SANDIA REPORT

SAND98-2840

Unlimited Release

Printed January 1999

RECEIVED

~~MS-0619~~

~~Review & Approval Desk, 15102~~
~~For DOE/OSTI~~

Density Functional Theory for Classical Fluids at Complex Interfaces: (TRAMONTO 1.0: Methods, Applications, & User's Guide)

Laura J.D. Frink and Andrew G. Salinger

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A11
Microfiche copy: A01

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SAND98-2840
Unlimited Release
Printed January 1999

Density Functional Theory for Classical Fluids at Complex Interfaces:

(TRAMONTO 1.0: Methods, Applications, & User's Guide)

Laura J.D. Frink
Materials Simulation Science Department

Andrew G. Salinger
Parallel Computational Sciences Department

Sandia National Laboratories
Albuquerque, NM 87185-1111

Abstract

This report contains a summary of work on a 2 and 3-dimensional (2D/3D) parallel density functional theory (DFT) code for inhomogeneous fluids. The behavior of fluids near surfaces or macromolecules is critically important in a wide range of physical systems including adsorbants, colloidal dispersions, sensors, and self-assembled systems. One of the premier tools for studying inhomogeneous fluids at the molecular length scale is DFT. However, until this LDRD was undertaken, no general purpose 2D or 3D codes existed. The state of the art was to perform 1D calculations, and then attempt to generalize the results (by using superposition approximations for example) to more complex systems. While this approach is useful, 2D and 3D calculations will necessarily give a much deeper insight into the structure and behavior of fluids near surfaces that are complex either in their geometry or surface chemical patterns. In this report we summarize the development of a general multipurpose 2D / 3D capability that may be easily extended to a wide range of surfaces from zeolites to polymers.

Intentionally Left Blank

Acknowledgements

LJD Frink would like to thank both Frank van Swol both for his collaboration in developing a preliminary 1D DFT code, and his collaboration in exploring surface forces in a variety of complex systems, and Jeff Brinker for providing both experimental results from beam bending experiments and many helpful discussions. The authors would also like to thank Steve Plimpton for supplying the recursive spectral bisection partitioning algorithm, and Ray Tuminaro for helping to connect to the Aztec library.

Intentionally Left Blank

Contents

Section I. Introduction.....	9
Section II. Density Functional Theory.....	11
II.A. Hard Sphere Fluids.....	12
II.B. Lennard-Jones Fluids.....	14
II.C. Electrolyte Fluids / Charged Systems.....	15
II.D. Ideal Gas and Poisson-Boltzmann Electrolyte.....	16
II.E. Adsorption, Surface Free Energy and Solvation Forces.....	17
II.F. Sum Rules.....	20
II.F.1 Charged Parallel Plate	20
II.F.2 Charged Cylindrical Surfaces	23
Section III. Numerical Methods	25
III.A. Meshing and Quadrature.....	25
III.B. Solution Algorithm.....	26
III.C. Algorithms and Scaling.....	29
III.D. Parallelization and Scaling.....	33
Section IV. Code Performance.....	36
IV.A. Algorithms for Improved Performance	36
IV.A.1 Mesh Coarsening	36
IV.A.2 Jacobian Coarsening.....	37
IV.A.3 Approximate Pre-Calculated Jacobian.....	38
IV.B. Performance Timings.....	38
Section V. Summary of Applications.....	45
V.A. Characterization of Microporous Thin Films.....	45
V.B. Surface Forces: Adhesion and Surface Roughness.....	48
V.C. Wetting and Self-Assembly / Chemically Heterogeneous Surface	51
V.D. Forces between Polyelectrolytes in Solution.....	55

Section VI. Users Guide to TRAMONTO.....	59
VI.A. Input Files and Parameter Description	59
VI.B Output Files.....	72
References.....	74

Section I. Introduction

This report contains a summary of the development of a massively parallel capability (TRAMONTO) for performing density functional theory (DFT) calculations for inhomogeneous fluid systems in complex geometries. Due to the complexity and memory requirements of the algorithms, massively parallel computing strategies were applied. The current version of the code can be used to perform calculations on fluids near a variety of surfaces ranging from infinite smooth parallel plates (a 1-D calculation) to collections of fixed atoms at known locations as might be the case in a zeolite (a 3-D calculation). The code can treat several model fluids in a variety of approximations including ideal gases, hard sphere fluids, Lennard-Jones fluids, charged hard sphere melts, finite sized electrolyte models, and point charge electrolytes.

DFT for inhomogeneous fluid systems has a similar theoretical basis as DFT for electronic structure calculations. In essence, it can be shown that there is a unique relationship between the external field that fluid molecules experience when they are near surfaces, or macromolecules, and the ultimate density distribution, $\rho(r)$ of the fluid molecules. Likewise in electronic systems there is a unique relationship between the electronic density distribution and the external field produced by the nuclei in the system. In the same way, while quantum monte carlo calculations can be used to check and test the underlying assumptions of electronic structure density functional theory, molecular simulations (molecular dynamics or monte carlo simulations) can be used to check the predictions of the DFT of inhomogeneous fluids.

The disadvantage of the inhomogeneous fluid DFT (as compared with electronic structure DFT) is that the exact Hamiltonian is not known. Nevertheless, a variety of functionals have been developed that reproduce molecular simulations quite well. The most accurate DFT approaches are nonlocal approaches. These are the principle focus of the work presented here.

The DFT starts with potentials that describe the interactions between molecules, and a functional that describes how those interactions contribute to the grand free energy of the fluid part of the system. This free energy functional is then minimized with respect to the density distribution. The minimum free energy corresponds to the equilibrium density distribution. Advantages over performing molecular simulations on identical systems include the ability to take advantage of symmetries in the geometry, the ease with which trace species and charged systems can be calculated, and the efficient generation of complete solutions curves as a function of parameters using continuation methods.

A density distribution for a simple one-dimensional (1D) calculation of a one-component hard sphere fluid in contact with a smooth planar hard wall of infinite extent is shown in Fig.I.1. The calculation is 1D because the fluid is uniform in the dimensions that are parallel to the surfaces, and so these integrals are done analytically. The equilibrium density of the fluid is seen to oscillate as a function of the

distance perpendicular to the wall, eventually decaying to its bulk value. In addition to the raw density data, the code can predict adsorption, solvation forces, surface free energies, liquid-vapor equilibrium, wetting properties (contact angles), and phase changes associated with inhomogeneous fluids including wetting transitions, pre-wetting transitions, layering transitions, and solvation transitions.

The resources of massively parallel computers are needed for certain two-dimensional (2D) and most three-dimensional (3D) calculations. Since density gradients can be large, and much of the post-processed information requires very smooth results, a fine mesh is required. In addition, the long range interactions in the fluid produce nested integral equations which are expensive to calculate, and which result in large memory requirements.

The code TRAMONTO has been developed to rapidly and robustly solve the DFT equations for a number of fluid models in 1D, 2D, and 3D. A collocation approach on a regular, structured grid has been employed, and stencils for the integrals in the equations are precalculated. Newton's method is used to solve the equations. The linear system at each iteration is solved using the Aztec package, which greatly simplified the parallelization of the code. A number of sophisticated algorithms were implemented in order to make the 3D calculations feasible even on the ASCI-Red T10p machine. These include the ability to coarsen the mesh away from surfaces while retaining the efficiency of a structured mesh code, intelligent load balancing, the use of a lower accuracy Jacobian matrix, and the use of an approximate, pre-calculated Jacobian matrix.

The applications that have been pursued in the course of the development of TRAMONTO have included:

- ◆ analysis of adhesion in the presence of capillary condensation* [1],
- ◆ characterization of microporous thin films from beam bending experiments* [2],
- ◆ predicting the effects of surface roughness on solvation forces* [3],
- ◆ characterizing wetting of chemically heterogeneous substrates [4],
- ◆ predicting the phase space for self-assembly on templated heterogeneous surfaces [4], and
- ◆ predicting interactions of polyelectrolytes in electrolyte solution.

Ongoing applications include modeling self-assembled structures, predicting transport through ion channel proteins, and studying competitive adsorption in zeolites.

In the following sections, we outline the theory behind the DFT code (section II) as well as the numerical approach used (section III). We detail various approaches for improving code performance, and show the consequences of those improved algorithms (section IV). Summaries of several of the applications listed above are given in section V, and a brief users guide is presented in section VI.

* The calculations for these applications were done using a preliminary 1D DFT code with a completely different numerical basis than TRAMONTO. This 1D code was developed with LDRD funds, and was helpful both for establishing the relevance of the DFT approach to real problems at Sandia National Laboratories, and as a basis for comparison and debugging of the TRAMONTO code.

Section II Density Functional Theory

In this section the theory on which the TRAMONTO code is based is reviewed briefly. Four types of fluid models may be chosen as a basis for computations at this time. They include hard sphere fluids, Lennard-Jones fluids, and electrolytes with finite sized hard cores, and electrolytes in the Poisson-Boltzmann approximation. The theory for each of the fluid models along with the equations needed to calculate adsorption, solvation forces, and surface free energies is presented here. The approach of this report is to present an operational perspective for the DFT calculations. A more sophisticated and complete approach to the theory and underlying statistical mechanics can be found elsewhere[5].

All of the systems that can be treated with TRAMONTO are composed of a known fixed number of surfaces in contact with an open fluid (vapor or liquid) system. The appropriate ensemble for the fluid particles in this case is the grand canonical ensemble where the volume (V), temperature (T), and chemical potentials (μ_α) in the fluid are held constant. Examples of open fluid systems include adsorption where an adsorbant substrate is exposed to a bulk vapor at constant pressure and two colloidal particles interacting in a large liquid reservoir. By performing calculations in the grand canonical ensemble it is not necessary to simulate the surrounding bulk. Rather, we rely on the constancy of chemical potentials to establish proper material exchange.

In essence the DFT approach solves for the equilibrium fluid density in the external field produced by the fixed surfaces. It is therefore based on the functional minimization of the grand potential, Ω with respect to the density distributions ρ_α all species, α .

$$\left(\frac{\delta \Omega}{\delta \rho_\alpha(r)} \right)_{\mu, T} = 0 \quad (1)$$

The expression associated with this minimization is an Euler-Lagrange equation.

In all that follows, expressions for $\Omega[\rho]$ are given in a perturbation approach where different terms correspond to different physical contributions to the free energy. For example, the free energy of a Lennard-Jones fluid is written as the sum of ideal gas, hard core, and attractive contributions to the Helmholtz free energy, F .

$$\Omega[\rho(r)] = F_{id} + F_{hs} + F_{att} + \sum_\alpha \int dr \rho_\alpha(r) [V_\alpha^{ext}(r) - \mu_\alpha] \quad (2)$$

The systems of interest for these calculations are inhomogeneous fluid systems where the fluid densities are spatially nonuniform due to the presence of the solid interfaces. These surfaces are represented in Eq.2 by the one-body external field, V_α^{ext} . As a result of the inhomogeneity, the total number of fluid particles of each species in the volume of interest will be $N_\alpha = \sum_\alpha \int dr \rho_\alpha(r)$. The grand potential is related to the Helmholtz free energy, F , by $\Omega = F - G$ where $G = \sum_\alpha N_\alpha \mu_\alpha$ is the Gibbs free energy. Clearly, the chemical potential term in Eq.2 is simply G .

In the following sections we systematically define the various approximations to hard sphere, attractive and electrostatic systems.

Section II.A Hard Sphere Fluids

Hard sphere fluids are defined by the interaction potential of two particles a distance r_{12} apart,

$$\begin{aligned} u_{\alpha\beta}(r_{12}) &= \infty & r_{12} < \sigma_{\alpha\beta} \\ u_{\alpha\beta}(r_{12}) &= 0 & \text{otherwise} \end{aligned} \quad (3)$$

which includes an infinitely steep repulsion due to particle overlap. For spheres of unlike size, we average the particle diameters: $\sigma_{\alpha\beta} = 0.5(\sigma_{\alpha\alpha} + \sigma_{\beta\beta})$. For hard sphere systems, the grand potential is

$$\Omega[\rho(r)] = F_{id} + F_{hs} + \sum_{\alpha} \int dr \rho_{\alpha}(r) [V_{\alpha}^{ext}(r) - \mu_{\alpha}] \quad (4)$$

where only ideal and hard sphere terms are included. The ideal gas contribution to the free energy is

$$F_{id} = kT \sum_{\alpha} \int dr \rho_{\alpha}(r) \{ \ln(\Lambda_{\alpha}^3 \rho_{\alpha}(r)) - 1 \} \quad (5)$$

and the hard sphere contribution as derived by Rosenfeld[6] is written in terms of the free energy density, Φ as

$$F_{hs} = \int dr \Phi \{ \bar{\rho}_{\gamma}(r) \} \quad (6)$$

where the nonlocal densities, $\bar{\rho}$, are defined by the weight functions, w , as

$$\bar{\rho}_{\gamma}(r) = \sum_{\alpha} \int dr_2 \rho_{\alpha}(r_2) w_{\alpha}^{(\gamma)}(|r - r_2|). \quad (7)$$

There are six weight function including both scalar and vector terms. These weight functions are

$$\begin{aligned} w_{\alpha}^{(3)} &= \theta(r - R_{\alpha}) \\ w_{\alpha}^{(2)} &= \delta(r - R_{\alpha}) \\ w_{\alpha}^{(1)} &= \frac{\delta(r - R_{\alpha})}{4\pi R_{\alpha}} \\ w_{\alpha}^{(0)} &= \frac{\delta(r - R_{\alpha})}{4\pi R_{\alpha}^2} \\ w_{\alpha}^{(V2)} &= \frac{\vec{r}}{r} \delta(r - R_{\alpha}) \\ w_{\alpha}^{(V1)} &= \frac{1}{4\pi R_{\alpha}} \frac{\vec{r}}{r} \delta(r - R_{\alpha}) \end{aligned} \quad (8)$$

Clearly, these weight functions are related to the fundamental measures of the particles in the fluid. More specifically, integrating over $w^{(3)}$ gives the volume of a particle, integrating over $w^{(2)}$

gives the surface area of a particle, integrating over $w^{(1)}$ gives the radius of the particle, and integrating over $w^{(0)}$ gives unity. The vector terms ($\vec{r}$ indicates a vector) are related to the surface normals of the particles.

The free energy density, Φ , is related to the Percus-Yevik description of hard sphere fluids. It is a sum of scalar and vector contributions, $\Phi = \Phi_s + \Phi_v$ where

$$\begin{aligned}\Phi_s &= -\bar{\rho}_0 \ln(1 - \bar{\rho}_3) + \frac{\bar{\rho}_1 \bar{\rho}_2}{1 - \bar{\rho}_3} + \frac{1}{24\pi} \frac{\bar{\rho}_2^3}{(1 - \bar{\rho}_3)^2} \\ \Phi_v &= -\frac{\bar{\bar{\rho}}_{v1} \cdot \bar{\bar{\rho}}_{v2}}{1 - \bar{\rho}_3} - \frac{1}{8\pi} \frac{\bar{\rho}_2 (\bar{\bar{\rho}}_{v2} \cdot \bar{\bar{\rho}}_{v2})}{(1 - \bar{\rho}_3)^2}\end{aligned}\quad (9)$$

The Euler-Lagrange equation that must be solved for hard sphere fluids is

$$0 = -\mu_\alpha / kT + \ln \rho_\alpha(r) + \frac{1}{kT} \int dr' \left(\sum_\gamma \frac{\partial \Phi(r')}{\partial \bar{\rho}_\gamma(r')} \frac{\delta \bar{\rho}_\gamma(r')}{\delta \rho_\alpha(r)} \right) + \frac{V^{ext}}{kT} \quad (10)$$

From Eq.7 we find that $\frac{\delta \bar{\rho}_\gamma(r')}{\delta \rho_\alpha(r)} = w_\alpha^{(\gamma)}(|r' - r|)$, so the Euler-Lagrange equation becomes

$$0 = -\mu_\alpha / kT + \ln \rho_\alpha(r) + \frac{1}{kT} \int dr' \left(\sum_\gamma \frac{\partial \Phi(r')}{\partial \bar{\rho}_\gamma(r')} w_\alpha^{(\gamma)}(|r' - r|) \right) + \frac{V_\alpha^{ext}(r)}{kT} \quad (11)$$

For reference, the derivatives of Φ in Eq.11 are

$$\begin{aligned}\frac{\partial \Phi}{\partial \bar{\rho}_0} &= -\ln(1 - \bar{\rho}_3) \\ \frac{\partial \Phi}{\partial \bar{\rho}_1} &= \frac{\bar{\rho}_2}{1 - \bar{\rho}_3} \\ \frac{\partial \Phi}{\partial \bar{\rho}_2} &= \frac{\bar{\rho}_1}{1 - \bar{\rho}_3} + \frac{1}{8\pi} \frac{\bar{\rho}_2^2}{(1 - \bar{\rho}_3)^2} - \frac{1}{8\pi} \frac{\bar{\bar{\rho}}_{v2} \cdot \bar{\bar{\rho}}_{v2}}{(1 - \bar{\rho}_3)^2} \\ \frac{\partial \Phi}{\partial \bar{\rho}_3} &= \frac{\bar{\rho}_0}{1 - \bar{\rho}_3} + \frac{\bar{\rho}_1 \bar{\rho}_2}{(1 - \bar{\rho}_3)^2} + \frac{1}{12\pi} \frac{\bar{\rho}_2^3}{(1 - \bar{\rho}_3)^3} - \frac{\bar{\bar{\rho}}_{v1} \cdot \bar{\bar{\rho}}_{v2}}{(1 - \bar{\rho}_3)^2} - \frac{1}{4\pi} \frac{\bar{\rho}_2 (\bar{\bar{\rho}}_{v2} \cdot \bar{\bar{\rho}}_{v2})}{(1 - \bar{\rho}_3)^3} \\ \frac{\partial \Phi}{\partial \bar{\bar{\rho}}_{v1}} &= -\frac{\bar{\bar{\rho}}_{v2}}{1 - \bar{\rho}_3} \\ \frac{\partial \Phi}{\partial \bar{\bar{\rho}}_{v2}} &= -\frac{\bar{\bar{\rho}}_{v1}}{1 - \bar{\rho}_3} - \frac{1}{4\pi} \frac{\bar{\rho}_2 \bar{\bar{\rho}}_{v2}}{(1 - \bar{\rho}_3)^2}\end{aligned}\quad (12)$$

Note that Eq.11 includes an volumetric integral, whose integrand is a nonlinear function of the various $\bar{\rho}$, which are themselves computed from volume integrals. These nested integrals present the main computational difficulty in solving DFT.

