Journal of Computational Physics **159,** 407–424 (2000)
doi:10.1006/jcph.2000.6454, available online at http://www.idealibrary.com on IDEAL®

Two- and Three-Dimensional Nonlocal Density Functional Theory for Inhomogeneous Fluids

I. Algorithms and Parallelization

Laura J. Douglas Frink*,[1] and Andrew G. Salinger†

*Computational Biology and Materials Technology Department, and †Parallel Computational Sciences Department, Sandia National Laboratories, Albuquerque, New Mexico 87185
E-mail: *ljfrink@sandia.gov, †agsalin@sandia.gov

Received August 23, 1999; revised January 19, 2000

Fluids adsorbed near surfaces, near macromolecules, and in porous materials are inhomogeneous, exhibiting spatially varying density distributions. This inhomogeneity in the fluid plays an important role in controlling a wide variety of complex physical phenomena including wetting, self-assembly, corrosion, and molecular recognition. One of the key methods for studying the properties of inhomogeneous fluids in simple geometries has been density functional theory (DFT). However, there has been a conspicuous lack of calculations in complex two- and three-dimensional geometries. The computational difficulty arises from the need to perform nested integrals that are due to nonlocal terms in the free energy functional. These integral equations are expensive both in evaluation time and in memory requirements; however, the expense can be mitigated by intelligent algorithms and the use of parallel computers. This paper details our efforts to develop efficient numerical algorithms so that nolocal DFT calculations in complex geometries that require two or three dimensions can be performed. The success of this implementation will enable the study of solvation effects at heterogeneous surfaces, in zeolites, in solvated (bio)polymers, and in colloidal suspensions.

Key Words: inhomogeneous fluids; solvation; density functional theory; molecular theory; parallelization; finite element.

1. INTRODUCTION

Fluids near surfaces or macromolecules have properties (e.g., viscosity and density) that are markedly different from the bulk properties of these fluids. Predicting the structure

[1] To whom correspondence should be addressed.

0021-9991/00 $35.00

of fluids in confined spaces is ultimately critical for calculating adsorption in, solvation forces on, and wetting of complex surfaces, macromolecules, and porous materials. The structure of these fluids may be found either with grand canonical Monte Carlo (GCMC) simulations [1] or molecular dynamics simulations or with molecular theories such as the density functional theory (DFT) discussed in this paper [2].

In DFT for inhomogeneous fluids, the surface (or macromolecule) generates an external field in which the fluid molecules equilibrate. Many formulations for DFTs have been explored in the past two decades [2]. The simplest of these are local DFTs. In these cases, the free energy density is assumed to depend only on the density at one point in the fluid. Local DFTs overestimate the energy penalties associated with rapidly varying density profiles and as a result are inadequate for describing density distributions in fluids near solid interfaces. The alternative is a nonlocal approach that defines the free energy density to be a function of a weighted average of all the densities in a nearby region of the fluid.

Unlike DFT for electronic structure calculations, there is no exact Hamiltonian to describe inhomogeneous fluids. Therefore nonlocal DFTs were initially developed on a somewhat ad hoc basis with the goal of reproducing GCMC simulations. The most widely applied of these ad hoc approaches was the Tarazona functional [3]. While the Tarazona approach is successful in treating both hard sphere and Lennard–Jones fluids near surfaces, it is difficult to extend to multicomponent systems with particles of unlike size. More recently, the development of a fundamental measures DFT by Rosenfeld [4] and its application to inhomogeneous fluids by Kierlik and Rosinberg [5] have allowed for straightforward extension of DFT approaches to multicomponent systems.

Applications of the Tarazona and, to a lesser extent, the Rosenfeld functionals have until recently [6, 7] been limited to problems where the density profiles are uniform in two dimensions (2D). The result is a 1D numerical problem that must be solved. Examples include adsorption [8] and capillary condensation [9] in slit-like pores, cylindrical pores [10], and spherical cavities [11]; wetting at homogeneous planar interfaces [12]; and nucleation of droplets [13]. Estimation of solvation forces between rough and curved interfaces, as well as adsorption in pore networks, has been obtained with superposition of 1D solutions [14–16].

While 1D calculations have provided a great deal of insight into the underlying physics controlling inhomogeneous fluids at homogeneous interfaces, these 1D calculations are ultimately limited in probing the behavior of inhomogeneous fluids at heterogeneous (either in geometry or chemistry) interfaces. A great deal of interfacial physics and chemistry from self-assembly to corrosion to molecular recognition depends on surface heterogeneities. Therefore, extending DFT capabilities to 2D and 3D is a needed development.

To further motivate the discussion, an example of a density profile (calculated using the algorithms described in this paper) in a fluid near a chemically heterogeneous surface is shown in Fig. 1. In this case, the surface was composed of alternating hydrophilic and hydrophobic stripes. The hydrophilic portion of the surface is located at $x/\sigma = 0$ and $0 \leq y/\sigma < 5$, and the hydrophobic portion of the surface is found at $x/\sigma = 0$ and $5 \leq y/\sigma \leq 10$, where σ is the diameter of a fluid particle. The solution domain has periodic boundaries in y and a bulk boundary at $x/\sigma = 10$. Figure 1 demonstrates the complexity in the fluid structure near this interface with rapid variations in density both parallel and perpendicular to the surface. A full analysis of wetting and phase transitions in fluids near this type of chemically heterogeneous surface may be found elsewhere [17–20].

