increase accuracy
- full parallelization using MPI: scalable to hundreds of processors to allow quick turn-around time
- general and applicable to all central potentials and pseudohamiltonians
- stores output in binary HDF5 format to maintain cross-platform accuracy

fitter++ takes output from squarer++ and produces a number of different fits for fast evaluation in PIMC++.

3.4 Data Analysis

The analysis code reads the output file and summarizes the output in HTML, allowing web publication. Its features include

- written in Python for easy modification
- automatically determines error bars
- generates trace plots of relevant quantities allowing visual detection of simulation anomalies
- coherently aggregates data from parallel runs
- creates Postscript graphs
- captures important run information such as run time, build, and code version



PIMC++

Run Information

BuildDate	Aug 29 2005
BuildTime	18:18:07
CommonVersion	unknown
HostName	tur19-35.turing.uiuc.edu
ProgramName	pimc++
RunTime	Mon Sep 5 22:14:12 2005
UserName	esler
Version	1.0
Input File	pimc.in

System

tau	0.0625
# of Slices	2048
beta	128.00
temperature	7.8125e-03
Box	[16.01, 16.01, 16.01]

Species

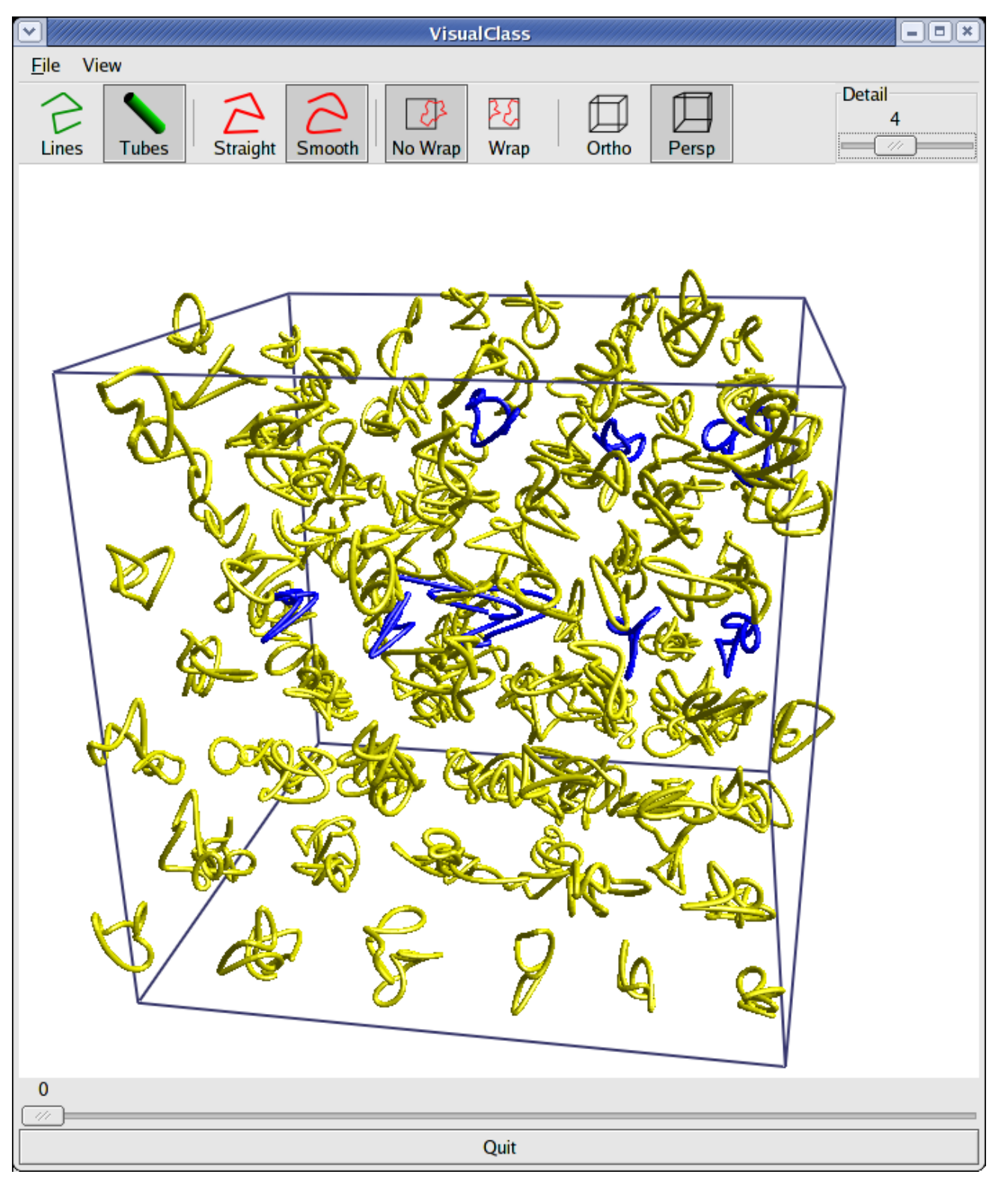
Name	NumParticles	lambda	Type
Na	16	0.0	Boltzmannon
edown	8	0.5	Fermion
eup	8	0.5	Fermion