Section II.B Lennard-Jones Fluids

Lennard-Jones (LJ) fluids have an interaction potential defined not only by a characteristic size, σ , but also by a characteristic energy, ϵ . The pairwise interaction potential for LJ fluids is

$$u_{\alpha\beta}^{LJ}(r) = 4\epsilon_{\alpha\beta} \left[\left(\frac{\sigma_{\alpha\beta}}{r} \right)^{12} - \left(\frac{\sigma_{\alpha\beta}}{r} \right)^6 \right]. \quad (13)$$

In DFT calculations as well as molecular simulations, it is most often necessary to cut and shift the potential at some finite cutoff. For molecular simulation this reduces the number of pair interactions that must be calculated at each time step. For DFT calculations, this limits the band width of the Jacobian matrix (see section III). The cut and shifted potential is

$$\begin{aligned} u_{\alpha\beta}(r) &= u_{\alpha\beta}^{LJ}(r) - u_{\alpha\beta}^{LJ}(r_{\alpha\beta}^{cut}) & r < r_{\alpha\beta}^{cut} \\ u_{\alpha\beta}(r) &= 0 & \text{otherwise.} \end{aligned} \quad (14)$$

In DFT calculations, a perturbation approach[7] is taken where u is split into attractive and repulsive parts, $u = u^{hs} + u^{att}$. The repulsive part is represented by a hard sphere potential (as defined in Section II.A). The attractive part is

$$\begin{aligned} u_{\alpha\beta}^{att}(r) &= u(r_{\alpha\beta}^{min}) & r \leq r_{\alpha\beta}^{min} \\ u_{\alpha\beta}^{att}(r) &= u(r) & \text{otherwise.} \end{aligned} \quad (15)$$

With this definition of the interaction potential, it is possible to write the grand potential as the sum of hard sphere and attractive contributions given in Eq.2. The hard sphere contribution is then given by the expressions in Section II.A while the attractive contribution to the Helmholtz free energy is

$$F_{att} = \frac{1}{2} \sum_{\alpha} \sum_{\beta} \int dr' \int dr \rho_{\alpha}(r) \rho_{\beta}(r') u_{\alpha\beta}^{att}(|r - r'|) \quad (16)$$

Note that this approximation is called the strict mean field or van der Waals theory of inhomogeneous fluids[8]. The most fundamental expression for F_{att} is

$$F_{att} = \frac{1}{2} \sum_{\alpha} \sum_{\beta} \int_0^1 d\chi \int dr' \int dr \rho_{\alpha\beta}^{(2)}(\phi_{\chi}; r, r') u_{\alpha\beta}^{att}(|r - r'|) \quad (17)$$

where χ is a "switching" parameter that switches between a fully repulsive potential at $\chi=0$, and the perturbed potential $\chi=1$ and $\rho_{\alpha\beta}^{(2)}(\phi_{\chi}; r, r')$ is the full two body distribution function with a given level of perturbation. Assuming no correlations, $\rho_{\alpha\beta}^{(2)}(\phi_{\chi}; r, r')$ is replaced by $\rho_{\alpha}(r) \rho_{\beta}(r')$ to obtain Eq.15. This is a rather crude approximation that is only generally successful when the hard cores are treated with nonlocal (or weighted) density approximations as in Section II.A.

The Euler-Lagrange equation for the LJ system is

$$0 = Eq.11 + \frac{1}{kT} \sum_{\beta} \int dr' [\rho_{\beta}(r') u_{\alpha\beta}^{att}(|r - r'|)]. \quad (18)$$

While this additional term may be significantly longer range than the hard sphere terms in Eq.11, it is only a single integral that is linear in ρ , and therefore much easier to calculate.

Section II.C Electrostatic Systems

For Coulombic systems, it is not a good approximation to cut and shift the long range parts of the potentials, and so a different approach from that outlined in Section II.B is required. The underlying potential of interest assumes that there are point charges with short range repulsive hard cores. The total potential is

$$u_{\alpha\beta}(r) = u_{\alpha\beta}^{hs}(r) + u_{\alpha\beta}^c(r) \quad (19)$$

where the hard sphere term was defined in Eq.3, and Coulombic interactions are

$$u_{\alpha\beta}^c(|r - r'|) = \frac{q_\alpha q_\beta}{4\pi\epsilon|r - r'|}. \quad (20)$$

The grand potential for the fluid portion of this system may be written as

$$\Omega[\rho(r)] = F_{id} + F_{hs} + F_{elec} + \sum_\alpha \int dr \rho_\alpha(r) [V_\alpha^{ext,hs}(r) + V_\alpha^{ext,c}(r) - \mu_\alpha] \quad (21)$$

where the hard core and Coulombic contributions of the external field are included separately. In the strict mean field approximation, the electrostatic contribution would be

$$F_{elec} = \frac{1}{2} \sum_\alpha \sum_\beta \int dr \int dr' \rho_\alpha(r) \rho_\beta(r') \frac{q_\alpha q_\beta}{4\pi\epsilon|r - r'|}. \quad (22)$$

However, taking the definition of the electrostatic potential,

$$\phi(r) = \sum_\alpha \left(V_\alpha^{ext,c} + \int dr' \frac{q_\alpha \rho_\alpha(r')}{4\pi\epsilon|r - r'|} \right) \quad (23)$$

the fluid part of the grand potential may be written as[8,9]

$$\Omega[\rho(r)] = F_{id} + F_{hs} + \frac{1}{2} \int dr' \phi(r') \sum_\alpha q_\alpha \rho_\alpha(r') + \sum_\alpha \int dr \rho_\alpha(r) [V_\alpha^{ext,hs}(r) - \mu_\alpha], \quad (24)$$

and the Euler-Lagrange equation becomes

$$0 = Eq.11 + \frac{q_\alpha \phi(r)}{kT}. \quad (25)$$

Introducing the new variable (the electrostatic potential) requires that an additional equation must be solved. The appropriate equation is Poisson's equation which may be written in differential form as

$$\nabla^2 \phi(r) = -\sum_{\alpha} q_{\alpha} \rho_{\alpha}(r) / \epsilon. \quad (26)$$

Finally, it is known that the strict mean field approach, which neglects correlations, will not reproduce certain important features of inhomogeneous electrolyte systems. One notable example is the attractive force between like charged plates in 2:2 electrolytes. In an effort to overcome this deficiency, the exact solution to the uniform MSA electrolyte is used. Specifically, the difference between the short range direct correlation function in a fluid of charged hard spheres and that of a neutral hard sphere mixture at the same density, Δc is used to estimate the correction. The expression for Δc [10] is

$$\Delta c_{\alpha\beta}(r) = -q_{\alpha} q_{\beta} \left[\frac{2B}{\sigma_{\alpha\beta}} - \frac{1}{r} - \left(\frac{B}{\sigma_{\alpha\beta}} \right)^2 r \right] \frac{e^2}{4\pi k T \epsilon \epsilon_0 \sigma} \quad (27)$$

where

$$B = (\kappa + 1 - \sqrt{1 + 2\kappa}) / \kappa \quad (28)$$

and

$$\kappa = \sqrt{\frac{e^2}{k T \epsilon \epsilon_0 \sigma} \sum_{\alpha} \rho_{\alpha}^b q_{\alpha}^2}. \quad (29)$$

With Δc , the grand potential becomes

$$0 = Eq. 23 - \frac{1}{2} k T \sum_{\alpha} \sum_{\beta} \int dr' \rho_{\alpha}(r) \rho_{\beta}(r') \Delta c_{\alpha\beta}(|r - r'|). \quad (30)$$

And the Euler-Lagrange equation is

$$0 = Eq. 11 + \frac{q_{\alpha} \phi(r)}{k T} - \sum_{\beta} \int dr' \rho_{\beta}(r') \Delta c_{\alpha\beta}(|r - r'|). \quad (31)$$

Calculations that either include or neglect the Δc term may be performed with TRAMONTO by toggling the `Sten_type[3]` logical described in the users guide (section VI).

Section II.D Ideal Gases and the Poisson-Boltzmann Electrolytes

In addition to the fluids presented above, both ideal gas fluids and Poisson-Boltzmann (PB) electrolytes may be treated. In the case of PB fluids, the solvent is a continuum, and the charges are point charges. In both PB electrolytes and ideal gases, $F_{hs}=0$ because the particles have no volume. In addition, $\Delta c=0$ in this limit. Thus, the Euler-Lagrange equation becomes

$$0 = -\mu_{\alpha} / k T + \ln \rho_{\alpha}(r) + \frac{V_{\alpha}^{ext}(r)}{k T}, \quad (32)$$

and the Euler-Lagrange equation for the PB electrolytes is

$$0 = -\mu_{\alpha} / k T + \ln \rho_{\alpha}(r) + \frac{q_{\alpha} \phi(r)}{k T}. \quad (33)$$

Recognizing that for an ideal solution, $\mu_\alpha / kT = \ln \rho_\alpha^b$, and substituting this expression into Eq.33 yields the familiar PB equation

$$\rho_\alpha(r) = \rho_\alpha^b(r) e^{-q_\alpha \phi(r) / kT}. \quad (34)$$

Section II.E Calculating Adsorptions, Surface Free Energies, and Solvation Forces

In this section, the principle outputs of the code and the methods by which they are calculated are discussed. One way to check that the code is running correctly is to check one or more of the sum rules that must apply for the system[5]. Several useful sum rules are presented in this and the next section.

The principle outputs of the DFT code are the density distributions, the total adsorption, the excess adsorption, the grand free energy, the surface free energy (or solvation potential), and the solvation force. For charged systems, the net charge in the fluid is also calculated. In order to calculate these system properties as well as test the sum rules effectively, it is necessary to define a Gibbs dividing surface. In all that follows, the Gibbs dividing surfaces are defined to correspond to the surfaces of the particles or walls that contribute to the external field. So, for hard wall-fluid interactions, there will be an exclusion region where the fluid densities must be strictly zero, but where bulk contributions to the integrals must be included.

The total adsorption in a given system is

$$\Gamma_\alpha = \int dr \rho_\alpha(r) \quad (35)$$

while the excess adsorption is

$$\Gamma_\alpha^{\text{ex}} = \int dr [\rho_\alpha(r) - \rho_\alpha^b] \quad (36)$$

The grand potential is calculated directly from Eq.2,4, or 24 depending on the type of system. The surface excess free energy, Ω^s , is

$$\Omega^s = \Omega - \Omega^b \quad (37)$$

where $\Omega^b = -pV$ is the grand free energy of a bulk system where p is the pressure. An alternative to calculating pV is to replace $\rho(r)$ by ρ^b in all the integrals in Eq.2,4, and 24. This is the approach used in TRAMONTO.

For surfaces of arbitrary geometry, the solvation force is vector, $f^s = (f_x^s, f_y^s, f_z^s)$. When there is only one surface in the system, the solvation force in each direction must be zero for hydrostatic equilibrium. In contrast when there is more than one surface, the solvation force may be nonzero. For a system with N surfaces, the solvation force in the i th dimension on the n th surface is related to the total surface free energy and the position of the surface, $R = (R_x, R_y, R_z)$ by the sum rule

$$f_i^{s,(n)} = -\frac{\partial \Omega^s}{\partial R_i^{(n)}} \quad (38)$$

For neutral systems, A derivation of the force begins with the functional derivative

$$\frac{\delta \Omega}{\delta(\mu_\alpha - V_\alpha^{ext}(r))} = -\rho_\alpha(r) \quad (39)$$

For an arbitrary number of surfaces locate at $\{R\}$, the external field that a particle of species α feels at a given location in the fluid, r is

$$V_\alpha^{ext}(r, \{R\}) = \phi_\alpha^{(1)}(r, R^{(1)}) + \phi_\alpha^{(2)}(r, R^{(2)}) + \dots + \phi_\alpha^{(N)}(r, R^{(N)}) \quad (40)$$

Keeping the chemical potential constant, the solvation force, f , is

$$\begin{aligned} f_i^{s,(n)} &= -\frac{\partial \Omega}{\partial R_i^{(n)}} = -\sum_\alpha \int dr \frac{\delta \Omega}{\delta V_\alpha^{ext}} \frac{\partial V_\alpha^{ext}}{\partial R_i^{(n)}} = -\sum_\alpha \int dr \rho_\alpha(r) \frac{\partial V_\alpha^{ext}}{\partial R_i^{(n)}} \\ &= -\sum_\alpha \int dr \rho(r) \frac{\partial \phi_\alpha^{(n)}}{\partial x_i} \frac{\partial x_i}{\partial R_i^{(n)}} = -\sum_\alpha \int dr \rho(r) \nabla_i \phi_\alpha^{(n)}. \end{aligned} \quad (41)$$

If, the wall-fluid interactions are of the Lennard-Jones type, Eq.41 is used to calculate the solvation force. If the wall-fluid interactions are hard-sphere/hard-wall, the external field is a step function, and its gradient is a delta function. As a result, the solvation force is the integral of the contact densities on the boundary, R_s of the surface of interest.

$$f_i^{s,(n)} = kT \sum_\alpha \int_S dR_s [\rho_\alpha(r = R_s)] \bar{n}_i \quad (42)$$

If we confined the discussion to two surfaces of arbitrary shape separated by the distance L , the solvation force acts along the line of centers of the two surfaces, and the sum rule in Eq.38 may be written

$$f_L^{s,(n)} = -\frac{\partial \Omega^s}{\partial L}. \quad (43)$$

Since the $\frac{\partial \Omega^s}{\partial L} = 0$ when $L \rightarrow \infty$, it is also possible to write the inverse of Eq.3, that is

$$\Omega^s(L) - \Omega^s(L = \infty) = - \int_{L=\infty}^L f^s(L) dL \quad (44)$$

The free energy difference in Eq.44 has been called the solvation potential, but is also strictly equal to the potential of mean force (PMF) for surfaces interacting in open systems[11]. Hence this PMF once calculated from DFT may be used in simulations of colloidal systems.

Finally, if there are coulombic forces in the system (with underlying hard cores), the solvation force on the n th surface is

$$f_i^{s,(n)} = kT \sum_{\alpha} \int_S dR_s [\rho_{\alpha}(r = R_s)] \bar{n}_i + f_i^{s,(n),C} \quad (45)$$

where $f_i^{s,(n),C}$ is the coulombic contribution to the force. The coulombic contribution is derived similarly to the neutral contributions. The derivation begins with the functional derivative

$$\frac{\delta \Omega}{\delta(\varphi(r))} = - \sum_{\alpha} q_{\alpha} \rho_{\alpha}(r) \quad (46)$$

The coulombic contributions are then derived as in Eq.40 to be

$$\begin{aligned} f_i^{s,(n),C} &= - \frac{\delta \Omega}{\delta R_i^{(n)}} = - \int dr \frac{\delta \Omega}{\delta \varphi(r)} \frac{\partial \varphi(r)}{\partial R_i^{(n)}} = - \int dr \left(\sum_{\alpha} q_{\alpha} \rho_{\alpha}(r) \right) \frac{\partial \varphi}{\partial R_i^{(n)}} \\ &= - \int dr \left(\sum_{\alpha} q_{\alpha} \rho_{\alpha}(r) \right) \frac{\partial \varphi}{\partial x_i} \frac{\partial x_i}{\partial R_i^{(n)}} = - \int dr \left(\sum_{\alpha} q_{\alpha} \rho_{\alpha}(r) \right) \nabla_i \varphi. \end{aligned} \quad (47)$$

By substituting $\epsilon \epsilon_0 \nabla^2 \varphi$ into Eq.47, performing appropriate matrix manipulations, and applying Gauss' theorem, it is reasonably straightforward to rewrite Eq.47 in terms of a surface integral. To total force on a given surface is then

$$f_i^{s,(n)} = kT \sum_{\alpha} \int_S dR_s [\rho_{\alpha}(r = R_s)] \bar{n}_i + \int_S \epsilon \epsilon_0 \left[\nabla_i \varphi \nabla_j \varphi - \frac{1}{2} \nabla \varphi \cdot \nabla \varphi \right] \cdot \bar{n} dR_s \quad (48)$$

When the force, calculated from Eq.48, is integrated via Eq.44 and compared with the surface free energy based on Eq.24, the sum rule is found to be incorrect. The discrepancy can ultimately be attributed to previously neglecting surface terms in the definition of the free energy. For charged systems, the total surface free energy depends not only on the state of the fluid, but also on the state of the surface. Thus, the total free energy is[12]

$$\begin{aligned} \Omega[\rho(r)] &= F_{id} + F_{hs} + \frac{1}{2} \int dr' \varphi(r') \sum_{\alpha} q_{\alpha} \rho_{\alpha}(r') + \sum_{\alpha} \int dr \rho_{\alpha}(r) [V_{\alpha}^{ext,hs}(r) - \mu_{\alpha}] \\ &\quad + \sum_{\xi} \int_{S^{\xi}} dr_s q_s(r_s) \varphi(r_s) + \frac{\epsilon \epsilon_0}{2} \sum_{\xi} \int_{V^{\xi}} dr_s (\nabla \varphi(r_s))^2 \end{aligned} \quad (49)$$

where the ξ sum indicates a sum over surfaces, S^{ξ} indicates a surface integral, and V^{ξ} indicates volume integrals confined within the surfaces.

Section II.F Sum Rules

The sum rule for the solvation force (Eq.38) is not the only sum rule that must hold for a correct numerical solution to the DFT. Others[5]are:

$$\begin{aligned}\frac{\partial(\Omega^s / A)}{\partial T} &= \frac{-S^{ex}}{A} \\ \frac{\partial(\Omega^s / A)}{\partial \mu_\alpha} &= -\Gamma_\alpha^{ex}\end{aligned}\tag{50}$$

where S^{ex} is the surface excess entropy. For some of the sum rules the two sides of the equation can be calculated independently, The force sum rule (Eq.38), and the adsorption sum rule (Eq.50b) are examples. In other cases, such as the surface excess entropy case (Eq.50a), the sum rule may be used to calculate a relevant thermophysical parameter that is not easy to calculate directly from the density distributions.

In this section, we present the results of sum rule tests for two examples. The first is two charged flat plates in electrolyte solutions. The other involves two charged cylinders in the same electrolyte solutions.

Section II.F.1 Charged Parallel Plates.

The charged planar system used to test sum rules consists of two planar surfaces immersed in an electrolyte solution. Each face of both surfaces has a surface charge density of $q\sigma^2/e=0.2$. The electrolyte is modeled in two ways.

The first model electrolyte (the 3CM) has 3 components of identical size; a neutral solvent ($\rho\sigma^3=0.6$), and a 1:1 electrolyte ($\rho\sigma^3=0.005$ for each species). The second model electrolyte consists of point charge ions in a continuum solvent. This is the Poisson-Boltzmann (PB) approximation. The geometry of the calculation is shown in Fig.II.1. The boundary conditions on the domain are set equal to the bulk densities, and the electric potential is zero at the boundaries.

The results (force and adsorption) will be presented as a function of the separation of the surfaces, h . As the surface separation increases, new mesh points are added so that the amount of fluid on the outside of the surfaces is held constant.

Fig.II.1 A schematic of the parallel planar calculation geometry. The fluid regions external to the surfaces are included explicitly in the calculation. The parallel plates are of infinite extent, so the calculation is 1-dimensional with densities varying only perpendicular to the surfaces.

Fig.II.2 shows density profiles as well as electrostatic potential profiles for both the 3CM and PB model electrolytes when the surface separation between the plates is 9σ (about $38\text{\AA}$) and the dimensionless temperature is $4\pi kT\epsilon\epsilon_0\sigma/e^2 = 0.7$. Clearly, the 3CM displays more complex structure near the surfaces. This structure is reflected in the solvation force and free energy plots shown in Fig.II.3.

The sum rule in Eq.38 is tested graphically in Fig.II.3. In both cases, the symbols are the result of direct calculation of the parameter in the plot. The lines show the result from either taking the derivative of the free energy (in the force plot) or integrating up the force (the free energy plot). There is very good agreement between direct calculation and the sum rule test over the entire range of surface separations for the PB electrolyte.

For the 3CM electrolyte, agreement is good for most surface separations. The exception in the case of the force (left plot in Fig.II.3) is found in the vicinity of $h/\sigma = 1$. At this separation, there is only one mesh point between the surfaces where fluid may be found. For this reason the force calculation is numerically troublesome. One way to improve the force curve in this region would be to perform calculations with smaller grid spacing thus allowing for more surface separations in this region. The discrepancy between the sum rule and the direct calculation of the free energy exists for all $h/\sigma \leq 1$. Again this arises from the difficulty in calculating the force when $h/\sigma = 1$. In this case, since the free energy at a given h is found by integrating up the force, this numerical integral cannot be correct whenever the $h/\sigma = 1$ point is included in the integral.

It is possible to estimate the force at $h/\sigma = 1$ by using the sum rule. Start with the limiting value of Ω^s at $h/\sigma = 1$ taken from below. Since the slope of the free energy is discontinuous at $h/\sigma = 1$, the, the force at $h/\sigma = 1$ can be estimated from a forward finite difference equation

such as $f_{h=1\sigma}^s = [\Omega_{h=1\sigma}^s - \Omega_{h=1\sigma+\Delta h}^s] / \Delta h$. For the case in Fig.II.3, the force at $h/\sigma = 1$ is estimated to be $f^s \sigma^3 / kT = 29.4$.