The remainder of this paper is focused on presenting our numerical implementation of DFT for 2D and 3D applications. The details of our algorithms and numerical methods are

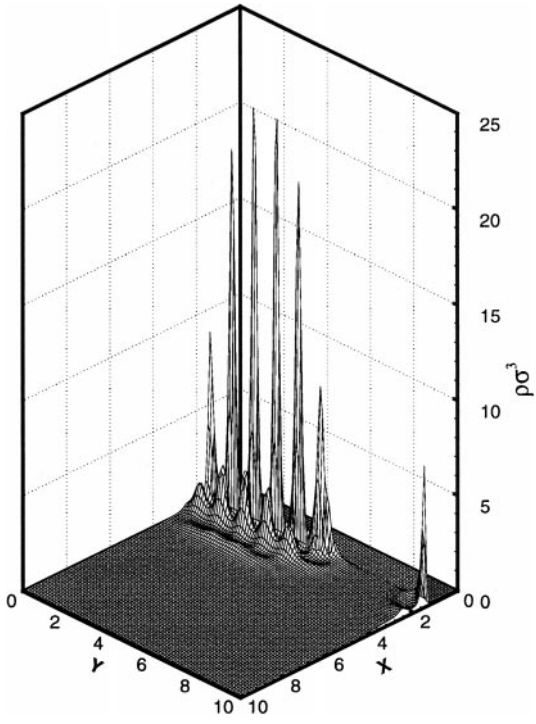

FIG. 1. The reduced number density, $\rho\sigma^3$, as a function of position (x and y in units of σ) near a surface that is chemically heterogeneous.

presented and applied to the simplest fluid model, a single-component hard sphere fluid. We defer the discussion of multicomponent systems, Lennard–Jones fluids, and charged systems to future papers. The outline of this paper is as follows. DFT is briefly reviewed in Section 2, our basic numerical approach is outlined in Section 3, a variety of algorithms for improved performance are presented in Section 4, and parallelization is discussed in Section 5. A detailed discussion of precision and solvated polymers may be found in a companion paper (Part II) [21].

2. THEORY

Density functional theory is a thermodynamic theory in which the free energy of the system is minimized with respect to the fluid density distribution, $\rho(\mathbf{r})$. The density distribution is inhomogeneous due to the surfaces or macromolecules in the system which exert an external field, V^{ext}, on the fluid molecules. The ensemble in which DFT calculations are most often performed is the grand canonical ensemble (constant chemical potential, μ; volume, V; and temperature, T). The grand canonical ensemble is open with respect to material exchange and so is suitable for any confined fluid that interacts with a bulk fluid.

The grand free energy, Ω is $\Omega = F - G$, where F is the Helmholtz free energy, $G = \mu N$ is the Gibbs free energy, and $N = \int dr\rho(r)$ is the number of fluid particles in the volume of interest. The equilibrium density distribution is the one that minimizes the free energy via

$$\left(\frac{\delta\Omega}{\delta\rho(\mathbf{r})}\right)_{T,\mu} = 0. \tag{1}$$

The specific definition of Ω depends on the fluid model of interest. We restrict our discussion to hard sphere fluids.

Hard sphere fluids are defined by the pair interaction potential

$$u(r_{12}) = \begin{cases} \infty & \text{if } r_{12} = |\mathbf{r}_2 - \mathbf{r}_1| < \sigma, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where σ is the diameter of a fluid particle.

Separating the Helmholtz free energy into ideal, F_{id}, and excess hard sphere, F_{hs}, contributions, the grand potential is

$$\Omega = F_{\text{id}} + F_{\text{hs}} - \int d\mathbf{r}\, \rho(\mathbf{r})[V^{\text{ext}}(\mathbf{r}) - \mu], \tag{3}$$

where

$$F_{\text{id}} = kT \int d\mathbf{r}\, \rho(\mathbf{r})\{\ln(\Lambda^3 \rho(\mathbf{r})) - 1\}, \tag{4}$$

and

$$F_{\text{hs}} = \int d\mathbf{r}\, \Phi\{\bar{\rho}_\gamma(\mathbf{r})\}. \tag{5}$$

The parameters in Eq. (4) are the deBroglie wavelength (Λ) and the Boltzmann constant (k). The hard sphere contribution is written in terms of a free energy density, Φ, that depends on the *nonlocal* densities, $\bar{\rho}$. These nonlocal densities are defined by the weight functions, $w^{(\gamma)}$, as

$$\bar{\rho}_\gamma(\mathbf{r}) = \int d\mathbf{r}' \rho(\mathbf{r}') w^{(\gamma)}(\mathbf{r} - \mathbf{r}'). \tag{6}$$

The six weight functions in Rosenfeld's theory [4] are

$$\begin{aligned} w^{(3)}(\mathbf{r}) &= \theta(r - R) \\ w^{(2)}(\mathbf{r}) &= 4\pi R w^{(1)}(\mathbf{r}) = 4\pi R^2 w^{(0)}(\mathbf{r}) = \delta(r - R) \\ w^{(V2)}(\mathbf{r}) &= 4\pi R w^{(V1)}(\mathbf{r}) = (\mathbf{r}/r)\delta(r - R), \end{aligned} \tag{7}$$

where $\mathbf{r}$ indicates a vector and $r = |\mathbf{r}|$. These weight functions are based on the geometry of the fluid particles as θ is the step function, δ is the Dirac delta function, and R is the radius of a particle. Thus the integrals over weight functions are related to the volume, surface area, and radius of the particle. The hard sphere free energy density is a sum of scalar and vector contributions, $\Phi = \Phi_s + \Phi_v$, with

$$\begin{aligned} \Phi_s &= -\bar{\rho}_0 \ln(1 - \bar{\rho}_3) + \frac{\bar{\rho}_1 \bar{\rho}_2}{1 - \bar{\rho}_3} + \frac{1}{24\pi} \frac{\bar{\rho}_2^3}{(1 - \bar{\rho}_3)^2} \qquad \text{and} \\ \Phi_v &= -\frac{\bar{\rho}_{V1} \cdot \bar{\rho}_{V2}}{1 - \bar{\rho}_3} - \frac{1}{8\pi} \frac{\bar{\rho}_2 (\bar{\rho}_{V2} \cdot \bar{\rho}_{V2})}{(1 - \bar{\rho}_3)^2}. \end{aligned} \tag{8}$$

Performing the functional minimization in Eq. (1) on Eq. (3) yields the Euler–Lagrange (EL) equation

$$\mu = kT\,\ln(\rho(\mathbf{r})) + V^{\text{ext}}(\mathbf{r}) + \int d\mathbf{r}' \left(\sum_{\gamma} \frac{\partial \Phi}{\partial \bar{\rho}_\gamma} w^{(\gamma)}(\mathbf{r} - \mathbf{r}') \right), \tag{9}$$

which must be solved at every point in the mesh.