Moves

Moves	Acceptance
BisectionBlock	0.555153437876
Stage	No acceptance available

Stages	Acceptance	Total Attempts
Stage1	0.933021066314	
Stage2	0.932478144404	

3.5 pathvis++

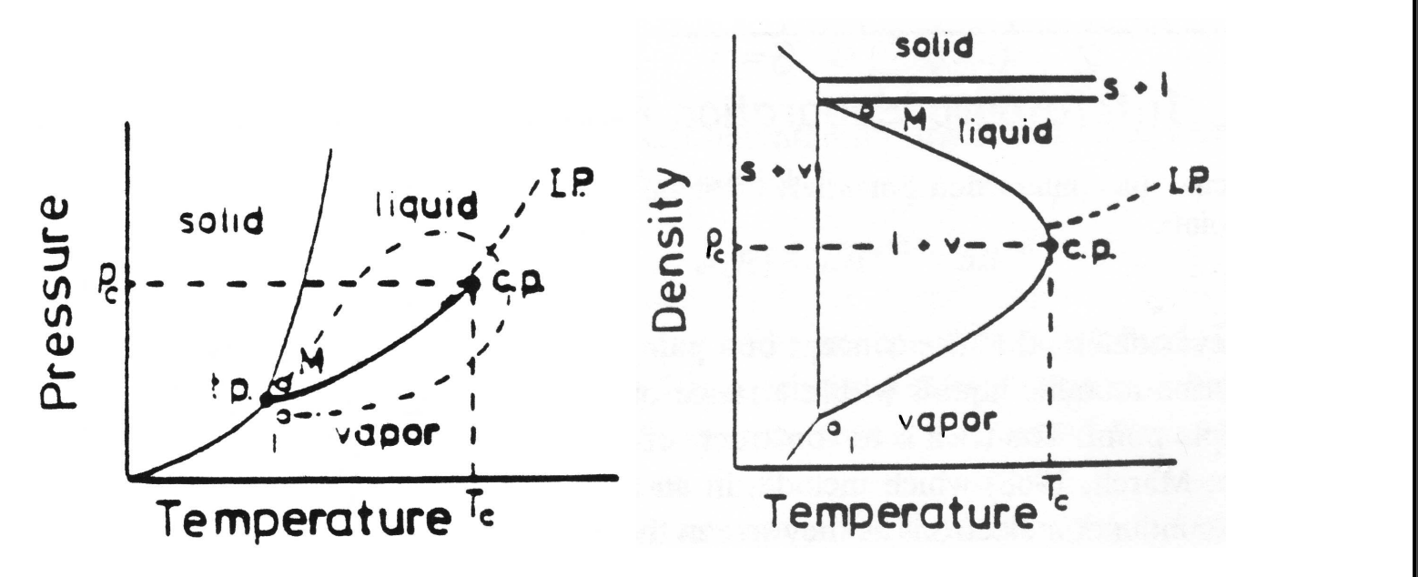


pathvis++ is an interactive 3D visualization package for viewing paths recorded from PIMC++. Its features include:

- Fully-interactive 3D visualization with OpenGL
- Fourier smoothing for easy viewing
- PovRay export for publication-quality ray traced output
- Exports MPEG4 movies for pedagogy
- Extremely useful for teaching and debugging
- Calculates isosurfaces for study of nodal surfaces.

4 Example Applications

4.1 Metal-Insulator transitions in fluid alkali metals



- Alkali metals have a metal-insulator transition near their liquid-vapor critical point.[7]
- The mechanism driving the transition is not well understood.
- Quantum effects and electron correlation are important.
- Transitions occur at a significant fraction of the Fermi temperature: thermal effects are important!
- PHs require only 1 electron per atom, so calculation should be no more computationally demanding than hydrogen.