Since the free energy curve in Fig.II.3 is the interaction potential between the surfaces, it is natural to consider where the equilibrium separation of the surfaces will be found. The global minimum of the interaction potential gives the equilibrium spacing of the surfaces. For the case in Fig.II.3, the equilibrium separation is $h = \infty$ for both the 3CM and the PB electrolytes. However, the 3CM case has several local minima at small surface separations. As the system parameters are varied (temperature, solution ionic strength, bulk pressure, etc.) any of these minima may become the global minima. Transitions between surface separations that correspond to the different minima are called solvation transitions[13].

Fig.II.3 Testing the sum rule for the solvation force in charged systems. The plot on the left shows the force, the plot on the right shows the free energy shifted by its infinite separation value (ie. It is the potential of mean force). The insets show results for a Poisson-Boltzmann electrolyte while the main figure shows the results for a three component model. In all cases, the points show the results of direct calculation of the variable from the density profiles while the lines show the sum rule result.

We turn now to the adsorption sum rule in Eq.50. Note that in the electrolyte model defined in section II.C, the chemical potential of each species in the bulk is controlled only by the total number density of the mixture provided that all the particles have the same size. Thus, the chemical potentials of counter-ions and co-ions are identical. Keeping the solvent chemical potential constant while varying the ion chemical potentials leads to the sum rule

$$\frac{\partial(\Omega^s / A)}{\partial \mu_{\pm}} = -(\Gamma_{+}^{ex} + \Gamma_{-}^{ex}) \quad (51)$$

Figure II.4 shows the direct adsorption calculation as compared with the chemical potential derivative of the surface free energy as a function of the separation of the plates. There is a systematic discrepancy between the direct calculations and the sum rule for both PB and 3CM models. This discrepancy is reduced when the mesh is refined by a factor of two near the surfaces. The results in this section demonstrate that the sum rules for the electrolytes near planar surfaces are satisfied. Performing the sum rule checks for other fluids yields similar results.

Section II.F.2 Charged Cylindrical Surfaces

To demonstrate that the sum rules must be obeyed for complex surface geometries as well as the planar surface case, we now turn to the interaction of two parallel cylinders of infinite length in an electrolyte solution.

The surface charge per unit area was $q\sigma^2/e=0.17$, and the diameter of the cylinder was 3σ . A sketch of the computational domain is shown in Fig.II.6. Since the cylinders are of infinite length, the fluid density does not vary in the axial direction, and a 2D calculation may be performed. As indicated in Fig.II.6, only one quadrant of the domain is needed so long as reflective boundaries are used on the top and right hand side of the domain. This is an example of how symmetries in a system may be exploited with DFT. In a molecular simulation, the entire domain would be needed for the calculation.

The force and potential of mean force are shown for the charged parallel cylinders in Fig.II.7. Again, the symbols are the direct calculations of the force and free energy while the lines show the application of the sum rule to numerical derivatives or integrals. As with the flat plate case,

the sum rule works almost perfectly for the point charge electrolyte (insets). However, there are now some discrepancies with the three component model.

Consider first the potential of mean force (right). As with the flat plate case, there is good agreement with the sum rule except for surface separations, $D < 1\sigma$. This discrepancy again arises from the difficulty in calculating the force when the surface separation (at any point on the surface) is exactly 1σ . Following the results from the planar surfaces, it is apparent that the direct free energy calculation should be used for these points.

Turning to the force calculation (left), there are quite large discrepancies between the direct calculation and the numerical derivative of the free energy. These discrepancies come from the scatter in the free energy data. This scatter arises from the underlying mesh used for the calculations. The mesh density for 3CM calculations was 0.1σ in both x and y . However, when the surfaces are separated by adding one row of mesh points to the right side of the domain in Fig.II.6, the center of the cylinder is shifted to the left by 0.05σ . Since this distance is smaller than the mesh density, there can be slight discrepancies in the surface shape when the cylinders are moved. This leads to the scatter in the free energy which obviously leads to inaccurate estimation of the force by numerical integration. These results highlight that the sum rules should be applied with care, and that mesh densities should be chosen so that a given surface has a consistent shape when moved. Fig.II.7 also demonstrates that the fundamental equations underlying force and free energy calculations as implemented in TRAMONTO are correct regardless of surface geometry.

Fig.II.7 Testing the sum rule for the solvation force between charged parallel cylinders. The plot on the left shows the force perunit length, while the plot on the right shows the potential of mean force. The main plots show results for the 3CM while the insets show results from the PB model.

Section III: Numerical Methods

In this section, the basic numerical approach used in TRAMONTO is outlined. Particular attention is paid to the complex nature of the integral equations in the DFT. In order to efficiently solve these equations, a variety of improvements to the algorithms have been implemented. These improved algorithms and the performance of the code are detailed in Section IV.

Section III.A Meshing and Quadrature

The algorithms for solving the Euler-Lagrange equations have been designed to for efficient calculation of the hard sphere terms, which are by far the most complicated and expensive part of a nonlocal DFT calculation. These terms include an integral over both δ and θ (step) functions (Eq.11) multiplied by a complicated nonlinear algebraic expression (Eq.12) that depends on the non-local densities, $\bar{\rho}$ (Eq.7). These nonlocal densities are themselves integrals over the δ and θ functions multiplied by the local density. The self-similarity of the equations at these different levels is ultimately very helpful for implementing efficient algorithms.

A collocation approach was chosen to discretize the problem. At first look, nonuniform grids would appear attractive for this problem; both to represent complex geometries and to put more approximating power near the walls where large density gradients are seen. However, the implementation of nonuniform meshes for the Euler-Lagrange equations is not straightforward. The primary complication for a nonuniform mesh is that it would be necessary to store a different integration quadrature for every point in the mesh. If there are 10^3 quadrature points in a 3D integral, and 10^6 mesh points in a desired system, the memory required to store the quadratures alone would be a prohibitive 12GB.

Given the expected memory requirements of a nonuniform mesh, TRAMONTO was initially implemented on a uniform cartesian mesh so that only a few quadrature stencils would need to be stored, and could then be translated around the mesh. A drawback of this is that curved walls are represented by a staircase boundary. However, since the dimensions of the staircase are much smaller than the dimensions of a solvent molecule, this coarsening is not detrimental to the final solution. We have also required that the domain be rectangular. This simplifies traversing of the mesh when doing the integrations. In section IV, we discuss one way to preserve the efficiency of a uniform mesh approach while allowing variable mesh spacing for greater accuracy near walls.

Given the weight functions in Eq.8, only two quadrature stencils are needed (per component, α) to describe all the hard sphere weight functions. These will be referred to as the *Delta* (δ) and *Theta* (θ) stencils. The prefactors of the weight functions in Eq.8 are denoted $C_\alpha^{(\gamma)}$ ($\gamma=0-3$) or $\tilde{C}_\alpha^{(\gamma)}$ ($\gamma=V1$ or $V2$). Integrals over the weight functions are replaced by summations over a finite set (N_{sten} total) of quadrature points where the function at the stencil points (as defined as offsets from some origin) are multiplied by the appropriate weights ($\omega_\alpha^{(\delta)}$ or $\omega_\alpha^{(\theta)}$).

$$\begin{aligned} \int w_\alpha^{(\gamma<3)} f(r) dr &\approx \sum_{i=1}^{N_{sten}} C_\alpha^{(\gamma)} \omega_\alpha^{\delta,i} f^i & \int w_\alpha^{(\gamma=V1,V2)} f(r) dr &\approx \sum_{i=1}^{N_{sten}} \tilde{C}_\alpha^{(\gamma)} \omega_\alpha^{\delta,i} f^i \\ \int w_\alpha^{(3)} f(r) dr &\approx \sum_{i=1}^{N_{sten}} C_\alpha^{(3)} \omega_\alpha^{\theta,i} f^i \end{aligned} \quad (52)$$

These weight function stencils are pre-calculated before beginning Newton iterations in TRAMONTO. They are stored as arrays of offsets (in each dimension), and weights. When the integrals are performed, the origin of the integral moves around the mesh, but the ijk offsets on the uniform mesh can be used along with simple integer operations to find the quadrature points for the integrals.

Section III.B. Solution Algorithm

The result of the collocation is a finite set of unknowns, the densities at the nodes, and the same number of residual equations, the Euler-Lagrange equations for each component at each grid point. For electrolyte systems, the electric potential at each grid point is calculated by solving Poisson's equation using Galerkin finite elements with linear basis function.

The solution to the resulting system of equations is found iteratively with Newton's method. This requires solving the matrix problem:

$$J_{ij}\Delta_j = -R_i \quad (53)$$

where $\Delta_j = \chi_j^{(k+1)} - \chi_j^{(k)}$ is the difference between the solution vector at the $k+1^{\text{th}}$ Newton iteration and the k^{th} iteration. The solution vector, χ_i is composed of both densities, ρ , and for charged systems, the electrostatic potential, ϕ . The Jacobian matrix, J_{ij} , is $J_{ij} = \frac{\delta R_i}{\delta \chi_j}$. When

$\chi = \rho$, the Jacobian is

$$J_{ij} = \frac{\delta^2 \Omega}{\delta \rho_i \delta \rho_j} = c^{(2)}(\vec{r}_i, \vec{r}_j) + \frac{\delta_{ij}}{\rho_i} \quad (54)$$

where $c^{(2)}$ is the two-body inhomogeneous direct correlation function. More explicitly, the density contributions to the Jacobian are

$$J_{ij} = \frac{\delta_{ij}}{\rho_i} + \beta u_{ij}^{\text{att}} - \Delta c_{ij} + \beta \int d\mathbf{r}_k \sum_{\gamma} \sum_{\epsilon} \frac{\partial^2 \Phi}{\partial \rho_{\gamma} \partial \rho_{\epsilon}} w_{\alpha}^{(\gamma)}(i, k) w_{\beta}^{(\epsilon)}(j, k) \quad (55)$$

where the u^{att} term applies to LJ systems and the Δc term may be used in electrostatic calculations. TRAMONTO is not currently configured to perform calculations of charged LJ systems. When, $\chi = \phi$, the Residual and Jacobian entries are

$$\begin{aligned} R_i &= \int \left(\nabla \phi_i \cdot \nabla \phi - \phi_i \sum_{\alpha} q_{\alpha} \rho_{\alpha} \right) d\mathbf{r} \\ J_{i\phi} &= \int (\nabla \phi_i \cdot \nabla \phi_j) d\mathbf{r} \\ J_{i\rho_{\alpha}} &= \int (-\phi_i \phi_j q_{\alpha}) d\mathbf{r} \end{aligned} \quad (56)$$

The matrix problem is solved in parallel at every Newton iteration using the Aztec linear solver library. While this library includes a variety of sophisticated preconditioners, a simple Jacobi scaling and GMRES solver (no preconditioning) works well for most of these systems.

The integrals over Poisson's equation multiplied by the linear basis functions, ϕ , in Eq.56 can be performed analytically on the uniform meshes used for these calculations. As a result, the implementation of Poisson's equation is straightforward. In 1D, for a three-component problem, it leads to 12 entries per row in the Jacobian. In 2D, there are 36 entries per row, and in 3D, there are 104 entries per row. In contrast, the integral in Eq.55 cannot be done analytically. Furthermore, the corresponding integral in the residual must be done to rather high precision in order to guarantee good accuracy in the solution vector. Using the same quadrature for the Jacobian as is used in the residual will ultimately lead to a nonzero Jacobian entry in every position of a given row that is within the cutoff dictated by the weight function of interest. For three-component systems, and a mesh size of 0.1σ , there will be $O(60)$ nonzero entries in 1D, $O(900)$ nonzero entries in 2D, and $O(12000)$ nonzero entries per row in 3D. Clearly, the Euler-Lagrange equations will result in far denser Jacobians than Poisson's equation (or other partial differential equations) alone.

Returning to the Euler-Lagrange Jacobian, there are two challenges to overcome for 2D and 3D calculations. They are: (1) the complexity involved in filling the Jacobian and (2) the memory required to store the Jacobian. To demonstrate the complexity involved in filling the Jacobian, consider Eq.55. The hard sphere term for J_{ij} is an integral

over a complex function $(\sum_{\gamma} \sum_{\epsilon} \frac{\partial^2 \Phi}{\partial \bar{\rho}_{\gamma} \partial \bar{\rho}_{\epsilon}})$

in the region of overlap $(w_{\alpha}^{(\gamma)}(i,k)w_{\beta}^{(\epsilon)}(k,j))$ of two quadrature stencil functions that start from the i th and the j th node respectively (see the 2D schematic in Fig.III.1). Note that the indices ij on the Jacobian refer to the indices of *unknowns* in the calculation. An unknown number is identified both by the node point in the mesh and by the particular component of interest. Thus, give i,j , one immediately knows the α,β indices on the weight functions. Values of the second derivatives of Φ are shown in Eq.57.

Fig.III.1: A schematic of the hard sphere part of the Jacobian fill routine.

$$\begin{aligned}
\frac{\partial}{\partial \bar{\rho}_0} \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{(\gamma)} \right) &= \frac{1}{1 - \bar{\rho}_3} C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} \\
\frac{\partial}{\partial \bar{\rho}_1} \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{(\gamma)} \right) &= \frac{1}{1 - \bar{\rho}_3} C_{\alpha}^{(2)} \omega_{\alpha}^{(\delta)} + \frac{\bar{\rho}_2}{(1 - \bar{\rho}_3)^2} C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} \\
\frac{\partial}{\partial \bar{\rho}_2} \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{(\gamma)} \right) &= \frac{1}{1 - \bar{\rho}_3} C_{\alpha}^{(1)} \omega_{\alpha}^{(\delta)} + \frac{1}{4\pi} \frac{\bar{\rho}_2}{(1 - \bar{\rho}_3)^2} C_{\alpha}^{(2)} \omega_{\alpha}^{(\delta)} - \frac{1}{4\pi} \frac{\bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^2} \cdot \bar{C}_{\alpha}^{(v2)} \omega_{\alpha}^{(\delta)} \\
&\quad + \left[\frac{\bar{\rho}_1}{(1 - \bar{\rho}_3)^2} + \frac{1}{4\pi} \frac{\bar{\rho}_2^2}{(1 - \bar{\rho}_3)^3} - \frac{1}{4\pi} \frac{(\bar{\rho}_{v2} \cdot \bar{\rho}_{v2})}{(1 - \bar{\rho}_3)^3} \right] C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} \\
\frac{\partial}{\partial \bar{\rho}_3} \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{(\gamma)} \right) &= \frac{1}{(1 - \bar{\rho}_3)^2} C_{\alpha}^{(0)} \omega_{\alpha}^{(\delta)} + \frac{\bar{\rho}_2}{(1 - \bar{\rho}_3)^2} C_{\alpha}^{(1)} \omega_{\alpha}^{(\delta)} + \\
&\quad + \left[\frac{\bar{\rho}_1}{(1 - \bar{\rho}_3)^2} + \frac{1}{4\pi} \frac{\bar{\rho}_2^2}{(1 - \bar{\rho}_3)^3} - \frac{1}{4\pi} \frac{\bar{\rho}_{v2} \cdot \bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^3} \right] C_{\alpha}^{(2)} \omega_{\alpha}^{(\delta)} \\
&\quad + \left[\frac{\bar{\rho}_0}{(1 - \bar{\rho}_3)^2} + 2 \frac{\bar{\rho}_1 \bar{\rho}_2}{(1 - \bar{\rho}_3)^3} + \frac{1}{4\pi} \frac{\bar{\rho}_2^3}{(1 - \bar{\rho}_3)^4} \right] C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} \\
&\quad + \left[-2 \frac{\bar{\rho}_{v1} \cdot \bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^3} - \frac{3}{4\pi} \frac{\bar{\rho}_2 (\bar{\rho}_{v2} \cdot \bar{\rho}_{v2})}{(1 - \bar{\rho}_3)^4} \right] C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} \\
&\quad - \frac{\bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^2} \cdot \bar{C}_{\alpha}^{(v1)} \omega_{\alpha}^{(\delta)} + \left[-\frac{\bar{\rho}_{v1}}{(1 - \bar{\rho}_3)^2} - \frac{1}{2\pi} \frac{\bar{\rho}_2 \bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^3} \right] \cdot \bar{C}_{\alpha}^{(v2)} \omega_{\alpha}^{(\delta)} \\
\frac{\partial}{\partial \bar{\rho}_{v1}} \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{(\gamma)} \right) &= -\frac{\bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^2} C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} - \frac{1}{1 - \bar{\rho}_3} \bar{C}_{\alpha}^{(v2)} \omega_{\alpha}^{(\delta)} \\
\frac{\partial}{\partial \bar{\rho}_{v2}} \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{(\gamma)} \right) &= -\frac{1}{4\pi} \frac{\bar{\rho}_{v2}}{(1 - \bar{\rho}_3)^2} C_{\alpha}^{(2)} \omega_{\alpha}^{(\delta)} + \left[-\frac{\bar{\rho}_{v1}}{(1 - \bar{\rho}_3)^2} - \frac{1}{2\pi} \frac{\bar{\rho}_2 (\bar{\rho}_{v2})}{(1 - \bar{\rho}_3)^3} \right] C_{\alpha}^{(3)} \omega_{\alpha}^{(\theta)} \\
&\quad - \frac{1}{1 - \bar{\rho}_3} \bar{C}_{\alpha}^{(v1)} \omega_{\alpha}^{(\delta)} - \frac{1}{4\pi} \frac{\bar{\rho}_2}{(1 - \bar{\rho}_3)^2} \bar{C}_{\alpha}^{(v2)} \omega_{\alpha}^{(\delta)}
\end{aligned} \tag{57}$$

Section III.C Algorithms and Scaling

The basic algorithm for the Newton solve of the coupled nonlinear equations is shown in Fig.III.2, and a more detailed description of the Residual/Jacobian load algorithm for the density unknowns is given in Figs.III.3 and III.4. Before the Jacobian is filled, the nonlocal densities, $\bar{\rho}_\gamma$, and the values of $\partial\Phi/\partial\bar{\rho}_\gamma$ are calculated on the entire mesh. Precalculating these parameters requires storing $[8+4*(Ndim)]$ doubles per node in serial where $Ndim$ is the dimensionality of the problem of interest. In parallel (see below), the nonlocal densities and hard sphere free energy derivatives are calculated at every point on an extended local domain. For regions of the domain where the presence of surfaces completely precludes the solvent as indicated by a switch (*Zero_density_TF*), the equation $x=0$ is solved. While these trivial equations could be removed from the system of equations entirely, the result would be a reduced domain with irregular boundaries. Indexing in this domain would not be as convenient as on the regular cartesian mesh.

As was indicated in Fig.III.1, each entry in the Jacobian, J_{ij} involves integrating the overlap region of two weight functions, $w^{(n)}(i,k)$ and $w^{(e)}(j,k)$. A very inefficient way to perform this operation would be to loop through the two stencils from the mesh points corresponding to the i th and j th unknowns ($inode$ and $jnode$) searching for overlaps. With this approach there would be much wasted effort identifying stencil points based on $inode$ that don't appear in the $jnode$ stencil. The scaling to fill one row of the Jacobian would go like $N_{comp} N_{sten}^3$.

An alternative approach is outlined in Figs.III.3 and III.4, and has been implemented in TRAMONTO. Instead of filling one Jacobian entry at a time, the Jacobian is filled by rows. The procedure begins at the i th unknown (i th row in the Jacobian) which corresponds to the α component located at $inode$. The outer loop of the Jacobian is over one of the stencils (first *Delta* and then *Theta*) associated with the α component. Each of the $isten$ nodes reached by the stencil is necessarily in the overlap region of $inode$ and any other $jnode$ that is in the stencil list that starts from $isten$. So, from each of the $isten$ nodes, two nested loops are performed. The outer loop goes over components β . The inner loop goes over a second set (both *Delta* and *Theta*) of stencils (denoted $jsten$) corresponding to β that have their origin at $isten$. Each $jsten$ node for a given β corresponds to a different column in the Jacobian. A given Jacobian entry is only completely filled when terms for all possible combinations of weight functions and β components have been calculated. For this algorithm, the scaling to fill one row of the Jacobian is $N_{comp} N_{sten}^2$.