Note that Eq. (9) includes a volumetric integral, whose integrand is a nonlinear function of the $\bar{\rho}$ functions, which are themselves integrals over the unknown density distribution. The resulting nested integrals present the main computational difficulty in solving DFT.

The computational domain in any calculation is rectangular with edges of length L_x, L_y, and L_z. There are four types of boundary conditions that may be applied at the edges of the computational domain. In the case of bulk boundaries, the fluid is assumed to be uniform with $\rho = \rho_b$ beyond the computational domain. For wall boundaries, the fluid densities are $\rho = 0$ beyond the computational domain; for periodic boundaries, the fluid densities beyond the edge of the domain (assuming the periodic boundary is applied in the x direction) are

$$\rho(x, y, z) = \begin{cases} \rho(x + L_x, y, z) & \text{if } x < 0 \\ \rho(x - L_x, y, z) & x > L_x \end{cases}. \tag{10}$$

Finally, for reflective boundaries in the x direction, the boundary conditions are

$$\rho(x, y, z) = \begin{cases} \rho(|x|, y, z) & \text{if } x < 0 \\ \rho(2L_x - x, y, z) & x > L_x \end{cases}. \tag{11}$$

3. NUMERICAL METHODS

3.1. *Mesh and Quadrature*

A collocation approach is used to determine the density distribution that satisfies the EL equation in the neighborhood of surfaces. The problem geometry is represented by a mesh. The densities at the nodes of the mesh are unknowns, and the density distribution is assumed to vary linearly between the nodes. The EL equations are required to be satisfied at the nodes.

The main computational complexity is computing the integrals of a function f over the weight functions in Eq. (7), whether it is to evaluate the $\bar{\rho}$ functions ($f = \rho$) or the integral term in the EL equation ($f = \partial\Phi/\partial\bar{\rho}$). The integrals are computed numerically, using a precalculated numerical integration stencil,

$$\int f(\mathbf{r}') w^{(\gamma)}(\mathbf{r} - \mathbf{r}') d\mathbf{r}' \approx \sum_{i=1}^{N_{\text{sten}}^{(\gamma)}} C_i^{(\gamma)} \omega_i^{(\gamma)} f_i, \tag{12}$$

where $N_{\text{sten}}^{(\gamma)}$ is the total number of points in the γth weight function stencil. The weight functions are split into their prefactors (e.g., $C_i^{(\gamma)} = \mathbf{r}/(4\pi Rr)$ for $w^{(V1)}$) and the fundamental θ and δ weight functions, ω. For 1D and 2D problems, the 3D θ and δ functions are collapsed analytically into integrals along a line or on a disk, respectively, before the numerical integration is performed. For the 2D stencils, we integrate Eq. (12) over one

dimension (z') analytically to get

$$2R \iint C^{(\gamma)} f(x', y') \sqrt{1 - x'^2 - y'^2}\, \theta(R - \sqrt{x'^2 + y'^2})\, dx' dy' \tag{13}$$

for integrations over the 3D θ function stencils, and

$$2 \iint C^{(\gamma)} \frac{f(x', y')}{\sqrt{1 - x'^2 - y'^2}}\, \theta(R - \sqrt{x'^2 + y'^2})\, dx' dy' \tag{14}$$

for integrations over the 3D δ function stencils. In 1D, Eq. (12) is integrated over two dimensions (z', y') to obtain

$$\pi R^2 \int C^{(\gamma)} f(x')(1 - x'^2)\, \theta(R - |x'|)\, dx' \tag{15}$$

for integrations over the 3D θ function stencils, and

$$2\pi R \int C^{(\gamma)} f(x')\, \theta(R - |x'|)\, dx' \tag{16}$$

for integrations over the 3D δ function stencils. It was found that very accurate stencils are needed to obtain accurate density profiles with the rapid variations as shown in Fig. 1. So, all integration stencils, $w_i^{(\gamma)}$, are calculated numerically by finding the contribution to each node from each element. In elements that fall entirely within the sphere (3D), disk (2D), or endpoints (1D), the integrand is smooth, and a simple Gauss quadrature is used. In elements that straddle the sphere (disk, endpoints), the integrand is discontinuous, so a simple midpoint rule is used with numerous equally spaced (and equally weighted) quadrature points.

For an arbitrary nonuniform mesh, the calculation and storage of these stencils would be prohibitively expensive. Instead, we restrict our mesh to be a rectangular, Cartesian mesh with constant node spacings, Δx in each direction. The rectangular mesh allows each node to be identified with an (i, j, k) integer location. With the uniform mesh, the stencils can be calculated once and stored in a list that contains the offsets in (i, j, k) space and the weights $\omega_i^{(\gamma)}$. The stencil calculation is thereby reduced to a quick preprocessing step. The main drawback of requiring a rectangular mesh is that surfaces with boundaries that do not align with the Cartesian axes must be represented by staircased boundaries. The consequences of surface staircasing are considered in Part II.

The numbers of quadrature points in the δ and θ function stencils for 1D, 2D, and 3D calculations on meshes with $\Delta x = 0.1\sigma$ and $\Delta x = 0.05\sigma$ are shown in Table I. The number of stencil points in the δ and θ function are identical in 1D and 2D due to analytical integration over dimensions with uniform densities.

For mesh nodes that fall inside any of the surfaces in the system, we solve the trivial equation $\rho = 0$ instead of the EL equation. At each boundary of the computational domain, one of the four possible types of boundary conditions must be chosen to describe the fluid outside the domain. Note that the constant density conditions ($\rho = \rho_b$ or $\rho = 0$) are *not* strongly enforced on the boundaries. Rather, these known densities are used to compute contributions to integrals that extend past the edge of the domain.