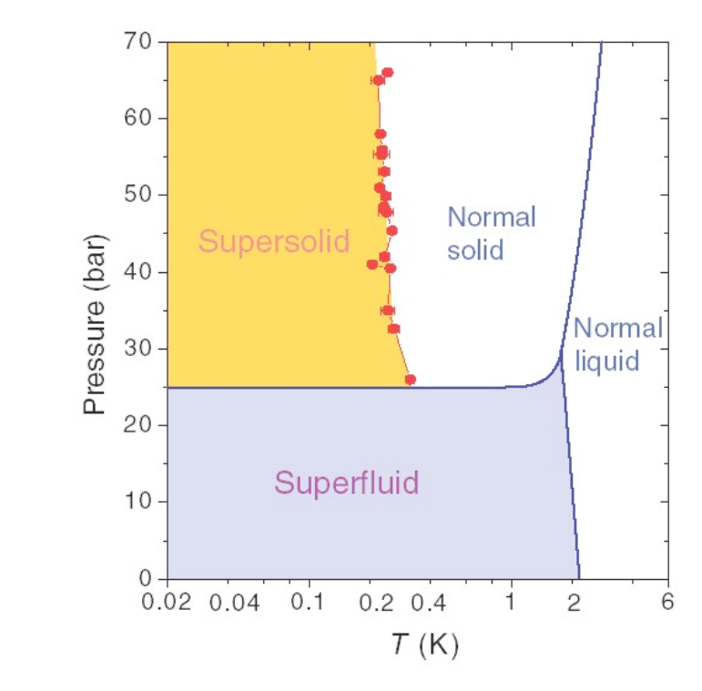
Validation: BCC cohesive energy

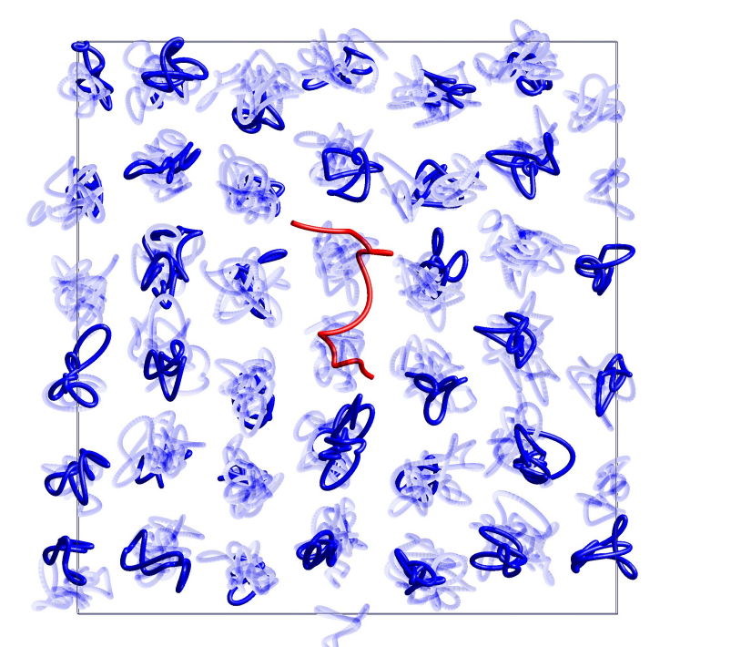
- Used a 16-atom cell with periodic boundary conditions.
- Utilized ground-state nodes from a custom plane-wave conjugate-gradient PH code.
- Error due to finite-size effects. Used LDA to approximate kinetic energy corrections.
- Twist-averaged boundary conditions used to integrate over the first Brillouin zone.