With the algorithm outlined in Figs.III.3 and III.4, the scaling for the entire Jacobian fill will go like $N_{nodes} N_{comp}^2 N_{sten}^2$. In terms of the number of mesh points in the domain, the scaling is linear (N_{nodes}), but in terms of the mesh spacing, Δx , the scaling is much worse because $N_{sten}, N_{nodes} \propto 1/\Delta x$ (1D), $N_{sten}, N_{nodes} \propto (1/\Delta x)^2$ (2D), and $N_{sten}, N_{nodes} \propto (1/\Delta x)^3$ (3D). Thus the total scaling is expected to go like $(1/\Delta x)^2$ in 1D, $(1/\Delta x)^4$ in 2D, and $(1/\Delta x)^6$ in 3D assuming that the grid is refined in all dimensions simultaneously. Several approaches have been taken to mitigate the expense of the Jacobian fill, and they are outlined in section IV of this report.

Calculate $\bar{\rho}_\gamma$ and $\bar{\bar{\rho}}_\gamma$ everywhere on mesh:

```

Loop over Nodes; inode{
  Loop over components,  $\alpha$ {
    Loop over Delta stencil: isten {
       $\bar{\rho}_{\gamma=3}[\text{inode}] += C_\alpha^{(\gamma)} \omega_\alpha^{\delta, \text{isten}} \rho_\alpha^{\text{isten}}$ 
      Loop over dimensions (idim){
         $\bar{\bar{\rho}}_{\gamma=v_1, v_2}[\text{idim}][\text{inode}] += \bar{C}_\alpha^{(\gamma)}[\text{idim}] \omega_\alpha^{\delta, \text{isten}} \rho_\alpha^{\text{isten}}$ 
      }
    }

    Loop over Theta stencil: isten{
       $\bar{\rho}_{\gamma=3}[\text{inode}] += C_\alpha^{(3)} \omega_\alpha^{\theta, \text{isten}} \rho_\alpha^{\text{isten}}$ 
    }
  } /* end of  $\alpha$  loop */
} /* end of loop over nodes */

```

Calculate $\frac{\partial \Phi}{\partial \bar{\rho}_\gamma}[\text{inode}]$.

Loop over Nodes; Calculate $\frac{\partial \Phi}{\partial \bar{\rho}_\gamma}$ from $\bar{\rho}_\gamma$'s.

Load Residual and Jacobian

```

Loop over Nodes; inode
Loop over Components;  $\alpha$ 
  Calculate unknown index; iunk
  If Zero_den_TF[iunk]=TRUE; resid[iunk] += x[iunk]
  Else {
    Load Diagonal Terms:
      Resid[iunk] +=  $\log(x[iunk]) + V^{\text{ext}} - \beta \mu$ 
      J[iunk][iunk] +=  $1/x[iunk]$ 

    Load Delta Stencil Contributions to Residual and all
    associated Jacobian Entries:
      Call load_nonlocal_hs_rosen(Delta, inode,  $\alpha$ )

    Load Theta Stencil Contributions to Residual and all
    associated Jacobian Entries:
      Call load_nonlocal_hs_rosen(Theta, inode,  $\alpha$ )
  }

```

Perform Linear Solve

Update solution vector

Repeat Until Convergence.

Fig.III.2: Basic Algorithm for Applying Newton's method to the DFT equations.

load_nonlocal_hs_rozen(DELTA, inode, α)

Loop over Stencil: isten {

$$\text{resid[iunk]} += \sum_{\gamma=0}^2 C_{\alpha}^{(\gamma)} \omega_{\alpha}^{\delta, \text{isten}} \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{\gamma}}$$

$$\text{resid[iunk]} += \omega_{\alpha}^{\delta, \text{isten}} \left\{ \bar{C}_{\alpha}^{(V1)} \cdot \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{V1}} + \bar{C}_{\alpha}^{(V2)} \cdot \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{V2}} \right\}$$

$$\text{calculate } \frac{\partial}{\partial \bar{\rho}_{\varepsilon \leq 3}} \left[\sum_{\gamma=0}^2 \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{\delta, \text{isten}} \right] \text{ for each } \varepsilon$$

$$\text{Loop over idim: calculate: } \frac{\partial}{\partial \bar{\rho}_{\varepsilon=V1, V2}} \left(\sum_{\gamma=0}^2 \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{\delta, \text{isten}} \right) [\text{idim}]$$

loop over components, β {

loop over Delta Stencil: jsten{

Calculate unknown number, j , from jnode and β

if (Zero_dens_TF[jnode, β]=FALSE) {

$$J_{ij} += \omega_{\beta}^{\delta, \text{jsten}} \sum_{\varepsilon=0}^2 C_{\beta}^{(\varepsilon)} \frac{\partial}{\partial \bar{\rho}_{\varepsilon}} \left[\sum_{\gamma=0}^2 \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{\delta, \text{isten}} \right] +$$

$$\omega_{\beta}^{\delta, \text{jsten}} \left(\frac{\vec{r}}{r} \right) \cdot \left(\sum_{\varepsilon=V1, V2} |C_{\beta}^{(\varepsilon)}| \frac{\partial}{\partial \bar{\rho}_{\varepsilon}} \left[\sum_{\gamma=0}^2 \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{\delta, \text{isten}} \right] \right)$$

}

} /* end of loop over Delta Function Stencil */

loop over Theta Stencil: jsten{

Calculate unknown number, j , from jnode and β

if (Zero_dens_TF[jnode, β]=FALSE) {

$$J_{ij} += \omega_{\beta}^{\theta, \text{jsten}} C_{\beta}^{(3)} \frac{\partial}{\partial \bar{\rho}_3} \left[\sum_{\gamma=0}^2 \frac{\partial \Phi^{\text{isten}}}{\partial \bar{\rho}_{\gamma}} C_{\alpha}^{(\gamma)} \omega_{\alpha}^{\delta, \text{isten}} \right]$$

}

} /* end of loop over Theta function stencil */

} /* end of loop over β components */

} /* end of loop over isten */

Figure III.3: Algorithm for Residual and Jacobian fill starting with the Delta Function Stencil.


```

load_nonlocal_hs_rosen (Sten_type=THETA, inode, α)

Loop over Stencil: isten {

    resid[iunk] +=  $C_{\alpha}^{(3)} \omega_{\alpha}^{\theta, isten} \frac{\partial \Phi^{isten}}{\partial \bar{\rho}_3}$ 

    calculate  $\frac{\partial}{\partial \bar{\rho}_{\epsilon \leq 3}} \left( \frac{\partial \Phi^{isten}}{\partial \bar{\rho}_3} C_{\alpha}^{(3)} \omega_{\alpha}^{\theta, isten} \right)$  for each  $\epsilon$ 

    Loop over idim: calculate:  $\frac{\partial}{\partial \bar{\rho}_{\epsilon=V1, V2}} \left( \frac{\partial \Phi^{isten}}{\partial \bar{\rho}_3} C_{\alpha}^{(3)} \omega_{\alpha}^{\theta, isten} \right) [idim]$ 

    loop over components, β{

        loop over Delta Stencil: jsten{
            Calculate unknown number, j, from jnode and β

            if (Zero_dens_TF[jnode, β]=FALSE) {
                
$$J_{ij} += \omega_{\beta}^{\delta, jsten} \sum_{\epsilon=0}^2 C_{\beta}^{(\epsilon)} \frac{\partial}{\partial \bar{\rho}_{\epsilon}} \left[ \frac{\partial \Phi^{isten}}{\partial \bar{\rho}_3} C_{\alpha}^{(3)} \omega_{\alpha}^{\theta, isten} \right] +$$


$$\omega_{\beta}^{\delta, jsten} \left( \frac{\vec{r}}{r} \right) \cdot \left( \sum_{\epsilon=V1, V2} |C_{\beta}^{(\epsilon)}| \frac{\partial}{\partial \bar{\rho}_{\epsilon}} \left[ \frac{\partial \Phi^{isten}}{\partial \bar{\rho}_3} C_{\alpha}^{(3)} \omega_{\alpha}^{\theta, isten} \right] \right)$$

            }

        } /* end of loop over Delta Function Stencil */

        loop over Theta Stencil: jsten{
            Calculate unknown number, j, from jnode and β

            if (Zero_dens_TF[jnode, β]=FALSE) {
                
$$J_{ij} += \omega_{\beta}^{\theta, jsten} C_{\beta}^{(3)} \frac{\partial}{\partial \bar{\rho}_3} \left[ \frac{\partial \Phi^{isten}}{\partial \bar{\rho}_3} C_{\alpha}^{(3)} \omega_{\alpha}^{\theta, isten} \right]$$

            }

        } /* end of loop over Theta function stencil */

    } /* end of loop over β components */

} /* end of loop over isten */

```

Figure III.4: Algorithm for Residual and Jacobian fill starting with the Theta Function Stencil

Section III.D Parallelization and Scaling

When implementing the DFT solve on massively parallel distributed memory computers, the strategy is to split up the domain so that each processor loads the Jacobian entries for the rows of the unknowns within this domain. Because the Jacobian and Residual calculations require information outside the local domain, three coordinate systems are ultimately required. Communications are performed using MPI calls and Aztec[16] utilities.

The first coordinate system is the local coordinate system. It contains all the nodes that a given processor owns. The indices on the local coordinate system do not follow any particular geometrical pattern. The second coordinate system is the global coordinate system. Global coordinates are needed to check for boundary conditions, and provide a reference frame for the integer operations performed on the mesh. Node numbering in the global coordinate system begins with 0 at the left(x), lower(y), back(z) boundary. The global nodes increment first in x then in y then in z, leading to a simple integer i-j-k coordinate system.

Finally, in parallel, it is necessary to introduce an extended local coordinate system. This coordinate system contains all of the local nodes on a given processor plus a rectangular cage around them that contains all the nodes needed for processing the Jacobian rows associated with the local nodes. The extended local coordinates are set up as a rectangular cage in order to be able to use an i-j-k integer numbering scheme as is used with the global coordinates. For hard sphere (or charged hard sphere) systems, the Jacobian entries for a given node are all those that are within two stencil lengths of a given local node. Therefore, the boundaries of the rectangular domain of the extended coordinate system will be found by subtracting/adding twice the maximum stencil length from the local node that has the minimum/maximum position in each dimension. For Lennard-Jones fluids, the cut-off of the potential is typically longer than twice the maximum hard sphere stencil length. In these cases, the LJ cut-off parameter is used to define the size of the extended local domain. Fig.III.5 shows sketches of the local and extended local coordinate systems.

At global domain boundaries, the extended local coordinate system is adjusted depending on the boundary

Fig.III.5 A comparison of local (A) and extended local (B) coordinate systems for parallel implementation of the DFT solve.

condition of choice. If the domain boundary condition is a bulk fluid or is in a wall or is a reflective boundary, it is not necessary to include points beyond the global domain boundary. On the other hand for periodic boundaries, it is necessary to extend the extended local coordinate system beyond the computational domain. The values of the unknowns on these extended points are set equal to their values on the opposite side of the box. The primary advantages of including these points explicitly are that boundary checking is not needed, and that communications with processors owning nodes on the opposite side of the domain can be eliminated.

Load balancing is one final challenge for the parallel solve of the DFT. Load balancing on our domain is based on a weighted recursive spectral bisection method. At the beginning of the calculation, nothing is known about the surface configuration so the nodes are given equal weight of one and split evenly between the processors. Once the surface boundary elements are identified, the load balance is redone. The nodes that are in the surface, and for which the equation $x=0$ is solved are given weights near zero. Nodes that are being treated with a residual coarsening method (outlined below in Section IV.A.1) also have weights below one. Otherwise if a fluid node is near a surface or domain boundary, the weights are higher than the bulk. This heuristic approach is modestly successful in balancing the work between processors. It will ensure that no processor owns only surface nodes that require no work. However, it is not sophisticated enough to determine which precise nodes cause the bottlenecks.

In an earlier implementation of the code, one approach taken to overcome the bottleneck of the load balance was to keep track of the time each node required for in the Residual/Jacobian fill routines, and then use these times as weights. This approach was successful in achieving very good load balance on the fill, but it sometimes caused significant imbalances in the matrix solve. As of November 1998, this particular option is not available in TRAMONTO. Further work in developing load balancing approaches is needed.

Finally, two series of runs are presented to demonstrate the parallel scaling of the algorithms. The first is a two-dimensional calculation of one component hard-sphere fluid adsorbing to a single hard wall, and decaying to a bulk density of $\rho\sigma^3=0.6$. The computational domain is sized $10\sigma \times 8\sigma$ with a mesh spacing of 0.1σ in both dimensions (where σ is the fluid molecular diameter). This same computation was performed 11 times on different numbers of processors, from 2 to 64, on the Sandia-Intel Tflops (ASCI-Red) computer. All calculations converged in 6 Newton iterations, and produced identical results to all decimal places.

Fig.III.6 presents the fixed-problem scaling results for this example. The speed-up is seen to drop below the linear theoretical limit for the larger number of processors, which is mostly due to the expense of communications and time for redundant calculations as each processor owns a smaller part of the domain, and partly due to load imbalances among processors. Still, with a 32-fold increase from 2 to 64 processors, the solution time is seen to be 81% of linear.

The second example is scaled speed-up example, where the problem size per processor is held fixed. A small, hard-sphere 3D problem was chosen, that had 1-component and 1 wall. The domain size on 10 processors was $8 \times 2 \times 2$ with a mesh size of $0.2 \times 0.2 \times 0.2$. The third dimension was increased proportional to the number of processors, to a maximum of 72 that was run on 360 processors.

Twenty timings over 1 Newton iteration are shown in Fig.III.7 as a function of number of processors. The timings remain very flat over all runs, with only a 5.5% difference between the

minimum at 10 processors to the maximum at 100 processors. The random nature of the timings is likely due to the integer decisions made in the load balancing.

Fig.III.6 . Speed-up curve of a 2D, 1-component problem. The solution time speed-up on 64 procesors is 81% of the theoretical limit.

Fig.III.7. Scaled speed-up curve of a 3D, 1-component problem, where the problem size was increased proportional to the number of processors. The solution time remains nearly constant over a 36 fold increase in the number of processors.

Section IV Code Performance

In section III, the basic algorithms in TRAMONTO were discussed. In this section a variety of algorithms for enhanced code performance are outlined. In addition, timings associated with different solver options and physics options are presented.

Section IV.A Algorithms for Improved Performance

Solving the DFT equations using the Jacobian fill routine outlined in Figs.III.3 and III.4 is expensive for 2D and particularly 3D problems. Therefore, several strategies for mitigating the expense of the fill were implemented. They include Jacobian coarsening, Mesh coarsening, and Approximate/pre-calculated Jacobian. All of these options are available in TRAMONTO as of November, 1998, and they are discussed briefly below. It should be noted that these strategies may be used simultaneously, and the performance of the code with one or all of these options is discussed in Section IV.B

Section IV.A.1 Mesh Coarsening

In section III.B it was pointed out that implementing a nonuniform mesh for the DFT equations would be costly due to the necessity to store dense quadrature stencils at every node. An alternative is to apply a nonuniform mesh that requires only a limited number of additional quadrature stencils. Such a mesh is shown in Fig.IV.1 where a mesh becomes coarser in steps away from a surface. For these nonuniform meshes, a complete set of quadrature stencils is needed only for each region of constant mesh density.

The regions of constant mesh density are called zones. The zone to which a given node belongs is determined by the shortest distance between the node and any of the surfaces in the system. The total number of zones in a calculation, and the distances corresponding to break points between zones, are adjustable inputs set by the analyst. In each successive zone away from the surface, the mesh is coarsened by a factor of 2. The maximum mesh density that can yield sensible results (in a uniform density region) is $\Delta x = 0$. Thus the maximum number of zones is determined by the number of

factors of two between the minimum mesh density and $\Delta x = 0.5 \sigma$. Typically, there are between 1 and 5 zones with most of the performance benefit achieved when there are 2 or 3 zones.

Fig.IV.1 A schematic of the coarsened mesh of the type used in TRAMONTO. The dashed lines indicate the finest underlying mesh. The bold lines indicate the remaining mesh after coarsening. There are 3 zones with a flat wall on the left side of the domain.

The way this nonuniform mesh is implemented without storing separate quadratures for each point is to retain in some form the rows of the Jacobian corresponding to the dropped nodes (intersection of dashed lines in Fig.IV.1). That is, stencils starting in zone 1 will at times land in zone 2 at nodes that are coarsened out of the mesh, and instead of removing them from the mesh, we retain these unknowns and just set the residual to the mean of the surrounding nodes. For example, the residual equation for the "dropped" nodes in 1D becomes

$$\rho_i - 0.5 * \rho_{i-1} - 0.5 * \rho_{i+1} = 0. \quad (58)$$

Since this equation is orders of magnitude easier to fill than the Euler-Lagrange equations, there is negligible expense with retaining these coarsened nodes in the mesh. The Jacobian contributions of Eq.58 are just 1 on the diagonal and -0.5 in the columns corresponding to the $i-1$ and $i+1$ node. Since nodes are not completely dropped from the calculation, it is straightforward to apply coarsened stencils at any point in the mesh. The particular stencil used will be dictated by the zone in which node of origin of the integration is found. For example consider a 1D calculation that has two zones. If the densest stencil is of length 13 with offsets of (-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6), the coarsened stencil will have 7 points with offsets of (-6,-4,-2,0,2,4,6). All the odd numbered entries are removed, and the weights are adjusted accordingly.

Another option is to solve for the accurate residuals with less accurate Jacobian entries based on coarser stencils. When the Jacobian quadrature stencils are coarsened, but the *Residual Coarsening* switch is turned off, there is still substantial savings both in fill time and in memory required. All of the columns that would correspond to the dropped nodes are removed from the Jacobian creating a much sparser matrix. Furthermore, since the fill scales like N_{sten}^2 , the fill time per node for zones 2,3,4,... will be 4,9,16,... times faster than the most dense zone.

The total savings due to mesh coarsening is dependent on the fraction of the fluid nodes that are near the surfaces and the quality of the load balancing. Typically, accurate results can be achieved when the finest zone has a range of approximately 2σ and a mesh density of $\leq 0.1\sigma$. If the fraction of fluid nodes in each zone is χ_{zone} , the fill time will be proportional to

$$\sum_{n=1}^{N_{zone}} \chi_n \frac{N_{nodes} N_{comp}^2 N_{sten,zone1}^2}{2^{2*N_{dim}*(n-1)}}. \quad (59)$$

Section IV.A.2 Jacobian Coarsening.

While mesh coarsening is a significant improvement over the basic algorithm in Section III.C if there are a significant number of nodes in the first zone away from the surfaces, performance can be improved further by applying Jacobian coarsening beyond what is dictated by the local zone. Several Jacobian coarsening options are possible in TRAMONTO. They include:

- ◆ Coarsen Jacobian in zone 1 (nearest to surfaces) only; by a factor of 2
- ◆ Coarsen all but most coarse zone by a factor of 2.
- ◆ Use stencil of coarsest zone everywhere
- ◆ Use stencil of 2nd coarsest zone everywhere except in coarsest zone

Applying further coarsening to the Jacobian will improve performance even when a significant number of fluid nodes are near surfaces. For example, the fill time in the most dense zone (zone 1) with the first option above will be proportional to

$$\chi_1 \frac{N_{nodes} N_{comp}^2 N_{sten,zone1}^2}{2^{2*N_{dim}}} \quad (60)$$

Section IV.A.3 Approximate - pre-calculated Jacobian

A completely different approach for improved performance is to recognize that the innermost loops in the Jacobian fill (over β components and stencils) are identical to the loops in the $\bar{p}$ calculation. This gives us the opportunity to pre-calculate and store $\omega_{\beta}^{\delta}[\text{inode}][\text{jsten}]$ and $\omega_{\beta}^{\theta}[\text{inode}][\text{jsten}]$ as well as the associated column numbers, $\text{jac_col_delta}[\text{iunk}][\text{isten}]$ and $\text{jac_col_theta}[\text{iunk}][\text{isten}]$ in the Jacobian. While these arrays are large requiring a large amount of storage, and only include some of the terms in the Jacobian, they are often good enough for Newton's method to converge. The entire Jacobian fill for the hard-sphere terms, which is the majority of the fill time, is replaced by some copies from memory.