TABLE I

Number of Stencil Points, N_{sten}, for the δ and θ Function Quadrature Stencils in 1D, 2D, and 3D When Grid Spacing Is either 0.1σ or 0.05σ

	δ ($\Delta x = 0.1\sigma$)	θ ($\Delta x = 0.1\sigma$)	δ ($\Delta x = 0.05\sigma$)	θ ($\Delta x = 0.05\sigma$)
1D	11	11	21	21
2D	109	109	385	385
3D	844	1015	3676	6205

3.2. *Solution Method*

The solution of the EL equations (Eq. (9)) is straightforward if the fluid density varies only in one dimension [8, 10, 13]. In these cases the DFT may be solved with either successive substitution (Picard iterations) or Newton's method on a desktop workstation.

The Picard and Newton approaches both have their advantages and disadvantages. Newton's method requires the storage of a large Jacobian matrix, but it is very stable. Solutions can often be found in $O(10)$ Newton iterations [22]. Picard iterations are more straightforward to implement and require a great deal less memory as no Jacobian is stored; however, this approach is less stable, requiring a careful mixing of old and new solutions, and $O(1000)$ iterations to convergence. We have implemented Newton's method to take advantage of its superior stability and convergence properties.

The solution to the resulting system of equations is found iteratively with Newton's method. This requires solving the matrix problem $J_{ij}\Delta_j = -R_i$, where J is the Jacobian matrix, $\Delta_j = \rho_j^{(k+1)} - \rho_j^{(k)}$ is the difference between the solution vector at the $(k+1)$st Newton iteration and the kth iteration, and R is the vector of residuals (from the EL equation). The Jacobian matrix, $J_{ij} = \delta R_i/\delta\rho_j$, is

$$J_{ij} = \frac{\delta^2\Omega}{\delta\rho_i(\mathbf{r})\delta\rho_j(\mathbf{r}')} = \frac{\delta_{i,j}(\mathbf{r},\mathbf{r}')}{\rho_i(\mathbf{r})} + \frac{1}{kT}\int d\mathbf{r}'' \sum_\gamma \sum_\xi \frac{\partial^2\Phi}{\partial\bar{\rho}_\gamma \partial\bar{\rho}_\xi}(\mathbf{r}'')w^{(\gamma)}(\mathbf{r}_i - \mathbf{r}'')w^{(\xi)}(\mathbf{r}'' - \mathbf{r}'_j). \quad (17)$$

The resulting matrix problem is solved using the Aztec [23] linear solver library. A GMRES solver with no preconditioning and Jacobi scaling usually works well and is the basis for all the results presented here.

The two primary challenges to overcome for nonlocal 2D and 3D DFT calculations are the complexity involved in filling the Jacobian and the memory required to store it. The quadrature stencils of Table I will result in a nonzero Jacobian entry for every position that is within $2R$ of the node of origin. More specifically, for the $\Delta x = 0.1\sigma$ mesh there are 21 (1D), 401 (2D), or 7221 (3D) nonzeros per row. Clearly, the EL equations result in a far denser Jacobian matrix than those coming from most discretizations of partial differential equations.

To demonstrate the complexity involved in filling the Jacobian, consider Eq. (17). The hard sphere term for J_{ij} involves integrating over the region of overlap, $w^{(\gamma)}(\mathbf{r} - \mathbf{r}'')w^{(\xi)}(\mathbf{r}'' - \mathbf{r}')$, of two quadrature stencil functions that start from the ith $(\mathbf{r})$ and the jth $(\mathbf{r}')$ unknown, respectively (see the 2D schematic in Fig. 2).

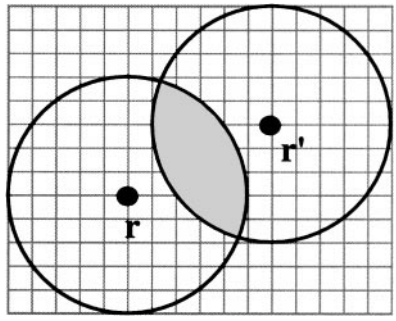

FIG. 2. A schematic of the Jacobian calculation. Each i, j (or $\mathbf{r}$, $\mathbf{r}'$) entry in the Jacobian is found by integrating the overlap regions of the weight functions that are centered at $\mathbf{r}$ and $\mathbf{r}'$ respectively. The overlap region is the central shaded region and contains all the $\mathbf{r}''$ with nonzero contributions to J_{ij} in Eq. (17).

A very inefficient way to calculate the integrals of overlapping weight functions would be to calculate one Jacobian entry at a time. With this approach one would loop through each combination of two stencils using the mesh points corresponding to the ith and jth unknowns as the origin and search for overlaps. With this approach there would be much wasted effort identifying nonoverlapping regions of the two stencils, and the scaling to fill one row of the Jacobian would go like N_{sten}^3, where $N_{\text{sten}} = \max(N_{\text{sten}}^{(\gamma)}) = N_{\text{sten}}^{(\theta)}$.

An alternative approach that we have implemented is to fill the Jacobian by rows. The procedure begins with a loop over one of the stencils with the origin at the ith unknown (ith row of the Jacobian). Each of the i_{sten} nodes reached by the stencil is necessarily in the overlap region of the ith unknown and a jth unknown that is hit by a second quadrature stencil that has the i_{sten}th unknown as its origin. Of course, each jth unknown hit from this second stencil corresponds to a different column in the Jacobian. As a result, a given ijth Jacobian entry is only completely filled when all contributions of all possible combinations of weight functions have been calculated. For this algorithm, the scaling to fill one row of the Jacobian goes like N_{sten}^2.