Results

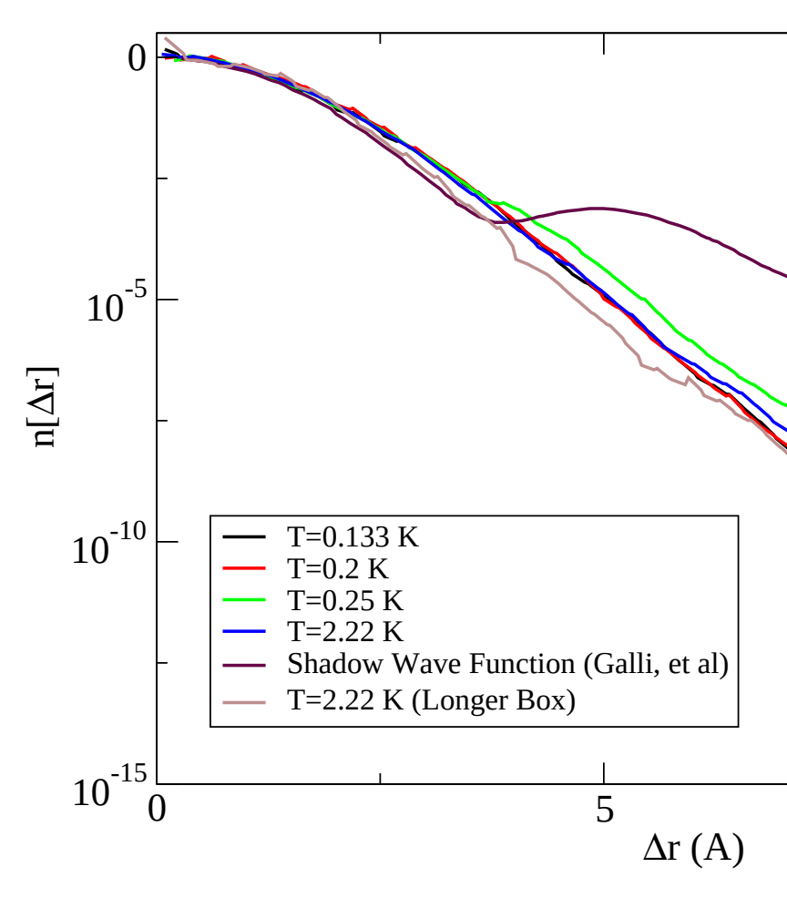
Source	E_c (eV)
Experiment [8]	1.13
PIMC++	1.268 ± 0.015
DMC[9]	0.99

4.2 Supersolid?: Off-diagonal long range order of solid ^{4}He





Kim and Chan[4] have observed non-classical rotational intertia in bulk solid ^{4}He . They attribute this to a "supersolid" phase of the equilibrium bulk ^{4}He as shown in their above phase diagram. Their experimental evidence indicates that topological properties of the system are important in observing the effect.



Using PIMC++, we calculate the off-diagonal terms of the density matrix in solid ^{4}He. This allows us to establish whether the system is Bose-condensed:

$$\lim_{|\Delta r|\to\infty} \rho_1(\Delta r) > 0$$

where ρ_1 is the single body density matrix.
The bottom five curves are our results and the upper curve is a comparison with a calculation done by D. Galli, et al[5] using the shadow wave function. As $r \to \infty$ the value of the off-diagonal terms decay very rapidly toward 0. Consequently, our results indicate there is no BEC in this system. This supplies evidence that a bulk equilibrium supersolid phase in ^{4}He is not responsible for Kim and Chan's results.

References

[1] D.M. Ceperley, Rev. Mod. Phys **67**, #2 (1995)
[2] G.B. Bachelet, D.M. Ceperley, and M.G.B. Chiocchetti, Phys. Rev. Lett. **62**, 2088 (1989)
[3] Vosko, Wilk, Nusair, Can. J. Phys. **58**, 1200 (1980)
[4] E. Kim and M. H. Chan, Science **305**, 1941 (2004)
[5] D. Galli, M. Rossi, and L. Reatto, Phys. Rev. B **71**, 140506 (2005)
[6] *Atomic Reference Data for Electron Structure Calculations*, **http://physics.nist.gov/PhysRefData/DFTdata/contents.html**
[7] Friedrich Hensel and William W. Warren, Jr. 1999. *Fluid metals: the liquid-vapor transition of metals*, Princeton: Princeton University Press
[8] K.A. Gschneidner, Jr., in *Solid State Physics*, edited by F. Seitz and D. Turnbull (Academic, New York, 1964), Vol. 16, p. 344
[9] R. Maezono, M. D. Towler, Y. Lee, and R. J. Needs, Phys. Rev. B **68**, 165103 (2003)

Acknowledgments

This work has been performed under the guidance of Professor D.M. Ceperley. Calculations were performed on NCSA's *Tungsten* cluster and UIUC's CSE *Turing* cluster.