In order to mitigate the storage required for the pre-calculated Jacobian, this approach is combined with an approximate Jacobian. There are two type of approximate Jacobians that can be applied. The first includes only two weight functions, $w^{(3)}$ and $w^{(2)}$. In both of these cases, $C_{\beta}^{(\gamma)} = 1$. The second case includes all the scalar weight functions, $w^{(0)}$ - $w^{(3)}$, but uses the radius of the first component for all the prefactors.

The performance gain from this approach comes from minimizing the number of operations performed in the innermost loop of the Jacobian. For the previous approaches, it is necessary in the final loop over stencils to: find $jnode$ given a set of offsets from an $isten$ node, check and apply appropriate boundary conditions, perform the final multiplication by the $\omega_{\beta}^{\delta,\theta}$ weights, and copy the result into memory. With the present approach, there is only the final multiplication and the copy to memory. The number of operations saved per loop is 2-10 (depending on the complexity of the boundary condition and the number of dimensions for which offsets must be applied). As a result, the fill time is immediately decreased by the same factor of 2-10. While considerable speed-up is obtained with the approximate Jacobians, the price to be paid in some cases is in decreased stability of solutions (see Section IV.B).

Section IV.B Performance Timings

The purpose of this section is to show what type of performance can be expected from the code under a variety of conditions. All of the calculations are based on the system configuration sketched in Fig.IV.2. The system consists of a single smooth planar wall on the left side (taking up the left 1σ) of the domain. The left boundary is assumed to be an infinite wall, the right boundary is set to the bulk fluid, and the top and bottom boundaries are periodic. The size of the domain is $10\sigma \times 10\sigma$. The mesh spacing perpendicular to the surface is $\Delta x = 0.05\sigma$ while the mesh spacing parallel to the surface is $\Delta y = 0.1\sigma$. The arbitrary characteristic length, σ , is set equal to the diameter of the fluid particles. In the subsequent sections, timings are given for solutions with various fill algorithms and underlying physics.

A series of 8 runs were performed to compare the different matrix fill options. The problem had a 1-component hard-sphere fluid near a hard wall with a bulk density of 0.7σ on the right. All runs were performed on 32 processors with the heuristic "weights" load balancing option. Three switches in the input file were toggled.

The number of zones was set at 1 or 3. When at 3, the mesh was coarsened by a factor of 2 in each direction at a distance of 2.0σ from the wall, and another factor of 2 a distance 4.0σ from the wall. While this theoretically decreases the accuracy of the solution, the integrated adsorption of the component changed only 0.1% between the runs. When the mesh was coarsened by a

factor of 2 everywhere, the adsorption changed by 30%, verifying that the fine mesh is needed within the first 2 diameters of the wall, but not outside of that region.

The second switch that was toggled was the approximate, pre-calculated versus exact Jacobian. The particular approximate Jacobian considered was the first option where only two weight functions are included. The third switch in these runs was the Jacobian coarsening switch, where a coarser stencil was used in the Jacobian calculation than in the Residual calculation. Neither of these switches affect the residual calculation, and therefore they do not affect the accuracy of the solution.

The results of these 8 runs are shown in Table.IV.1.

Run #	Nzone	Residual Coarsening	Jacobian	Jacobian Coarsening	# Newton Iterations	Solution Time (secs)
1	1	No	Exact	No	7	271
2	3	No	Exact	Yes	7	107
3	1	No	Approximate	No	8	231
4	3	No	Approximate	Yes	8	102
5	3	Yes	Exact	No	7	82
6	3	Yes	Exact	Yes	7	37
7	3	Yes	Approximate	No	8	73
8	3	Yes	Approximate	Yes	8	36

Table IV.1 A comparison of Solution times for various choices of the fill algorithm, for a 2D, 1-component hard sphere problem on 32 processors.

Comparing runs 1-4 with 5-8 shows that the mesh coarsening saved a factor of 3 in the solution time (with only 0.1% added error). The exact vs. approximate (pre-calculated) Jacobian flag has surprisingly little effect on this problem. The small savings in fill time is countered by an increase

from 7 to 8 Newton iterations. A factor of 2-3 is gained by coarsening the Jacobian. Overall, the solution time is reduced over a factor of 7 by appropriate choices of mesh and Jacobian coarsening. Evidence indicates that this can be increased to a factor of 10 with improved load balancing for the Jacobian coarsening examples.

We now turn to code performance as a function of the physical model of interest. We specifically compare the slowest algorithms and the fastest algorithms. In the case of the slowest algorithms, the exact Jacobian is calculated, there is only 1 zone in the calculation (i.e. the mesh is uniform), and there is no Jacobian coarsening. In the case of the fastest algorithms, the pre-calculated approximate Jacobian is used. In addition, the mesh has three zones away from the surface. The break between the first and second is found at $x=2\sigma$, while the break between the second and third zones is found at $x=4\sigma$. Finally, Jacobian coarsening is applied in the first zone closest to the surface.

The three fluid models we consider are:

- ◆ A hard sphere fluid in contact with a hard wall.
- ◆ A 12-6 Lennard-Jones fluid in contact with a 9-3 Lennard-Jones wall.
- ◆ A 1:1 3-component finite size electrolyte model in contact with a charged wall.

In the hard sphere/hard wall case, the bulk fluid density is representative of a liquid-like fluid, and is set to $\rho\sigma^3=0.7$. The density as a function of perpendicular distance from the surface for this case is shown in Fig.IV.2.

In the LJ case, the bulk fluid density is in the liquid regime, and is $\rho\sigma^3=0.71$. The fluid-fluid (and fluid-wall) cut-off distances are $r_{\text{cut}}=2.5\sigma$, and the wall-fluid interaction potential parameter relative to the fluid-fluid interaction potential parameter is $\epsilon_{\text{wf}} = 3\epsilon$. This wall-fluid interaction corresponds to favorable (hydrophilic) wall-fluid interactions. The density as a function of perpendicular distance from the surface for this case is shown in Fig.IV.3

In the electrolyte case, the fluid has three components including a solvent at a liquid-like density of $\rho\sigma^3=0.7$, and two ions (counter-ions and co-ions) both with a bulk density of $\rho\sigma^3=0.005$. The three components all have identical size of $\sigma = 0.425\text{nm}$. The temperature parameter is set to $4\pi kT\epsilon\epsilon_0\sigma/e^2 = 0.7$. Finally, the surface charge density is $q\sigma^2/e = 0.2$. The densities and electrostatic potential as a function of perpendicular distance from the surface for this case are shown in Fig.IV.4.

Fig.IV.2 The density profile in a hard-sphere fluid perpendicular to a smooth planar hard wall. The wall-fluid surface is located at 1σ

Fig.IV.3 The density profile in a 12-6 LJ fluid perpendicular to a smooth planar LJ wall. The wall-fluid surface is located at 1σ .

The performance studies discussed below were performed either on 40 processors (Hard-sphere and Lennard-Jones) or 80 processors (Electrolyte) of the ASCI Red Computer at Sandia National Laboratories. Results corresponding to a variety of the fill algorithms, linear solver tolerances, and load balancing switches are presented. Two types of approximate (pre-calculated) Jacobians are considered. Option #1 uses only $w^{(3)}$ and $w^{(2)}$ terms in the innermost loop of the Jacobian fill while in option #2, all the scalar weight functions, $w^{(0)}$ - $w^{(3)}$ are included. In both cases, the vector weight functions are left off. The only Jacobian coarsening used for these timings has the finest zone of the mesh coarsened by a factor of two with respect to the integrals in the Jacobian fill. For all the cases with mesh coarsening, three zones are used. The ranges for the zones with respect to distance from the surface are 0 - 2σ , 2 - 4σ , and 4 - 9σ .

The two load balance options presented in these timings are labeled *geometry* and *weight* where the *geometry* option indicates a mesh partition where each processor has nearly the same number of nodes. The *weight* option refers to a weighted bisection where each node is assigned a weight based on how complex the fill is expected to be for that node. The bisection then attempts to get close to an equal weighting on all the processors.

Table IV.2 details the different parameter combinations, and tables IV.3 and IV.4 summarize the number of Newton iterations, the total time needed for a complete solve, the fill time (residual and Jacobian), and the linear solve time per Newton iteration for each run.

Run #	Jacobian	Nzone	Jacobian Coarsening	Linear Solver Toler	Load Balancing
1	Exact	1	No	10^{-4}	Geometry
2	Exact	3	Yes	10^{-4}	Geometry
3	Approximate #1	3	Yes	10^{-4}	Geometry
4	Approximate #2	3	Yes	10^{-4}	Geometry
5	Exact	1	No	10^{-2}	Geometry
6	Exact	3	Yes	10^{-2}	Geometry
7	Approximate #1	3	Yes	10^{-2}	Geometry
8	Approximate #2	3	Yes	10^{-2}	Geometry
9	Exact	1	No	10^{-2}	Weights
10	Exact	3	Yes	10^{-2}	Weights
11	Approximate #1	3	Yes	10^{-2}	Weights
12	Approximate #2	3	Yes	10^{-2}	Weights

Table IV.2 *A description of the fill algorithms and linear solver and load balancing parameters used to compare different physical systems.*

The timings in Tables IV.3 and IV.4, demonstrate that for all the physical systems of interest the solution time can be decreased by about a factor of 10 if the appropriate combination of fill algorithms are used. However, the optimal combination of fill algorithms is problem dependent. The approximate Jacobian #1 option reduces the fill time per Newton iteration by about a factor of two, it also causes an increase in the number of Newton iterations required to solve the problem. The increase in the required iterations is reduced when the option #2 approximate Jacobian is used. However, the additional floating point operations needed for this option cause the fill time per iteration to increase somewhat. The result is that for the electrolyte the fastest solution (case 11) is found with the option #1 approximate Jacobian while for the hard sphere and Lennard-Jones cases, the option #2 approximate Jacobian is better. It should also be noted that for the hard sphere and Lennard-Jones cases, it is nearly as fast (in terms of the total run time) to use the exact Jacobian with zones and Jacobian coarsening. In these cases, the number of extra iterations required for the approximate Jacobian make its use only marginally helpful. Therefore, before applying the approximate Jacobian it is prudent to check its effect on stability.

In addition to fill algorithms, Tables IV.3 and IV.4 show the effect of adjusting the linear solver tolerance and the load balancing. Reducing the required accuracy of the linear solve by two orders of magnitude decreases the linear solve time by roughly a factor of two. Using the weighted load balance switch when mesh coarsening is turned on is critical, decreasing the Jacobian fill time by about a factor of four and the solve time by a factor of two. When there is no mesh coarsening, there is not as much difference between the geometry and weight based bisection methods for load balancing. This result is expected since the assumption in the geometric weighting, that all nodes take equal time, is a reasonable assumption for this problem and these settings.

Finally, it should be noted that the cases chosen for these studies are all dense liquid cases where the fluid structure is sharply peaked. The best fill algorithms will be dependent on the state point of the fluid as well as the type of fluid of interest. At lower bulk densities or higher temperatures, the density distributions will not be as sharply peaked. In these cases, the approximate Jacobians will be more closely represent the true Jacobian, and so more speed-up can be expected in these cases.

Run #	Hard Spheres		Lennard-Jones		Electrolyte	
	Niter	Time/sec	Niter	Time/sec	Niter	Time/sec
1	6	180	15	515	12	2138
2	6	63	15	209	16	1542
3	13	72	30	342	21	1491
4	8	61	16	200	17	1300
5	6	179	17	538	16	2408
6	6	62	17	191	17	1200
7	13	71	36	300	18	746
8	8	56	20	178	17	862
9	6	154	17	551	17	2088
10	6	19	17	59	22	439
11	13	25	36	98	18	208
12	8	19	20	63	22	323

Table IV.3 *The number of Newton iterations and total solve time required for a variety of physical systems, fill algorithms, and linear solver tolerances.*

Run #	Hard Spheres		Lennard-Jones		Electrolyte	
	Fill / sec	Solve / sec	Fill / sec	Solve / sec	Fill / sec	Solve / sec
1	25.1	1.2	25.6	6.2	123	46.1
2	8.1	1.1	6.4	6.1	44.3	48.8
3	3.7	1.1	4.1	6.3	21.2	47.7
4	5.4	1.3	2.5	6.7	24.5	49.7
5	25.1	1.0	25.6	3.8	122	20.6
6	8.0	0.8	6.4	3.4	44.3	23.2
7	3.7	1.0	4.1	3.3	21.2	18.0
8	5.4	0.77	4.4	3.4	26.1	22.2
9	21	1.2	26.1	3.8	105	18.7
10	2.0	0.43	1.7	1.0	9.4	9.2
11	1.1	0.37	1.3	1.0	4.6	5.7
12	1.4	0.38	1.3	1.0	5.4	8

Table IV.4 *The time required for a single fill (Jacobian and residual) and a single linear solve for a variety of physical systems, fill algorithms, and linear solver tolerances.*

Section V. Summary of Applications

In this section, several applications of the DFT are briefly outlined. More details are given in the several peer-reviewed journal articles that have been recently published. The purpose of this section is to present a spectrum of applications to which the DFT code has been applied in order to demonstrate the utility of the method. Before proceeding with the applications, we note that the calculations in Section VA. and VB, were performed using a 1D serial code developed in a collaboration between Laura Frink (org. 9225) and Frank van Swol (org. 1841). This 1D code solves the DFT equations using successive substitution (or Picard iterations), and relied on assumptions that were not extensible to 2D and 3D calculations. Furthermore it is a FORTRAN 77 code while TRAMONTO was written in C for memory management capabilities and to facilitate interfacing with the parallel linear iterative solver tools (Aztec). If both codes are run on identical 1D problems, TRAMONTO is more stable. TRAMONTO usually requires fewer than 20 Newton iterations to reach a solution while the Picard method often requires more than 1000 iterations. Furthermore, Newton's method guarantees quadratic convergence near the solution while Picard iterations have no similar convergence guarantees. For these reasons, the algorithms in TRAMONTO are superior to the preliminary 1D code.

This reason for development of the 1D code was to effectively and quickly demonstrate the applicability of DFT to problems of interest at Sandia National Laboratories. Results from 1D DFT calculations also clearly demonstrate that is not always necessary to apply 2D and 3D models to gain needed insight into complex physical systems. Indeed, the analyst is encouraged to consider the possibility of applying 1D models (along with superposition for example; see Section VB) before proceeding to more computationally intensive 2D and 3D calculations.

In addition to results from application of the preliminary 1D code, two 2D applications (results obtained with TRAMONTO) are presented. They include wetting and self-assembly at a chemically heterogeneous surface (Section V.C), and solvation forces between polyelectrolytes in solution (Section V.D). These applications were done concurrently with development and refinement of the TRAMONTO code in a collaboration between Laura Frink (org. 9225) and Andrew Salinger (org. 9221).

Section VA. Characterization of Microporous Thin Films

The first application of the 1D DFTcode discussed here involves providing a theoretical interpretation for beam bending experiments on microporous ($<10\text{\AA}$ pore diameters) thin film materials. The experiments were performed by Jeff Brinker and Josh Samuel (org. 1831), and the analysis was done in a collaboration between Josh Samuel, Jeff Brinker, Laura Frink, and Frank van Swol. A schematic of the beam bending experiment is shown on the left in Fig.VA.1. A porous thin film is attached to a silicon wafer and suspended in a closed chamber. The relative pressure of sorbate in the chamber is controlled, and the substrate bends slightly in response to adsorption in the film. A He/Ne laser is used to detect deflection of the substrate, and given the elastic properties of the silicon wafer it is possible to relate the deflection to the stress exerted by the solvent on the thin film. When the relative pressure of the sorbate is varied

between vacuum and saturation, a solvation *stress* isotherm is obtained.

Typical experimental stress isotherms are shown in Fig.VA.2. The films exhibit monotonic compressive stresses with a limiting slope (at saturation $p/p_0=1$) predicted by Laplace-Kelvin theory (LKT). Observing the LKT slope is remarkable because the films are known to be microporous, but LKT is well known to fail in pores that are only a few sorbate diameters in size. The challenges for the DFT calculations were:

- to reproduce the experimental results
- to understand how LKT can be correct for the slope, and
- to reconcile the results with other microscopic measures of solvation forces.

The most notable surface forces measurement with which beam bending experiments must be reconciled are surface forces apparatus (SFA) measurements where surface forces are routinely found to be oscillatory with respect to surface separation when the surfaces are less than 20Å apart.

Our analysis centered on a slit-like pore model, and began with a single pore (or monodisperse porous network). In this geometry, 1D DFT calculations could be performed. The dotted line in Fig.VA.3 shows the solvation force as a function of surface separation for a single uniform slit pore. Clearly, the solvation force oscillates as has been measured in SFA experiments. Unfortunately, when the solvation force is plotted as a function of relative pressure at constant pore size, the results of BB experiments (e.g. Fig. VA.2) are not reproduced. In the monodisperse pore model, the oscillatory nature of the underlying force leads to a wide variation of behaviors (some tensile, some compressive) depending on the particular pore size chosen.

The reproducibility of the BB experiments when a variety of sorbates and film types are used precludes the notion that the pores just happen to be of a size that will yield compressive forces. Rather, given the random nature of the matrix, it is more likely that there is some polydispersity in pore size.

Therefore, a narrow Gaussian pore size distribution was assumed, and the distribution was cut off at two standard deviations, t , from the mean. The standard deviation was allowed to range from $t=0.2-1.0\sigma$. The mean solvation force in a pore network with $t=0.5\sigma$ is shown in the solid line of Fig.VA.3. The introduction of this small degree of polydispersity reduces significantly the solvation force oscillations. When $t=0.3\sigma$ damped oscillations are still observed, both when $t=0.5\sigma$ as in Fig.VA.3, the oscillations are almost completely eliminated. However, while the oscillations disappear, the mean solvation force is not zero everywhere. Rather for small pore sizes, a residual compressive stress is found.

Repeating DFT calculations for a variety of relative pressures resulted in the stress isotherms shown in Fig.VA.4. Just as with the experiments, a monotonic compressive stress is obtained. Going beyond the qualitative predictions, we were also able to predict observed experimental trends with changing sorbate size. Ultimately, the experimental measurements and theoretical predictions could be directly compared to determine the mean pore size and width of the pore size distribution. In addition, by studying a variety of pore sizes, we were able to show that there are three qualitatively different kinds of stress isotherms that can be measured as the pore size increases from microporous to mesoporous. These qualitative difference allow for a rapid assessment of the approximate pore sizes in a given material. Thus, while the BB experiment provides a new means of measuring solvation forces in complex porous materials, the analysis provided by DFT calculations gives a new means for characterizing microporous materials from those solvation stress isotherms. More details of the experimental measurements, and DFT analysis can be found in[2].

Section VB. Surface forces: Adhesion and Surface Roughness

In addition to developing the underlying theory for the BB experiments, Surface Forces Apparatus (SFA) measurements for several kinds of systems have been analyzed. In this section, we briefly summarize two cases. In the first, adhesion of two surfaces in the presence of capillary condensation is assessed. In the second solvation forces between rough surfaces are predicted. Both studies were done in a collaboration between Laura Frink (org. 9225) and Frank van Swol (org. 1841).