The scaling for the entire Jacobian fill will go like $N_{\text{nodes}} N_{\text{sten}}^2$, where N_{nodes} is the number of mesh points in the domain. With respect to the number of mesh points in the domain, the scaling is linear, but with respect to mesh spacing, Δx, the scaling is potentially much worse because both N_{sten} and $N_{\text{nodes}} \propto \Delta x^{-D}$ for large enough N_{sten}. Thus in the worst case, the Jacobian fill scaling will go like Δx^{-3D} (where D is the number of spatial dimensions in the problem) assuming that the grid is refined in all dimensions simultaneously. However, when N_{sten} is small (for a coarse grid), the surface contributions, as represented by $N_{\text{sten}}^{(\delta)}$, can be dominant. Thus, the lower bound on the scaling will be $\Delta x^{-(3D-2)}$.

Figure 3 shows the observed scaling of the Jacobian and Residual fills in 1D, 2D, and 3D with the number of nodes at fixed Δx, and with varying Δx at fixed domain size. The physical system used for these timings was a uniform fluid with bulk boundary conditions on all edges of $\rho_b \sigma^3 = 0.6$. The timings were done on a 433-MHz DEC Alpha workstation.

4. ALGORITHMS FOR IMPROVED PERFORMANCE

Solving the DFT using the Jacobian fill algorithm described in the previous section is expensive for 2D and particularly 3D problems. Therefore, several strategies aimed at mitigating the expense of the fill have been implemented, and are outlined here.

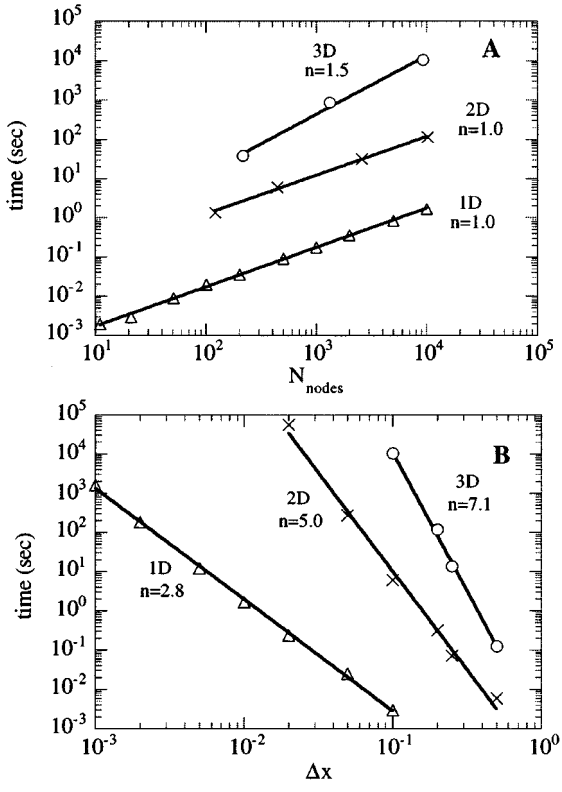

FIG. 3. Jacobian fill time as a function of the size of the mesh, $L = L_x = L_y = L_z$, in units of σ (A), and as a function of the grid spacing $\Delta x = \Delta y = \Delta z$ (B) in units of σ. In (A), the grid spacing is $\Delta x = \Delta y = \Delta z = 0.1\sigma$. In (B) the domain size is $L_x = L_y = L_z = 2\sigma$. The scaling coefficients, n (time $\propto N_{\text{nodes}}^n$ or time $\propto \Delta x^{-n}$), are indicated.

4.1. *Jacobian Coarsening*

One strategy for reducing the cost of DFT calculations is to recognize that the role of the Jacobian is to efficiently point the vector of unknowns toward the direction of the equilibrium solution. Thus, a Jacobian matrix that does not have the same degree of accuracy as the residual equations can still enable convergence of the Newton's method, without sacrificing any accuracy in the solution. One option is to coarsen the quadrature stencils for the Jacobian.

In Fig. 4, we present scalings for two types of Jacobian coarsening. In the first case, the Jacobian integrals are coarsened by a factor of two as compared with the residuals. In the second case, the Jacobian integrals are all based on mesh densities of $\Delta x = 0.2\sigma$. For comparison, the cases where there is no Jacobian coarsening (Fig. 3) are also included. In the first case, the scaling of the code is unchanged although the performance is improved by about a factor of 3. In the second case, the scaling of the code as well as the performance is significantly improved.

4.2. *Minimal Set Jacobian*

Another approach to improving the performance of the Jacobian fill relies on the similarity of the integrand in the $\bar{\rho}$ calculation (Eq. (6)) and the integrand for the hard sphere term in the Jacobian (Eq. (17)).

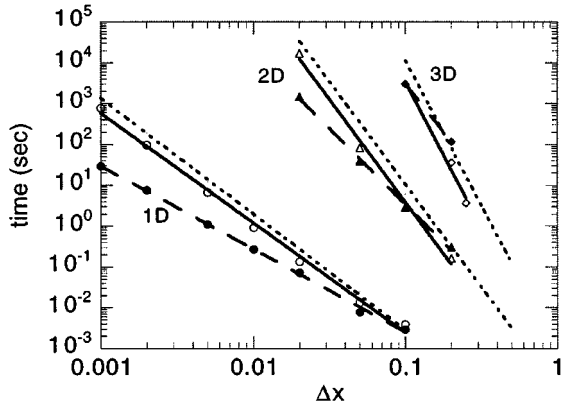

FIG. 4. The Jacobian fill time as a function of the mesh density, $\Delta x/\sigma$, used for the *residual* calculations. The various curves have Jacobian stencils coarsened by a factor of two (solid lines), Jacobian stencils based on $\Delta x = 0.2\sigma$ mesh always (dashed lines), and no Jacobian coarsening (dotted lines). The scaling exponents associated with solid lines are $n = 2.7$ (1D), $n = 5.0$ (2D), and $n = 7.0$ (3D). The scaling exponents associated with dashed lines are $n = 2.0$ (1D), $n = 3.7$ (2D), and $n = 4.7$ (3D).