Adhesion is very important to the stability of materials, and one way to measure the possibility of failure in a solid-solid interface is to measure the force required to pull two surfaces apart. This pull-off force will be affected not only by direct surface-surface interactions, but also by the presence of capillary condensed fluid at the interface. The surface forces apparatus consists of a geometry of two crossed cylinders which is equivalent to a sphere interacting with a flat plate. In order to analyze pull-off forces, solvation forces were predicted for two flat plates interacting through a solvent. The net force for the sphere-plate geometry was then found by superposition of the flat plate results with a weighting dictated by the sphere-plate geometry.

Pull-off forces were predicted for a variety of relative pressures of the capillary condensed fluid and for a variety of surface-fluid, fluid-fluid, and surface-surface interaction energy parameters. Data from pull-off experiments was compared directly with DFT calculation as is shown in Fig.VB.1.

These calculations helped to identify the existence of three distinct regimes in SFA pull-off data. First there were cases where there was almost no change in pull-off force with relative pressure. These cases were found to correspond to interaction parameters where the surface-surface interactions were dominant. There were also cases where the pull-off force approached its limiting value from below. These cases were identified as cases where fluid-fluid interactions dominate. Finally, there were cases where the limiting value was approached from above. In these cases, wall-fluid interactions are dominant. These calculations were the first systematic molecular analysis of pull-off measurements that have been performed. They showed that capillary condensation can play a major role in the adhesion of two surfaces, and showed that a molecular interpretation is needed to fully understand the effects. More details of the calculations and results can be found in [1].

A similar approach of superposition of forces was taken in an analysis of the solvation forces that can be expected between rough surfaces. In this case, a surface was divided up into a number of plaquettes as sketched in Fig.VB.2.

The plaquettes were then shifted from their mean position according to a desired distribution. The maximum displacement was defined by a roughness parameter, τ . Depending on the number and size of the plaquettes, the underlying distribution from which the plaquette displacements were drawn was reproduced in the actual plaquette configuration with varying accuracy. In order to predict solvation forces between these

rough surfaces, superposition was applied. A set of DFT results for slit pores consisting of smooth surfaces was generated first with the 1D DFT code. These forces were then averaged according to the actual roughness distribution in a given system.

In order to test the superposition approximation for rough surfaces, Grand Canonical Monte Carlo simulations on fluids confined between the rough surfaces were performed. The superposition approximation was found to work surprisingly well, and several examples of a comparison with MC simulations are shown in Fig.IV.B.3 for different values of the roughness parameter. Given the success of the superposition approximation, it was used extensively to predict effects of surface roughness on solvation forces for a wide variety of surface roughness distributions.

These calculations demonstrated the intricate nature of surface roughness effects. It has commonly been assumed that surface roughness will destroy the solvent ordering that leads to oscillatory solvation forces[14]. Our calculations show that as the magnitude of the surface roughness increases, this effect is observed initially. However, when the roughness reaches roughly the size of the solvent molecules, the magnitude of the solvation oscillations increases again. This unexpected effect can be seen in the trends of Fig. IV.B.3, and arises from the fact that when the surface roughness is large enough, the first layer of solvent molecules will fit into the depressions in the surface. The second layer of molecules then 'sees' an effectively smoothed surface with modified chemical interactions.

Taking the analysis further, we also considered surfaces with oscillatory patterns on the surface. These surfaces were inspired by DNA strands which are essentially cylinders with a screw-like profile. We showed that the solvent mediated interactions would force this type of surface to alternate registration as the surfaces approach one another. Understanding how solvation forces act when the surfaces have complex geometry or chemical patterns will ultimately provide insight for manipulating and engineering surfaces and colloidal particles in solution. In addition, most molecular biology occurs in the presence of solvation effects, and understanding the impact of solvation forces on the registration and interaction of proteins and DNA is of critical importance for molecular recognition and drug design. More details of the calculations can be found in [3].

Section VC. Wetting and Self-Assembly at Chemically Heterogeneous Surfaces

While 1D models can be used for an initial (and sometimes final) analysis of complex physical systems, there is a place for higher order models and calculations when the surface geometry is complex or the surface is chemically heterogeneous. Proteins are one example where the functionality of the protein depends critically on the pattern of charged groups on its surface.

As a first step to understanding the role of chemical heterogeneity in self-assembly, DFT calculations were performed on Lennard-Jones fluids interacting with a chemically heterogeneous planar surface. The surface is sketched in Fig.VC.1. The chemical properties of the two stripe types are defined by the wall-fluid interaction energy parameter ϵ_{wf} . The length of the stripes are given by l . In this series of calculations, one stripe type has constant ϵ_{wf} such that its interactions with the fluid are hydrophobic. The properties of the other stripe type are allowed to vary over a large range of ϵ_{wf} . The calculations involved varying l as well as ϵ_{wf} . The calculations were done in a collaboration between Laura Frink (org. 9225) and Andrew Salinger (org. 9221) using TRAMONTO. The calculations were performed on 50-100 processors of the ASCI-Red computer. Each solution required on the order of 100 sec of CPU time.

Fig. VC.1 A schematic of a chemically heterogeneous surface model.

To be more specific, the purpose of the calculations was to identify the effect surface chemical heterogeneity has on the wetting of that surface by a nearby fluid. The wetting properties of a surface may be encapsulated in a calculation of the contact angle which can be found from the surface free energies associated with wall-liquid, wall-vapor, and liquid-vapor interfaces[14]. When a surface/fluid combination is non-wetting (or more loosely very hydrophobic), the contact angle is $\theta=180^\circ$. When a surface/fluid combination is completely wetting (or more loosely very

hydrophilic), the contact angle is $\theta=0^\circ$. The calculations discussed below involve calculating the contact angle as a function of the variable parameters, ϵ_{wf} and l .

Fig.VC.2 shows two examples of the density distributions that were obtained. In both cases, the striped surface is in contact with a bulk vapor. In one case (Fig.VC.2 left), a single adsorption peak that varies in magnitude parallel to the surface is observed. In this case the maximum density is approximately $\rho\sigma^3=0.8$. In the second case (Fig.VC.2 right), a local condensation of liquid-like fluid has occurred near the hydrophilic stripe, and the maximum density is approximately $\rho\sigma^3=4.0$. In the former case, density oscillations parallel to the surface are related to the underlying pattern of the surface. In the latter case, density oscillations parallel to the surface have the period of a solvent diameter. In this case, the oscillations parallel to the surface are solvation (or solvent packing) oscillations that arise from the presence of the liquid-vapor interface perpendicular to the surface at the junction of the two surface types.

The difference between adsorbed gas-like layers and condensed liquid-like layers is marked. The transition between these phases could be quite important in self-assembly processes. If several chemically heterogeneous surfaces (whether they are planes, proteins, or polymers) are interacting with each other in solution, the transition between random and self-assembled configurations may well be linked to a catastrophic change in the wetting properties of the surfaces or macromolecules. While this link to self-assembly is yet to be firmly established, we have proceeded to determine what are the relevant phase transitions in the fluid near the chemically heterogeneous surfaces. These phase transitions will be discussed further below.

As the strength of wall-fluid interactions in the hydrophilic portion of the surface are increased further, the solvation structure in the nanoscopic condensed phases can become extreme. Fig.VC.3 shows one such case. Here, the maximum density is approximately $\rho\sigma^3=20$. In addition, parallel structure can be observed in all of the three layers away from the surface. When the wall-fluid interaction strength is increased further, the system of equations becomes unstable. The sharp density peaks along with the onset of numerical instability strongly suggest that at some point, the fluid would prefer to crystallize near the hydrophilic portion of the surface.

However, a true crystallization would require symmetry breaking in the z-direction, and as a result a 3D DFT calculation would be needed for a complete prediction of such crystallization.

Given density profiles and surface free energies for both surface-vapor and surface-liquid interfaces, it is straightforward to calculate the contact angles. Calculating the contact angles for a wide variety of surface stripe widths and hydrophilic interaction parameters leads to the results in Fig.VC.4. This result shows that a chemically heterogeneous surface with stripes of small size is more easily wet than a chemically homogeneous surface. In contrast, when the stripes are larger than approximately $l=2\sigma$, significant negative deviations from the homogeneous wall results are found. This result is a consequence of the presence of large hydrophobic patches on the surface. For the cases $l=5\sigma$ and $l=20\sigma$, the contact angle data stop where the solutions became numerically unstable due to the appearance of the sharply peaked profiles as shown in Fig.VC.3.

Finally, by investigating the behavior of the adsorption and surface free energy as well as looking for multiple solutions in phase space, both the wetting transition and some local layering transitions were found. These transitions along with the lines of numerical instability are indicated in Fig.VC.5. The wetting transition corresponds to a transition between the partial wetting regime and a complete wetting regime. At complete wetting, a thick liquid film forms

over the entire surface when the wall is in contact with a vapor. The local layering transitions are what remains of the wetting transitions when the surface heterogeneities become large. In these cases, the fluid molecules near the center of the hydrophilic domains have no idea that the hydrophobic regions are present on the surface. As a result, when the wall-fluid interaction strength gets large enough, these molecules try to go through the wetting transition, but only succeed in increasing the densities discontinuously one layer at a time in close proximity to the center of the hydrophilic stripe. These local layering transitions arise from the local condensed phases like that in the right side of Fig.VC.2.

The lines of numerical instability (dashed lines) in Fig.VC.5 are **not** true phase transitions. In order to properly locate these phase transitions, a 3D calculation is required. However, these

lines do serve a purpose in an operational sense with respect to soft-lithography or other self-assembly/templating processes. The region to the right of these lines indicates where one could expect to form local crystalline structures on a patterned surface. It also indicates that there is some minimum pattern size below which local crystallization (or soft-lithography) will not be possible. The local crystallization will ultimately be limited by collective adsorption phenomena that will lead to global wetting rather than local crystallization. This minimum pattern length is predicted to be approximately $l=5\sigma$, and so its absolute magnitude depends on the solvent of interest.

These studies are the first systematic studies of chemically heterogeneous surfaces that apply a detailed molecular density functional theory. They have provided a greatly improved picture of the structure of solvation layers near chemically heterogeneous surfaces, new insight into the effect of chemical heterogeneity on wetting transitions, and operational guidelines that can be applied to self-assembled templating processes such as soft-lithography. Ultimately, these calculations provide a basis for a more quantitative understanding of the molecular effect that control self-assembly. More details of the calculations can be found in [4].

Section VD. Forces between Polyelectrolytes in Solution

The final example we present involves both 2D calculations and electrolyte fluids. The DFT method has several advantages over molecular dynamics or Monte Carlo approaches for electrolytes. Specifically, simulations of electrolyte systems are limited by the time scales needed to equilibrate small concentration components (the ions). This limitation is severe if the solvent is treated as anything other than a continuum. In addition, while molecular simulations treat the electrostatic forces as long-range effects, the DFT approach utilizes a direct solution of Poisson's equation. As noted earlier, the Poisson solve is the easier part of the problem to solve, and so adds very little extra computational expense either in operations or memory requirements.

The case presented here involves the interaction of polyelectrolyte chain molecules in solution. The model polyelectrolyte is assumed to consist of uniformly charged cylinders, and the forces between two of these charged cylinders is calculated for a variety of model electrolytes. This work is inspired by osmotic stress experiments on DNA. In these experiments, the DNA self-assembles into hexagonally close packed arrays of parallel strands. By applying an osmotic stress to the solution, the solvation of the DNA strands can be altered. Using X-ray scattering, the separation of the DNA strands can be measured [15]. Thus the force between the macromolecules as function of surface separation can be measured. More generally, polyelectrolytes are important industrial materials with a wide variety of applications.

The goal of this work was to determine whether solvation packing plays an important role in surface forces when the surface geometry is curved rather than flat. Most of the recent calculations on forces in polyelectrolyte systems focus on models where the solvent is treated as a continuum. This allows for tractable molecular simulations of only the ions; however, it is a rather crude approximation of an electrolyte. The parameters of our model calculations are taken to mimic the DNA example. Specifically, the diameter of the cylinders is 3σ (where $\sigma=4.25\text{\AA}$) and the surface charge per unit area is $q\sigma^2/e=0.17$. Results for both 1:1 and 2:2 electrolytes are presented. The electrolyte is assumed to be composed of ions and solvent particles of identical size. Correlations between electrostatics and hard sphere terms are included via the Δc term discussed in Section II.

The computational domain of the calculations was sketched earlier in Fig.II.6. The grid density for these calculations is 0.1σ in both dimensions. Both the right and top boundaries are reflective while the left and bottom boundaries are set to bulk. Clearly if the domain is reflected in both x and y , the total calculation involves a pair of cylinders isolated in solution. Three model electrolytes were considered. They included a three component model (3CM) that has finite ions and solvent molecules, a two component model (2CM) with finite sized ions in a continuum solvent, and a Poisson-Boltzmann (PB) model that consists of point charges in a continuum solvent. Comparing density profiles for these models, the 3CM has the most structure as would be expected, and the PB and 2CM distributions are similar except near the cylinder surfaces where there is an exclusion zone in the 2CM due to the finite ion size.

Surface forces were calculated as described in Section II. The forces between the cylinders in both 1:1 and 2:2 PB and 2CM electrolytes are shown in Fig.VD.1 while the 3CM is compared to the PB model in Fig.VD.2. The ion densities associated with the 1:1 electrolytes are $\rho\sigma^3=0.05$ (solid lines) and $\rho\sigma^3=0.005$ (dashed lines) while the ion densities associated with the 2:2 electrolytes are $\rho\sigma^3=0.01$ (solid lines) and $\rho\sigma^3=0.001$ (dashed lines).

Fig. VD.1 The total force per unit length between charged parallel cylinders as a function of their surface separation, D . PB results are monotonic while 2CM results display complex short range behavior.

Fig. VD.2 The total force per unit length between charged parallel cylinders as a function of their surface separation, D . PB results are monotonic while 3CM results are oscillatory. The 3CM curves are cubic spline fits to the calculated data points.

The PB forces are all monotonically repulsive while the 3CM results are oscillatory. All of the 3CM force curves display both positive and negative segments with multiple crossings of the line of mechanical stability (zero force). In the 2CM, only repulsive forces are found for 1:1 electrolytes while each of the 2:2 electrolytes display one attractive segment. The attractive minimum in the 2:2 2CM has a shallower, and wider bowl than the minima in the 3CM case.

At very short distances ($<1.5\sigma$), a strong negative (attractive) force is found in the 3CM. This attractive force is an example of osmotic depletion. At these separations, the fluid molecules cannot fit in the gap between the cylinders, and a net attraction of the particles is the consequence. Since this attraction increases to zero separation, this force may ultimately be the portion of the solvation force curve that controls agglomeration (irreversible sticking) of particles. Agglomeration is an important industrial problem when one is processing or pumping mixtures containing colloidal particles. This osmotic depletion effect is apparently not as strong as the electrostatic repulsion in the 2CM where additional structure is observed at short separations, but the total force remains repulsive.

Apart from agglomeration, *reversible* assembly in solution requires attractive forces, and clearly all of the 3CM cases as well as the 2:2 2CM cases have attractive force segments. However, it is a necessary but not sufficient condition that the force to be attractive in order for assembly to occur. The definitive test lies instead in the behavior of the potential of mean force (PMF). Potentially stable surface separations will correspond to the local (negative) minima in the PMF. These PMFs are shown in Fig.VD.5 for the 2CM and the PB model while Fig.VD.6 shows the 3CM and PB model.

The grid used in these calculations has the same density as that used in the sum rule calculation in Section II.F. In that section, it is discussed that while the force sum rule is accurate when $D/\sigma > 1$, it is problematic for smaller surface separations. As a results, the 2CM figures are based on a direct calculation of the free energy, while the 3CM results use the integrated force for $D/\sigma > 1$, and show the results of the direct calculation for smaller separations.

Fig.VD.3 shows that attractions between the cylinders can be expected for the 2CM when they are immersed in 2:2 electrolytes. This attraction is due mostly to counterion condensation in the gap between the cylinders. Fig.VD.4 demonstrates that the 3CM also has a deep minima at short range. The osmotic depletion effect is stronger for the 3CM than the 2CM due to the higher number density of fluid molecules in the calculation. In addition to counter-ion condensation and osmotic depletion, the 3CM also exhibits solvation (solvent packing) effects. The solvation structure results both in more complex short range behavior (see 1:1 electrolyte results in Fig.VD.4, and multiple local free energy minima. These multiple local free energy minima provide a basis for observing reversible transitions between different solvation states as is observed in osmotic stress/DNA experiments.

These results demonstrate, that electrostatics and solvent packing are coupled in a nontrivial way even when the surfaces are curved and have dimensions comparable to polymer chain molecules such as DNA. Thus, models that treat the solvent as a continuum are completely inadequate for understanding self-assembly of polyelectrolytes. Of course the models presented here are missing some of the physics of electrolyte solutions as well. Specifically, neither hydrogen bonding nor solvent polarizability have been included in the calculations presented here. In addition, the correlations between hard sphere and electrostatic effects are treated rather crudely. A complete understanding of self-assembly in these complex systems will require assessment of the importance of these other effects. A manuscript on these results is in preparation.

VI. Users Guide

This section is meant to guide the new user through the input files needed to run TRAMONTO. Executables are currently available for several workstation architectures including Suns, DEC Alphas, and SGIs as well as for the massively parallel architectures of the T-flops machine and the Intel Paragon. A description of the primary input files is followed by a description of the output files. This document contains instructions as of November 1998; input files are subject to change as the code matures. Some degree of error checking for conflicting inputs is built into the code; however, a complete cross check of parameters is not guaranteed. Therefore, the analyst is encouraged to double check input files before submitting jobs. Currently funds are not available for extensive support services for TRAMONTO. With this caveat, questions, suggestions for enhancing capabilities, or comments about possible bugs should be directed to either Laura Frink (lfrink@cs.sandia.gov) or Andrew Salinger (agsalin@cs.sandia.gov).

VI.A Input Files

Ex 1: `dft_surface.dat` for 3 surfaces of two types in 1D located at $x=-10.0\sigma$, $x=0.0\sigma$, and $x=10.0\sigma$.

0	-10.0
0	0.0
1	10.0

Ex 2: `dft_surface.dat` for 1 surface in 2D located at $x=0.0$, $y=5.0$.

0	0.0	5.0
---	-----	-----

Ex 3: `dft_surface.dat` for 3 surfaces (e.g. atoms) of different types in 3D.

0	-2.0	0.0	0.0
2	0.0	2.0	0.0
1	0.0	0.0	2.0

Fig.VI.A.1: Three example `dft_surfaces.dat` files for 1D, 2D, and 3D calculations

There are two primary input files for TRAMONTO. The first contains the locations of the centers of the surfaces of interest. This data must be located in a file called "`dft_surfaces.dat`". The first column should indicate the type of the surface of interest, the remaining columns give the position of the center of the surface of interest. The number of position coordinates that should be found in this file is equal to the number of dimensions in the calculation. The number of surfaces in the calculation is found automatically based on how many lines are present in the `dft_surfaces.dat` file. If extraneous columns of data are included, the code will bomb or produce nonsense. Unique surface types are distinguished by any property from surface shape to interaction strengths. The array of surface types starts with zero (C conventions). Example `dft_surfaces.dat` files are shown in Fig.VI.A.1.

The second data file is called "`dft_input.dat`", and it contains all of the remaining input needed for a given run. Each line on which data is located begins with an @ symbol signaling the code to begin reading data. Numerous comments are included in the `dft_input.dat` file. These comments are meant to guide the analyst as he/she sets up the input file. The input data is split into different categories as described in the following sections. The example input file shown below is for a calculation of a 3 component electrolyte in a charged square channel. After the "`dft_input.dat`" file is read in, it is written out to the file "`dft_out.lis`". When the code isn't behaving properly, first check that the code has read the input file as expected.

Note that the user does not need to make any modifications to the input file to run in parallel, not even the number of processors to run on. The code determines the number of processors with an MPI call and automatically partitions the mesh.

VI.A.1 Mesh Parameters

These parameters are used to set up the basic variables controlling the computational domain. Fig.VI.A.2 shows an example from an input file for a 3D calculation. Each parameter is described in more detail below.

```
***** MESH PARAMETERS *****
@ 3          Ndim
@ 4.0 4.0 1.0 Size_x(idim); idim=1,Ndim
@ 0.2 0.2 0.2 Esize_x(idim): idim=1,Ndim
@ 0          Lmesh_refine (0=no,1=yes: auto mesh refinement)
@ -1 -1      Type_bc(x0: left,right) (-1=wall,0=bulk,1=pbk,2=ref)
@ -1 -1      Type_bc(x1: down,up) (-1=wall,0=bulk,1=pbk,2=ref)
@ 1 1        Type_bc(x2: back,front) (-1=wall,0=bulk,1=pbk,2=ref)
*****
```

Fig.VI.A.2: Sample input for Mesh Parameters.

Ndim: Indicates the number of dimensions (1,2, or 3) in the calculation. In all cases, the calculation will be performed for a 3-dimensional fluid; however, in some cases, symmetries allow for analytical integration in some dimensions. For example, for a fluid confined by planar walls, the fluid density only varies perpendicular to the wall allowing for a 1-dimensional calculation.

Size_x[Ndim]: This array contains the total domain size in each of the dimensions of the calculation. The input should be given in dimensionless units of σ , where σ is an arbitrary characteristic length most often chosen to be equal to the size of a fluid molecule of the first component.

Esize_x[Ndim]: This array contains the grid size in each dimension of the calculation. In cases where several zones (see Section VI.A.11) will be applied, this field contains the density of the mesh in the zone with the highest density. Typically Esize_x ranges from 0.01σ to 0.2σ depending on the desired accuracy, dimensionality of the problem, and the available compute time.

Lmesh_refine: This logical (0=False, 1=True) switches on mesh refinement. The code will automatically begin calculations with a mesh of size 0.2σ and then use the resulting solution for an initial guess with the more refined mesh. This may produce improved performance with respect to a poor initial guess for the most refined mesh.

Type_bc[Ndim][2]: This array contains the boundary conditions for the domain. Values in Type_bc[idim][0] give the boundary conditions on the left (x1), bottom (x2), and back (x3) boundaries while Type_bc[idim][1] give the boundary conditions on the right (x1), top (x2), and front (x3) boundaries. Choices for boundary conditions, the flag value, and the treatment of densities and the electric potential at the domain boundaries are indicated in Table VI.A.1.

Boundary Type	Flag	Condition on ρ	Condition on ϕ
In Wall	-1	$\rho=0$	$\phi=0$
In Bulk	0	$\rho=\rho_b$	$\phi=0$
Periodic	1	$\rho(x=N_x+1)=\rho(x=0)$	$\phi(x=N_x+1)=\phi(x=0)$
Reflective	2	$\rho(x=N_x+1)=\rho(x=N_x-1)$	$\phi(x=N_x+1)=\phi(x=N_x-1)$

Table VI.A.1: Boundary conditions for the edges of the computational domain.

VI.A.2 Surface Parmeters

These parameters are used to set up the geometric variables for the surfaces of interest.

Fig.VI.A.3 shows the example of the square channel where the channel is created from two types of planar surfaces. The two types have different orientations. Each parameter is described in more detail below.

```
***** SURFACE PARAMETERS *****
@ 2      Nwall_type
@ 0 0     Surf_type[iwall_type]; iwall_type=0,Nwall_type
@ 0 1     Orientation[iwall_type]; iwall_type=0,Nwall_type
@ 1.0 1.0 WallParam[iwall_type]; iwall_type=0,Nwall_type
@ 1.0 1.0 WallParam2[iwall_type]; iwall_type=0,Nwall_type
@ 1.0 1.0 WallParam3[iwall_type]; iwall_type=0,Nwall_type
*****
```

Fig.VI.A.3: Sample input for Surface Parameters.

Nwall_type: This is the number of different types of surfaces in the system. A surface has a different type if it has any unique characteristics (Orientation, strength of interactions, shape, surface charge, etc.)

Surf_type[Nwall_type]: This array stores the basic shapes of the surfaces. Several choices are currently possible, and addition of new surfaces in the code is straightforward. The current surface types are: infinite planar surfaces, finite planar surfaces, bumpy surfaces, colloidal spheres (3D) or cylinders (2D), spherical cavities (3D) or infinite cylindrical pores (2D), spherical (3D) or cylindrical (2D) atoms, finite length cylinder (3D) or slit (2D) pores, finite length cone shaped cylindrical (3D) or slit (2D) pores.

Orientation[Nwall_type]: This parameter identifies the orientation of the surface where appropriate. If no orientation is needed, any number may be present in the input file. For infinite planar surfaces or bumpy surfaces, the Orientation gives the dimension perpendicular to the surface. For cylindrical or conical pores, the Orientation gives the axial direction of the pore. For all other surfaces, the Orientation is irrelevant.

Wall_Param[Nwall_type], Wall_Param2[Nwall_type], Wall_Param3[Nwall_type]: These arrays store the parameters needed to describe a given surface type. The parameters are given in Table VI.A.2. Each surface type must have an entry for each type of Wall_Param although some wall types only need one parameter. In these cases any number may appear in the list since these numbers are never used. It is important to note that TRAMONTO needs to read in the arrays in their entirety so entries may not be omitted from the input file even if those entries are meaningless. Also note that when surfaces are taken to be atoms (Surf_type=5), none of the Wall_Param arrays are used. Rather, the diameter of the surface atoms is taken from the list of Sigma_w in the Surface Particle Parameters (see Section VI.A.5).

Surf_type	Flag	Wall_Param	Wall_Param2	Wall_param3
Planar Wall	0	Wall thickness		
Finite Planar Wall	1	Wall thickness: x0	Wall thickness: x1	Wall thickness: x2
Bumpy Wall	2	Radius of Bumps		
Colloids cyl/sphere	3	Cyl./Sphere Radius		
Cyl.Pore/Sphere Cavity	4	Pore/Cavity Radius		
Atoms cyl/sphere	5	-- NOT USED ---		
Finite Cyl.Pore	6	Pore Radius	Pore Length	
Finite Cone pore	7	Cone Radius (Left/Bottom/Back)	Cone Radius (Right/Top/Front)	Cone Length

Table VI.A.2: The Wall_Param arrays expected for each surface type.

VI.A.3 Potential Type Parameters

These switches are used to indicate which types of potential models will be used in a given calculation. Fig.VI.A.4 shows the example of an electrolyte in the charged square channel. Each parameter is described in more detail below.

```

***** POTENTIAL TYPE SELECTIONS *****
@ 1      Ipot_ff_n (0=Ideal Gas, 1=Hard Spheres, 2=LJ12_6)
@ 1      Ipot_wf_n (0=No wall-fluid, 1=Hard Wall, 2=LJ9_3
                3=normalized LJ9_3, 4=LJ12_6)
@ 1      Ipot_ff_c (0=No Charges, 1=Coulomb)
@ 1      Ipot_wf_c (0=No Charges, 1=Coulomb)
@ 1 1 0 0 Sten_Type_TF[isten] (isten=1,Nsten) 0=off, 1=on
*****

```

Fig.VI.A.4: Sample input for Potential Type Parameters.

Ipot_ff_n: This switch indicates the type of neutral interactions between fluid particles. The options are: ideal gas (0), hard spheres (1), 12-6 Lennard-Jones (2).

Ipot_wf_n: This switch indicates the type of neutral interactions between the fluid particles and the surfaces. The options are: none (0), hard-sphere/hard-wall (1), Lennard-Jones 9-3 (2), renormalized Lennard-Jones 9-3 (3), numerically integrated 12-6 Lennard-Jones (4).

Ipot_ff_c: This switch indicates the type of charged interactions between fluid particles. The options are: no charges (0), coulomb charges (1).

Ipot_wf_c: This switch indicates the type of charged interactions between the fluid particles and the surfaces. The options are: no charged surfaces (0), coulomb (1).

Sten_type_TF[]: This array of switches is set to true (1) only for the stencils needed for the calculation. The stencil types are Dirac delta function (Sten_type[0]), Heaviside step function (Sten_type[1]), Heaviside step function with weights by attractions (Sten_type[2]), and Heaviside step function with weights of the Δc function (Sten_type[3]). All entries should be set to false (0) for ideal gas or Poisson-Boltzmann electrolyte calculations.

VI.A.4 Fluid Particle Parameters

These parameters describe the characteristics of the fluid model used. Fig.VI.A.5 shows the example of the three component electrolyte. Each parameter is described in more detail below.

```
***** FLUID PARTICLE PARAMETERS *****
@ 3      Ncomp
***** IF Ipot_ff_n>0 *****
@ 1. 1. 1.  Sigma_ff(icom,icom): icomp=1,Ncomp
***** IF Ipot_ff_n>1 *****
@ 1. 1. 1.  Eps_ff(icom,icom): icomp=1,Ncomp
@ 1. 1. 1.  Cut_ff(icom,icom): icomp=1,Ncomp
***** IF Ipot_ff_c>0 *****
@ 0. 1. -1. Charge_f(icom): icomp=1,Ncomp
*****
```

Fig.VI.A.5: Sample input for Fluid Particle Parameters.

Ncomp: Number of components.

Note: The following 3 lines ask for just the Ncomp diagonal components of the matrices.

Sigma_ff[Ncomp][Ncomp]: The size of the fluid atoms (in units of an arbitrary characteristic length σ). Unless otherwise indicated (see Mixing Rules in Section VI.A.7), the cross terms are taken to be $\sigma_{ij} = 0.5*(\sigma_i + \sigma_j)$. These parameters are only read in if the fluid is *not* and ideal gas or Poisson-Boltzmann electrolyte.

Eps_ff[Ncomp][Ncomp]: The interaction energy parameter associated with the fluid-fluid interactions in units of an arbitrary energy scale, ϵ , that is most often set equal to $Eps_ff[0]$. Unless otherwise indicated (see Mixing Rules below), the cross terms are taken to be $\epsilon_{ij} = (\epsilon_{ii} * \epsilon_{jj})^{0.5}$. Interaction energy parameters are only read in if the fluid is a Lennard-Jones fluid.

Cut_ff[Ncomp][Ncomp]: The cut-off length for the fluid-fluid interactions in units of an arbitrary characteristic length σ . The cross terms are taken to be $r_{ij}^{cut} = r_{ii}^{cut} \sigma_{ij}$. Cut-offs are only read in if the fluid is a Lennard-Jones fluid.

Charge_f[Ncomp]: The valence associated with each component. Charges are only read in if the fluid is Coulombic.

VI.A.5 Surface Particle Parameters

These parameters describe the characteristics of the particles that make up the surfaces of interest. Fig.VI.A.6 shows a typical example of the input. Each parameter is described in more detail below.

```

***** SURFACE PARTICLE PARAMETERS *****
***** IF Ipot_wf_n>0 *****
@ 1. 1.      Sigma_w(iwall_type): iwall_type=1,Nwall_type
***** IF Ipot_ff_n>1 *****
@ 1. 1.      Eps_w(iwall_type): iwall_type=1,Nwall_type

@ 1. 1.      Rho_s(iwall_type): iwall_type=1,Nwall_type
*****

```

Fig.VI.A.6: Sample input for Surface Particle Parameters.

Sigma_w[Nwall_type]: The characteristic size of a wall atom in units of the characteristic length, σ

Eps_w[Nwall_type]: The interaction energy parameter associated with wall-wall interactions in units of the characteristic energy, ϵ .

Rho_s[Nwall_type]: The density of the solid surface of interest entered as $\rho_s \sigma^3$.

Note that wall-fluid interactions are found using the Lorentz-Berthelot mixing rules unless indicated otherwise in the Mixing Rule section (see below). The Lorentz-Berthelot mixing rules say that the characteristic interaction length for a wall with a fluid particle will be $\sigma_{wf} = 0.5(\sigma_w + \sigma_{ff})$ and the characteristic energy parameter will be $\epsilon_{wf} = (\epsilon_w * \epsilon_{ff})^{0.5}$.*

VI.A.6 Charged Surface Boundary Conditions

These parameters define the boundary conditions applied at the surface-fluid interfaces when charged surfaces are present in the system ($Ipot_wf_c > 0$). Fig.VI.A.7 shows the example of the charged square channel. Each parameter is described in more detail below.

```

***** CHARGED SURFACE BOUNDARY CONDITIONS *****
@ 2 2      Type_bc(iwall_type): iwall_type=1,Nwall_type
@ .05 .05  Elec_param_w(iwall_type): iwall_type=1,Nwall_type
@ 0        Nlocal_charge:

.1 1. -5. 0. 0.  Charge[i], Charge_Diam[i], Charge_x[i][idim]

@ 0        Lpoint_charge: (0=false,1=true)
*****

```

Fig.VI.A.7: Sample input for Charged Surface Boundary Condition Parameters.

Type_bc_elec[Nwall_type]: The type of boundary condition for charged systems. Possible choices and the associated flags are: no charge (0), constant potential (1), constant surface charge (2), and charged atoms (3).

Elec_param_w[Nwall_type]: This parameter stores the value of the electrostatic boundary condition of choice. This may be the surface potential (given as $e\phi/kT$), the surface charge (given as $q\sigma^2/e$), or the charge associated (in a volumetric sense) with each of the atom types that make

up the surface (given as Q/e) where e is the magnitude of the unit charge. If a surface is neutral, enter 0.0 for the Elec_param_w[iwall_type].

Nlocal_charge: The number of local volumetric charges that exist within the computational domain. These charges may or may not be located within boundaries of surfaces from which solvent is precluded.

Charge_loc[Nloc_charge]: This array stores the total charge associated with each site of local charge.

Charge_diam[Nloc_charge]: This array stores the diameter over which a given local charge should be spread. If Charge_diam is set to zero, all of the local charge will be put in one element of the domain.

Charge_x[Nloc_charge][Ndim]: This array stores the positions of the centers of the local volumetric sources of charge.

Note: A single line in dft_input.dat contains all the Charge_loc, Charge_diam, and Charge_x information for a given local source of charge. So, there should be a line present for each volumetric charge site in the system (ie. Nlocal_charge total lines). If Nlocal_charge=0, the @ symbol should be removed from any/all lines that contain Charge_loc, Charge_diam, and Charge_x information.

Lpoint_charge: This logical indicates whether or not the volumetric charges (either in Nlocal_charge or when Type_bc_elec=3) are meant to be point charges located entirely within one element. If this is false (set to 0), the charges will be smeared over all the elements within either Charge_diam (if Nlocal_charge > 0) or Sigma_ff[iwall_type] (if Type_bc_elec=3).

VI.A.7 Mixing Rules

This section allows the analyst to define mixing rules other than the standard Lorentz-Berthelot rules by manually inputting in entirety the interaction parameter arrays. Fig.VI.A.8 shows an example (not related to the square channel problem). Except for Lmix_rule, all the parameters were described earlier. Lmix_rule is simply a switch that indicates Lorentz-Berthelot rules (flag=0) or manual input (flag=1). The example shown below gives input for a system with two fluid species and two wall types.

```
***** MIXING RULE PARAMETERS *****
@ 1                               Lmix_rule: (0=L-B, 1=manual input)

@ 1. .5 .5 .6      Sigma_ff[icomp][jcomp]:order of input is:
                        [0][0],...,[0][Ncomp-1]; [1][0],...,[1][Ncomp-1]...;
                        [Ncomp-1][0],...[Ncomp-1][Ncomp-1]

@ 1. 2. 2. 3.      Eps_ff[icomp][jcomp]:
@ 2.5 2.5 2.5 2.5 Cut_ff[icomp][jcomp]:
@ 1. .5 .5 .6      Sigma_wf[icomp][jcomp]:
@ 1. 2. 2. 3.      Eps_wf[icomp][jcomp]:
@ 2.5 2.5 2.5 2.5 Cut_wf[icomp][jcomp]:
*****
```

Fig.VI.A.8: Sample input for Mixing Rule Parameters.

VI.A.8 State Point Parameters

These parameters define the thermodynamic state point at which the calculations will be performed. Fig.VI.A.9 shows the example of the electrolyte fluid in the charged square channel. Each parameter is described in more detail below.

```
***** STATE POINT PARAMETERS *****
@ 1.0          Temp: kT/eps_ff[0][0] -- LJ fluid
                =1.0 -- HS fluid
@ 0.7          Temp_elec: Coulombic systems only
                =kT eps*eps_o*sigma_f[0]*4*pi/(e*e)
                where eps is dielectric constant
@ 0.6 0.05 0.05 Rho_b[icompl];icompl=1,Ncomp
*****
```

Fig.VI.A.9: Sample input for State Point Parameters.

Temp: The temperature of the system in units of kT/ϵ where ϵ is an arbitrary energy scale. This temperature scale is used for Lennard-Jones fluids. For neutral hard sphere fluids, the temperature is irrelevant so set equal to 1.0. For Coulombic systems with point charges or hard cores, use the Temp_elec param to set the energy scale, and set Temp=1.0.

Temp_elec: This is the relevant temperature scale for electrolyte calculations: $4\pi kT\epsilon\epsilon_0\sigma/e^2$, where ϵ is the dielectric constant, ϵ_0 is the permittivity of free space, e is the unit charge, and σ is the characteristic length.

Rho_b[Ncomp]: The number density of each component in the bulk fluid given as $\rho\sigma^3$.

VI.A.9 Run Type/Guess Type Parameters

These parameters define the type of calculation to be run along with the type of initial guess that will be applied. Fig.VI.A.10 shows an example. Each parameter is described in more detail below.

```
***** RUN TYPE/GUESS PARAMETERS *****
@ -1          Iliq_vap (-1=none,1=W-V,2=W-L,3=L-V profiles)
@ -3 8.0      Igness, Thickness
                (Igness flags ....
                -3,-2,-1: CONST_RHO,CONST_RHO_L,CONST_RHO_V
                0,1,2  : EXP_RHO,EXP_RHO_L,EXP_RHO_V
                3,4,5  : STEP_RHO, STEP_RHO_L, STEP_RHO_V
                6      : STEP_LV)
@ 0           Restart
*****
```

Fig.VI.A.10: Sample input for Run Type/Guess Type Parameters.

Iliq_vap: Indicates whether this run is to use the state point parameters given above (Section VI.A.8) or whether the profile should be set up for a fluid at liquid-vapor coexistence. In the latter case, the Temp parameter is used to first find coexistence, and then the bulk densities/chemical potentials are set to coexistence values. The choices are: off coexistence(-1),

wall-vapor (1), wall-liquid (2), liquid-vapor(3). This parameter is particularly useful when performing wetting studies or calculating contact angles as in Section V.C. Liquid-vapor coexistence can only be calculated at this time for Lennard-Jones fluids of one component.

Iguess: This parameter sets the initial guess type for cases where there is no restart file. There are many options, the best of which depends on the type of run being performed. The possibilities are:

- constant density with $\rho=\rho_b$ (-3), $\rho=\rho_{\text{coex}}^{\text{liq}}$ (-2), or $\rho=\rho_{\text{coex}}^{\text{vap}}$ (-1),
- ideal gas profile with $\rho=\rho_b e^{(-V_{\text{ext}}/kT)}$ (0), $\rho=\rho_{\text{coex}}^{\text{liq}} e^{(-V_{\text{ext}}/kT)}$ (1), or $\rho=\rho_{\text{coex}}^{\text{vap}} e^{(-V_{\text{ext}}/kT)}$ (2),
- stepped profile with discontinuities at a distance given by *Thickness* away from the surface. The density to which the profile is stepped far from the surface may be ρ_b (3), $\rho_{\text{coex}}^{\text{liq}}$ (4), or $\rho_{\text{coex}}^{\text{vap}}$ (5),
- stepped profile between $\rho_{\text{coex}}^{\text{liq}}$ and $\rho_{\text{coex}}^{\text{vap}}$ (6), Use this option for liquid-vapor profiles.