For all of the results presented in the previous sections, the $\bar{\rho}_\gamma$ were calculated on the entire mesh prior to loading the Jacobian. However, the loops for the $\bar{\rho}$ calculation (over $N_{\text{sten}}^{(\delta)}$ and $N_{\text{sten}}^{(\theta)}$) along with the operations performed (locating stencil points and applying boundary conditions) are identical to the loops and operations performed in the Jacobian fill for the last term in Eq. (17). Thus the weight functions, $w^{(\gamma)}$, may be enumerated and stored as a function of mesh point as well as the quadrature point when the $\bar{\rho}_\gamma$'s are calculated.

While this method has potential both for reducing the number of operations in the fill (by eliminating boundary checking) and for further generalization to a nonuniform mesh (because the $w^{(\gamma)}$ functions are stored for every point in the mesh), this approach requires considerable memory. The memory requirement is minimized by assuming that the $w^{(2)}$ and $w^{(3)}$ are dominant in determining the solution. So, only this *minimal set* of weight functions is stored, and the Jacobian is based only on these two (of four possible) scalar weight-function contributions. Neglecting the vector contributions and the $w^{(1)}$ and $w^{(0)}$ terms leads to a Jacobian that is not exact.

The performance gain from this approach comes from minimizing the number of operations performed in the innermost loop of the Jacobian. Specifically, the stencil offset and boundary checking procedures are replaced by a multiplication of $C^{(\gamma)} \times w_i^{(\gamma)}$ and a copy into memory. The specific number of operations saved depends on the type and proximity of domain boundaries as well as the dimensionality of the problem.

While considerable speedup is obtained with these minimal set Jacobians (see Fig. 5), the price to be paid can be decreased robustness of the solution method.

4.3. *Mesh Coarsening*

While the uniform fluid solution considered above serves to demonstrate the scaling behavior of various algorithms, it is an uninteresting case from a physical point of view. The forte of the DFT approach is to calculate density distributions near surfaces. It turns out that the presence of surfaces also provides an opportunity for further improvements in code performance.

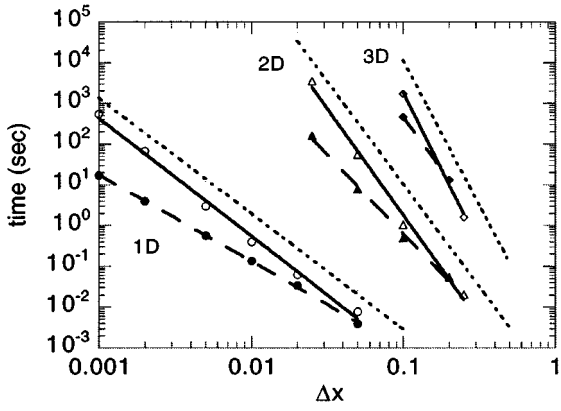

FIG. 5. The scaling of the Jacobian fill as a function of the mesh density, $\Delta x = \Delta y = \Delta z$, used for the *residual* calculations in units of σ. The dotted lines are the exact Jacobian results (see Fig. 3), the solid lines are the minimal set Jacobian without Jacobian coarsening, and the dashed lines have both the minimal set Jacobian and Jacobian coarsening where the Jacobian integrals are based on $\Delta x = 0.2\sigma$. The scaling exponents associated with solid lines are $n = 2.9$ (1D), $n = 5.2$ (2D), and $n = 7.5$ (3D). The scaling exponents associated with dashed lines are $n = 2.1$ (1D), $n = 3.9$ (2D), and $n = 5.1$ (3D).

In all the cases described above, the meshes were uniform. However, when there are surfaces present, the solutions can be expected to be most rapidly varying near the surfaces. Thus a mesh that is denser near the surfaces would be appropriate. Unfortunately an unstructured mesh would require storage of quadrature stencils for every node and thus would be unfeasible.

One alternative we have implemented is to apply a nonuniform mesh that requires only a limited number of additional quadrature stencils. Such a mesh is shown in Fig. 6, where a mesh becomes coarser in steps away from a flat planar surface on the left side of a 2D domain. For this type of nonuniform mesh, a complete set of quadrature stencils is needed only for each region (or zone) of constant mesh density.

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. In each successive zone away from the surface, the mesh is coarsened by a factor of two.

Quadrature stencils need not be stored for each point if the dropped nodes (intersection of light lines in Fig. 6) are retained in some form in the calculation. Then when starting from

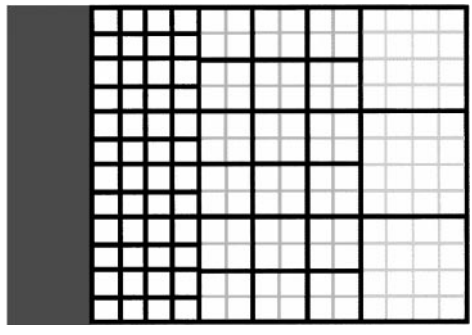

FIG. 6. A schematic of a coarsened mesh. The light lines show the underlying fine mesh, and the dark solid lines show a mesh that is coarsened by factors of two in steps away from a surface.

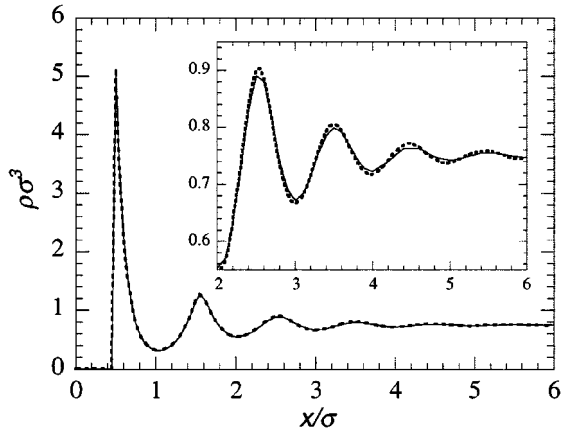

FIG. 7. The number density, $\rho\sigma^3$ as a function of the distance away from a hard planar surface. The dotted line show the result for a uniform mesh with $\Delta x = 0.05\sigma$. The solid line shows the result for a case with mesh coarsening and three zones. The first zone has $\Delta x = 0.05\sigma$ in the range $0 < x/\sigma < 2$, the second zone has $\Delta x = 0.1\sigma$ in the range $2 < x/\sigma < 4$, and the third has $\Delta x = 0.2\sigma$ in the range $4 < x/\sigma < 6$. The inset shows an expanded view of the region $2\sigma < x < 6\sigma$.

a region of dense quadratures, the dense stencil may be applied without difficulty. In our implementation, the fluid density at a dropped node is taken to be the mean of the surrounding nodes. The corresponding residual and Jacobian equations are orders of magnitude easier to fill than the EL equations. Thus there is negligible expense with retaining these coarsened nodes in the solution vector. This approach results in a much sparser matrix and a faster fill because many of the integrals are calculated with far fewer quadrature points.

As an example of the benefits of mesh coarsening, consider the 1D density profiles perpendicular to a smooth infinite planar surface immersed in a liquid-like fluid with bulk density $\rho\sigma^3 = 0.75$ shown in Fig. 7. This figure compares results from a uniform mesh ($\Delta x = 0.05\sigma$) with those of a coarsened mesh with three zones. The total solve time for the coarsened mesh (0.255 s) was 2.2 times faster than that for the uniform mesh (0.572 s). The total number of Newton iterations required in both cases was six, and the errors in critical output parameters were all less than 1%, as is detailed in Table II.

The total savings due to mesh coarsening is dependent on the fraction of the fluid nodes that are near the surfaces as well as the mesh density of the finest grid. Table III shows timings and required number of Newton iterations for four different 1D, 2D, and 3D cases using

TABLE II

Comparison of Critical Output Parameters: Contact Value of Density, ρ_w; Surface Free Energy, Ω^s; and Excess Adsorption, Γ^{ex}, for Uniform and Mesh Coarsened Calculations Shown in Fig. 6

Parameter	Uniform	Coarsened	Error (%)
$\rho_w\sigma^3$	5.1224	5.1202	0.04
$\Omega^s\sigma^2/kT$	1.16291	1.16277	0.01
$\Gamma^{ex}\sigma^2$	−0.1159	−0.1163	0.34

Note. The surface free energy is reduced by the Boltzmann constant, k, and the temperature, T.

TABLE III
Comparison of Various Algorithms for Problems of Different Dimensionality, D, and Mesh Spacing, $\Delta x/\sigma$

D(Δx)	Basic[a]	Coarse[b,c] mesh	Coarse Jac[d] mnimal Jac	All[e]	Speedup
			Jacobian Fill		
1D(0.05)	0.051	0.020	0.019	0.0077	7
2D(0.1)	8.86	2.97	0.575	0.215	41
2D(0.05)	397.9	114.1	8.56	2.46	162
3D(0.1)	5589	1798	167.8	52.1	107
			Iterations		
1D(0.05)	6	6	11	10	
2D(0.1)	6	6	12	11	
2D(0.05)	6	6	13	12	
3D(0.1)	6	6	13	12	
			Total Time		
1D(0.05)	0.578	0.257	0.657	0.276	2
2D(0.1)	64.8	22.2	14.4	5.5	12
2D(0.05)	2777.1	798.2	225.8	64.0	43
3D(0.1)	38112	12121	2907	866	44

Note. The comparison is based on Jacobian fill time (top), Newton iterations (middle), and total solve time (bottom). All timings are given in seconds.

[a] Results for the basic algorithm (Section 2).

[b] Results for the mesh coarsening algorithm (Section 4.3).

[c] When $\Delta x = 0.05$ see Fig. 7 caption for mesh coarsening details. When $\Delta x = 0.1$ we used a two-zone mesh with $\Delta x = 0.1\sigma$ for $0 < x/\sigma < 2$ and $\Delta x = 0.2\sigma$ for $2 < x/\sigma < 6$.

[d] Results for combined coarsened Jacobian (Section 4.1) and minimal Jacobian (Section 4.2) algorithms.

[e] Results for combined mesh coarsening, Jacobian coarsening, and minimal Jacobian algorithms.

[f] Maximum speedup achieved with algorithms from Section 4.1–4.3.

several combinations of the algorithms presented in this section. The maximum speedups over the basic algorithm are detailed in the final column, with the total time to solution seeing a factor of order 40 speedup for the largest two problems.

4.4. *Enumerated Nonlocal Densities*

One final option for improving code performance at the cost of increased memory is to explicitly include residual equations for the nonlocal densities (denoted R_γ). In this case, the system of equations to be solved includes both the EL equation (see Eq. (9)) and $4 + 2D$ nonlocal density equations (see Eqs. (6) and (7)). In this case, the Jacobian entries corresponding to EL equations are

$$J_{ij} = \frac{\delta R_i(\mathbf{r})}{\delta \rho_j(\mathbf{r}')} = \frac{\delta_{ij}(\mathbf{r}, \mathbf{r}')}{\rho_i(\mathbf{r})} \tag{18}$$

and

$$J_{ij} = \frac{\delta R_i(\mathbf{r})}{\delta \bar{\rho}_{\gamma,j}(\mathbf{r}')} = \sum_{\epsilon} \frac{\partial^2 \Phi}{\partial \bar{\rho}_\epsilon \partial \bar{\rho}_\gamma}(\mathbf{r}'_j) w^{(\gamma)}(\mathbf{r}_i - \mathbf{r}'_j) \tag{19}$$

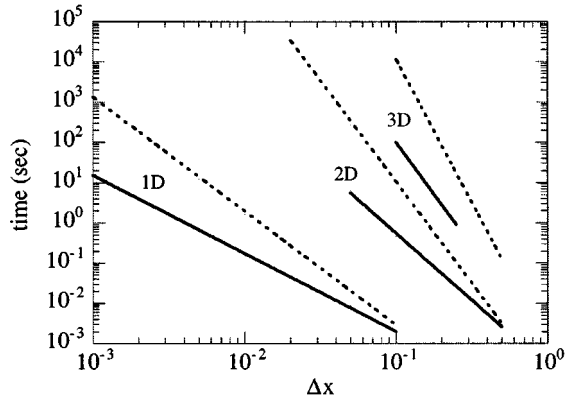

FIG. 8. The matrix fill time as a function of mesh spacing for the enumerated nonlocal density algorithm (solid lines) and the basic algorithm of Fig. 3 (dotted lines). The slopes of the solid lines are 1.9 (1D), 3.3 (2D), and 5.1 (3D).