Thickness: This parameter sets the distance away from the surfaces at which the profile will be stepped. It will work with a restart file as well as one of the guess types described above.

Restart: This switch controls whether the run will start from a new initial guess or from an old density file (dft_dens.dat). Options are:

- No restart file - generate best possible initial guess flag = 0
- Use the restart file as is flag = 1
- Use restart file, but step the profile at Thickness as indicated by Iguess flag = 2

VI.A.10 Automatic Continuation Parameters

In many cases, multiple solutions will be desired (e.g. adsorption isotherms or solvation force curves). In order to facilitate generating multiple data sets, and storing them in multiple data files, continuation can be performed in up to two fields. The output may be sent either to one or to multiple files. Fig.VI.A.11 shows an example of input for continuation in both mesh size and fluid density in a three component fluid. The parameters are described in more detail below.

```
***** AUTO CONTINUATION PARAMETERS *****
@ 5 0      N_runs  Cont_type_1
@ 0 0      Plane_new_nodes  Pos_new_nodes
@ 0.1      Del_1[]
@ 5 2      N_runs2  Cont_type_2
@ 0. 0.01 0.01  Del_2[]
@ 1        Print_force_type
@ 0        Print_rho_type
@ 0 0      Print_rho_switch  Print_force_switch
@ 1        Print_header
@ 3        Iwrite
*****
```

Fig.VI.A.11: Sample input for Automated Continuation Parameters.

N_runs: The number of data sets in the first continuation field. Set N_runs=1 if only one solution is desired.

Cont_type_1: The type of continuation in the first continuation field. Continuation options are:

- Change mesh size: flag=0

- Temperature: flag=1
- Densities; Rho_b[Ncomp]: flag=2
- Wall-Interactions, Eps_w[Nwall_type]: flag=3
- Cut-offs; Cut_ff[Ncomp]: flag=4
- Elec_Param_w[Nwall_type]: flag=5

Plane_new_nodes: If Cont_type_1=0, continuation involves changing the mesh size. In this case, the orientation of the new plane of nodes must be indicated. The options are: yz-plane (0), xz-plane (1), and xy-plane (2).

Pos_new_nodes: Again, if Cont_type_1=0 (Mesh continuation), the position of the new plane(s) of nodes must be indicated. The options are: left-bottom-back (-1), center of box (0), and right-top-front (1).

Del_1[]: This array stores the amount the Cont_type_1 parameter(s) will be varied. This variable is an array because several of the continuation fields are vector fields (e.g. Rho_b[Ncomp], Eps_w[Nwall_type], Cut_ff[Ncomp], Elec_param_w[Nwall_type]). In these cases continuation may be done with respect to any desired number of components or wall types. In the example in Fig.VI.A.10, continuation is performed in the densities of components 1 and 2, but not component 0.

N_runs2: The number of data sets in the second continuation field. Set N_runs2=1 if only one solution is desired or if continuation in only one field is desired.

Cont_type_2: The type of continuation in the second continuation field. See list under Cont_type_1 for the options. Note that continuation in the mesh (flag=0) cannot be done from the second field.

Del_2[]: This array stores the amount the Cont_type_2 parameter(s) will be varied.

Print_force_type: This switch sets the way the basic force data set (described below in the section on output files) output will be printed. The options are to:

- Put all output in the file *dft_force.dat* : flag=0
- Separate output where each file has a constant value of the Cont_type_1 parameter: flag=1
- Separate output where each file has a constant value of the Cont_type_2 parameter: flag=2

Print_rho_type: This switch sets the way the solution data (the densities) will be written to files. The options are to:

- Put all output in *dft_dens.dat*: flag=0
- Put the output from each run in a different file: flag=1

Note that if continuation is performed and Print_rho_type is set to 0, the *dft_dens.dat* file will be overwritten at each new solution.

Print_rho_switch: This switch determines how densities will be printed in the *dft_force.dat* file. Depending on the type of run being performed, different renditions of the densities may be useful. The options are:

- All densities as $\rho\sigma^3$: flag=0
- The value of p/p_0 : applicable only to a 1 component LJ fluid: flag=1
- The debye screening length κ^{-1} : applicable to electrolytes: flag=2

- The chemical potentials (μ/kT): flag=3

Print_mesh_switch: This switch sets how the mesh will be represented in the file *dft_force.dat*. The options are to print the surface separations between all pairs of surfaces in the domain (flag=0) or to print the surface positions at the center of each surface (flag=1).

Print_header: This logical indicates (0=false, 1=true) whether to print a header in the *dft_force.dat* file(s). If a run is going to start from an old file and add to an existing force file, it might not be desirable to print the header.

Iwrite: This switch controls how much output will be printed. For minimal output (no density profiles) enter 0; for minimal output plus density profiles enter 1; for avs compatible output files enter 2; for verbose printing enter 3. The files printed in each case are detailed below in the section on output files.

VI.A.11 Coarsening Switches

These switches allow for the nonuniform meshes, coarsened Jacobians, and approximate Jacobians discussed in Sections III and IV of this report. Fig.VI.A.12 shows an example of the switch options in *dft_input.dat*. The parameters are described in more detail below.

```
***** COARSENING SWITCHES *****
@ 2      Nzone
@ 1.5    Rmax_zone[Nzone-1]
@ 1      Coarsen Residual ? (0=NO, 1=YES)
@ 1      Coarser_jac; 5 possible options
@ 0      Matrix_fill_flag (0=Exact Jac,1=Jac_Save1,2=Jac_Save2)
*****
```

Fig.VI.A.12: Sample input for Switches controlling various forms of mesh coarsening, Jacobian coarsening, and Jacobian approximation.

Nzone: The number of mesh zones in the calculation. The zones are defined by the distance from any surface, and each successive mesh is coarser than the most dense by the next power of 2. So, starting with a mesh of 0.05 near the surfaces, the next three zones will have *Esize_x* values of 0.1, 0.2, and 0.4.

Rmax_zone[Nzone-1]: This array stores the distances that separate the various zones. These distances are measured from the surfaces in the system.

Coarsen Residual: This is a logical for switching on a coarsened residual along with the coarsened Jacobian that results from implementing zones.

Coarser_jac: With this switch, the integrals in the jacobian are coarsened further although the mesh remains unchanged. Several options are:

No further Jacobian coarsening	flag = 0
Coarsen finest Jacobian zone by factor of 2	flag = 1
Coarsen all but coarsest zone by factor of 2	flag = 2
Use coarsest <i>Esize_x</i> everywhere	flag = 3
Use 2 nd coarsest <i>Esize_x</i> everywhere except in coarsest zone	flag = 4

Matrix_fill_flag: This switches between the exact Jacobian (0), the approximate Jacobian (Jac_Save1) option that includes only $w^{(3)}$ and $w^{(2)}$ terms (1), and the approximate Jacobian (Jac_Save2) option that includes all scalar weight functions, $w^{(0)}$ - $w^{(3)}$ (2). As a rule of thumb, the Jac_Save1 options work best if the bulk fluid density is not too high or for hard cores systems.

Note: Quadratic convergence of Newton's method can only be expected when the Jacobian is exact for a given residual, which is when Coarser_jac=0 and Matrix_fill_flag = 0.

VI.A.12 Nonlinear Solver Parameters

These switches set the convergence parameters for Newton iterations. Fig.VI.A.13 shows an example of the options in *dft_input.dat*. The parameters are described in more detail below.

```
***** NON-LINEAR SOLVER PARAMETERS *****
@ 100          Maximum # of Newton Iterations
@ 1.e-3 1.e-8  Relative and Absolute Convergence Tolerances
@ 1           Load_bal_switch (0=linear,1=box,2=weights,3=timings)
*****
```

Fig.VI.A.13: Sample input for nonlinear solver parameters.

Tolerances: The Relative and Absolute Convergence Tolerances, ϵ_{Rel} and ϵ_{Abs} , are used to determine if the update vector in Newton's method is small enough to consider the calculation converged. The convergence criterion is:

$$\frac{1}{N_{unk}} \sum_{i=1}^{N_{unk}} \frac{\Delta x_i}{\epsilon_{Rel} x_i + \epsilon_{Abs}} < 1.0$$

Load_bal_switch: Options currently available include:

- Box (flag = 1) where the domain is divided with near to equal nodes per processor
- Weights (flag = 2) where each node in the domain is given a weight, and the bisection of nodes per processor is done on a weighted basis.

Linear (0) and Timings (3) based load balancing are not available in the current version of the code.

VI.A.13 Linear Solver Parameters

This section sets convergence parameters for iterative linear solver, which are passed to the Aztec linear solver library [16]. Fig.VI.A.14 shows an example. The parameters are described in more

```
***** LINEAR SOLVER PARAMETERS *****
@ 0          Solver
@ 0          Scaling
@ 1          Preconditioner
@ 1.0e-4     Tolerance
*****
```

Fig.VI.A.14: Sample input for linear solver parameters.

detail below.

Solver: Switch setting solver type in Aztec. Options are: gmres(0), conjugate-gradient(1), tfqmr(2), cg2(3), bicgstab(4). The GMRES option is most often used. See an Aztec manual for more information on these options.

Scaling: Switch setting scaling type in Aztec. Options are: row sum scaling (0), Jacobi scaling (1), symrow_sum scaling (2), and none (-1). Generally, scaling is not needed, but row sum or Jacobi scaling may help convergence.

Preconditioner: Switch setting preconditioner in Aztec. Options are: ilu preconditioning (0), Jacobi (1), symGS (2), Lspoly3(3), and none(-1). As a general rule, preconditioning is not needed except for Poisson-Boltzmann systems with dense meshes. In those cases ilu works best. Some convergence improvement can be obtained with ilu or Jacobi preconditioning.

Tolerance: Convergence tolerance for the reduction in the scaled residual for the iterative linear solver.

Section VI.B Output Files

In this section, the output files that TRAMONTO produces are described briefly. In addition to discussing the contents of the file, the IWRITE switch required to produce a given file is noted.

dft out.lis: This file is always printed, and it contains several miscellaneous pieces of information. They include the following in order:

- 1 an echo of the input file
- 2 thermodynamic information about the bulk fluid (pressure, chemical potential etc.)
- 3 domain information: number of nodes/elements in each dimension
- 4 number of elements in the surfaces
- 5 identification of boundary nodes
- 6 surface normals per surface element
- 7 surface areas

dft dens.dat: This file contains the solution vector. The first Ndim columns contain node positions. The second Ncomp columns contain densities of each component, the next column contains the electrostatic potential (if the fluid is an electrolyte). This file is printed if Iwrite>0

dft force.dat: This file contains all output that are calculated based on the solution vectors. The order of the columns is Cont_Param_1 variable(s), Cont_Param_2 variable(s), number of Newton iterations, Time for the run, Charge in the fluid, solvation force(s), surface free energy, excess surface adsorptions for each component. This file is always printed.

dft vext.dat: This file contains the value of the (neutral part) of the external field as a function of position. The first Ndim columns contain x0, x1, and x2 coordinates of a given node, the next Ncomp columns contain the external field (V/kT). This file is only printed when Iwrite=VERBOSE.

dft zeroTF.dat: This file contains the value of the switch that indicates whether a given node should solve $x=0$ rather than the full DFT equations. The first Ndim columns contain x0, x1, and

x2 coordinates of a given node, the next Ncomp columns contain the switch. This file is only printed when Iwrite=VERBOSE.

dft_zones.dat: This file contains the value of the zone number of each node. The first Ndim columns contain x0, x1, and x2 coordinates of a given node, the next columns contains the zone number. This file is only printed when Iwrite=VERBOSE.

rho_init.dat: This file contains the initial guess vector. The first Ndim columns contain node positions. The second Ncomp columns contain densities of each component, the next column contains the electrostatic potential (if the fluid is an electrolyte). This file is printed if Iwrite=VERBOSE

stencil.out: This file contains a listing of the offsets and weights associated with each quadrature stencil in the calculation. The first column is the stencil point index number, the next Ndim columns contain offsets from the origin to the stencil point of interest, the next column contains the aggregate weight at a given node. The final set of columns contains the weight associated with each element that touches a given node. These are needed when hard surface-fluid interactions are present in the system. This file is printed if Iwrite=VERBOSE.

rb.out: This file contains all of the nonlocal densities, $\bar{\rho}$ as a function of node position. The first Ndim columns contain the node coordinates. The next four columns contain the scalar nonlocal densities. The final 2*Ndim columns contain the vector nonlocal densities. This file is printed if Iwrite=VERBOSE.

dphi.out: This file contains all of the derivatives of the hard sphere free energy density as a function of the nonlocal densities, $\partial\Phi/\partial\bar{\rho}$, as a function of node position. The first Ndim columns contain the node coordinates. The next four columns contain the scalar derivatives. The final 2*Ndim columns contain the vector derivatives. This file is printed if Iwrite=VERBOSE.

References

- [1] Douglas Frink, L.J. and F. van Swol, A Molecular Theory for Surface Forces Adhesion Measurements, *J. Chem. Phys.*, v.106, p. 3782 (1997).
- [2] Samuel, J., C.J. Brinker, L.J. Douglas Frink, and F. van Swol, *Direct Measurement of Solvation Forces in Complex Microporous Media: A New Characterization Tool*, *Langmuir*, v.14, p.2602 (1998).
- [3] Douglas Frink, L.J. and F. van Swol, *Solvation Forces Between Rough Surfaces*, *J. Chem. Phys.*, v.108, p. 5588 (1998).
- [4] Douglas Frink, L.J. and A.G. Salinger, *Wetting of a Chemically Heterogeneous Surface*, submitted to *J. Chem. Phys.*, Oct., 1998.
- [5] Henderson, D., ed., *Fundamentals of Inhomogeneous Fluids*, Marcel Dekker, Inc., New York, NY, 1992.
- [6] Rosenfeld, Y., *Phys. Rev. Lett.*, v. 63, p.980 (1989).
- [7] Weeks, J.D., D. Chandler, and H.C. Anderson, *J. Chem. Phys.*, v. 54, p. 5237, (1971).
- [8] Tang, Z. M.-y. Teran, H.T. Davis, L.E. Scriven, and H.S. White, *J. Chem. Phys.*, v.71, p. 369 (1990).
- [9] Douglas Frink, L.J. and F. van Swol, *Oscillatory surface forces: A test of the superposition approximation*, *J. Chem. Phys.*, v. 105, p. 2884 (1996).
- [10] Waisman, E. and J.L. Lebowitz, *J. Chem. Phys.*, v.56, p.3093 (1972).
- [11] Frink, L.J.D. and F. van Swol, *Solvation forces and colloidal stability: A combined Monte Carlo and density functional theory approach*, *J. Chem. Phys.*, v. 100, p. 9106 (1994).
- [12] Reiner, E.S. and C.J. Radke, *J. Chem. Soc. Faraday Trans.*, v. 86, p. 3901 (1990).
- [13] Douglas, L.J., M. Lupkowski, T. Dodd, and F. van Swol, *On the role of solvation forces in colloidal phase transitions*, *Langmuir*, v. 9, p. 1445 (1993).
- [14] van Swol, F. and J.R. Henderson, *J. Chem. Soc. Faraday Trans. 2*, v. 47, p. 4455 (1992).
- [15] Rau, D.C. and V.A. Parsegian, *Biophysical J.*, v. 61, p. 260 (1992).
- [16] Hutchinson, S.A., J.N. Shadid, and R.S. Tuminaro, *Aztec User's Guide*, Sandia National Labs Technical Report, SAND95-1559 (1995).

UNLIMITED RELEASE - EXTERNAL DISTRIBUTION

John Wooley,
Deputy Associate Director, Health and Environmental Research
Office of Biological and Environmental Research
SC-70
Rm G-163
19901 Germantown Road
Germantown, Maryland 20871-1290

CDR Shaun Jones, M.D., USN
DARPA/DSO
3701 N. Fairfax Drive
Arlington, VA 22203-1714

Dr. Dennis M. Healy, Jr.
Program Manager,
Applied and Computational Mathematics Program,
Defense Sciences Office
DARPA/DSO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Dr. Stephen S. Morse,
Program Manager, Defense Sciences Office
DARPA/DSO
3701 North Fairfax Drive
Arlington, VA 22203-1714

Dr. Ellen Stechel
Ford Motor Company
MD 3028/SRL
PO Box 2053
Dearborn, MI 48121-2053

Dr. Niels Jensen
T-11, MS B262
Los Alamos National Laboratory,
Los Alamos, NM 87545

Professor Richard Cairncross
Department of Chemical Engineering
Drexel University
31st and Market Streets
Philadelphia, PA 19104

Professor Charles Zukoski
Dept. of Chemical Engineering
107 Roger Adams Lab
600 S. Mathews Ave,
Urbana, IL 61801

Professor Robert Eisenberg
Department Chair,
Dept. of Molecular Biophysics and Physiology
Rush Medical Center
1750 W. Harrison St.,
Chicago, IL 60612

Professor Shankar Subramaniam
4041 Beckman Institute
405 N. Mathews
Urbana, IL 61801

Professor Martha Mitchell
Department of Chemical Engineering
New Mexico State University
2040 Guthrie Pl
Las Cruces, NM 88001

Professor Paul van Tassel
Department of Chemical Engineering
Engineering Building
Wayne State University
5050 Anthony Wayne Drive
Detroit, MI 48202

Professor Andrew Hillier
Department of Chemical Engineering
Thornton Hall
University of Virginia
Charlottesville, VA 22903-2442

Professor David Ford
Department of Chemical Engineering
Texas A&M University,
College Station, TX 77843-3122

Marcus Martin
Department of Chemistry
University of Minnesota,
Minneapolis, MN 55455

UNLIMITED RELEASE - INITIAL DISTRIBUTION

#	MS	Name, org	#	MS	Name, org
1	0151	Gerry Yonas, 9000	1	0340	Jeff Braithwaite, 1832
1	0513	Bob Eagan, 1000	1	0340	Mike Dugger, 1832
1	0321	Bill Camp, 9200	1	1407	Mike Kent, 1832
1	1427	Tom Picraux, 1100	1	0340	Lysle Montes, 1832
1	9004	Miriam John, 8100	1	0340	Rob Sorensen, 1832
1	1111	Grant Heffelfinger, 9225	1	0958	John Emerson, 1472
1	1111	Sudip Dosanjh, 9221	1	1425	Tony Ricco, 1715
1	0340	Wendy Cieslak, 1800	1	0827	Channy Wong, 9114
1	0601	Jeff Nelson, 1113	1	0367	Terry Guilinger, 1811
1	1349	Alan Hurd, 1841	1	1405	B.G. Potter, 1812
1	1110	David Womble, 9222	1	1411	Celeste Drewien, 1834
1	1411	Julia Phillips, 1834	1	0710	Tina Nenoff, 1845
25	1111	Laura Frink, 9225	1	1111	Steve Plimpton, 9221
10	1111	Andrew Salinger, 9221	1	1111	John Shadid, 9221
1	1349	Frank van Swol, 1841	1	1111	Karen Devine, 9226
1	1349	Jeff Brinker, 1831	1	1110	Bill Hart, 9222
1	1349	John Shelnut, 1841	1	1110	Sorin Istrail, 9222
1	1349	John Curro, 1841	1	1111	Peter Schultz, 9225
1	1411	Gary Grest, 1834	1	1111	Mark Sears, 9225
1	9214	Carl Melius, 8130	1	1111	Richard Fye, 9225
1	9042	Robert Nilson, 8345	1	1111	Mark Stevens, 9225
1	9042	Stewart Griffiths, 8345	1	1111	Aidan Thompson, 9225
1	0826	Randy Schunk, 9111	1	1111	Jean-Loup Faloun, 9225
1	0826	Ken Chen, 9111	1	1110	Richard Lehoucq, 9222
1	9018	Central Technical Files, 8940-2	1	0834	Lisa Mondy, 9112
2	0899	Technical Library, 4916			
1	0619	Review & Approval Desk, 15102			
		For DOE/OSTI			
1	0188	LDRD Office, 4001			