while the Jacobian entries corresponding to the nonlocal density equations have the form

$$J_{ij} = \frac{\delta R_{\gamma,i}(\mathbf{r})}{\delta \bar{\rho}_{\gamma,j}(\mathbf{r}')} = -\delta_{ij}(\mathbf{r}, \mathbf{r}') \tag{20}$$

and

$$J_{ij} = \frac{\delta R_{\gamma,i}(\mathbf{r})}{\delta \rho_j(\mathbf{r}')} = w^{(\gamma)}(\mathbf{r}_i - \mathbf{r}'_j). \tag{21}$$

This Jacobian has far less complexity than the Jacobian of Eq. (13). There are no integrals, and so no calculation of overlapping weight functions is required. We have implemented this approach in conjunction with the mesh coarsening described earlier. We have also implemented a minimal set Jacobian that includes only the scalar nonlocal density equations. The scaling of these algorithms with mesh spacing is shown in Fig. 8. The scaling of the algorithms is much improved over that shown in Fig. 3. The total solve times using this approach for the cases outlined in Table III are 0.44, 3.4, 44, and 194 s for the 1D/0.05, 2D/0.1, 2D/0.05, and 3D/0.1 cases, respectively. Clearly this approach is particularly powerful for performing 3D calculations. However, one disadvantage is that Jacobian coarsening (which can provide a large speedup for the implicit nonlocal density algorithms) often fails due to the explicit inclusion of the integrals over the rapidly varying $\rho(r)$. As a result the optimal algorithm remains problem dependent.

5. PARALLELIZATION

All the timings in the previous sections were performed on a DEC Alpha workstation. They demonstrate that on this type of platform, many 2D problems can be performed, yet only small 3D problems are possible. In order to consider larger systems, we have developed a massively parallel implementation of the DFT code.

When implementing the DFT solve on massively parallel, distributed memory computers, the strategy is to split up the global domain so that each processor loads the residual and Jacobian entries for the rows of the unknowns within a unique local domain. Because the Jacobian and residual calculations require information outside the local domain, three coordinate systems are ultimately required.

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, but are ordered to minimize communication costs. 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. Finally, in parallel, an extended local coordinate system (ELCS) is also needed. This coordinate system contains all the local nodes on a given processor plus a larger rectangular cage that contains all the nodes needed for calculating the Jacobian entries of the local nodes. The ELCS is set up as a rectangular cage in order to simplify traversing the mesh.

At global domain boundaries, the ELCS is adjusted depending on the boundary condition. If the domain boundary condition is a bulk fluid, 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 ELCS 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 computational domain. 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 done all at once at the end of each Newton iteration.

Many issues regarding the parallelization of the code were facilitated by the Aztec parallel, iterative linear solver package [23]. In addition to efficiently solving the distributed matrix problem at every iteration of Newton's method, Aztec performed the key preprocessing step of identifying the ghost unknowns (those unknowns not owned by the current processor but needed to calculate the residual equations) and setting up the communications for sharing the residual and unknown vectors among processors.

Load balancing is one final challenge for solving the DFT on a distributed memory parallel computer. We have applied a weighted recursive spectral bisection method to determine which nodes on the mesh end up on which processors. 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 inside any surface, and for which the equation $\rho = 0$ is solved, are given weights near zero. Nodes that are being treated with a residual coarsening method also have weights near zero. Otherwise if a fluid node is near a surface or domain boundary, the weights are higher than the bulk. This heuristic approach is essential when mesh coarsening is performed. It allows for migration of the computational load away from the processors that own nodes near surfaces. However, it is only modestly successful in balancing the work between processors with respect to checking surfaces and boundary conditions.

Figure 9 demonstrates the parallel scaling of the code. This figure shows three two-dimensional calculations where the domain size is $6\sigma \times N_{\text{proc}}\sigma$ in size, where N_{proc} is the number of processors used for the calculation and ranges from 1 to 512. The timings were performed on the Sandia–Intel Tflops (ASCI-Red) computer, which is composed of 333-MHz Pentium processors. The three curves show the parallel scaling behavior of the different algorithms described in the previous section.

All three algorithms scale well to a large number of processors, though three different behaviors are found for small numbers of processors in Fig. 9. When the exact Jacobian is used with no mesh or Jacobian coarsening ($\times$ in Fig. 9), the time per iteration in the solution

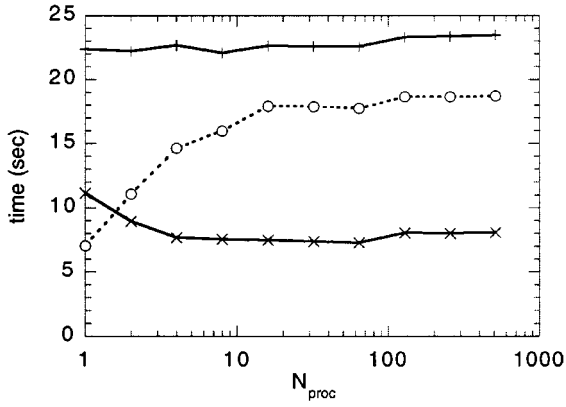

FIG. 9. The time per Newton iteration (load and linear solve) as a function of the number of processors, N_{proc}, used where the system size increases with N_{proc}. The various curves are ($\times$) $\Delta x = \Delta y = 0.1\sigma$, exact Jacobian, no mesh or Jacobian coarsening; ($+$) $\Delta x = \Delta y = 0.05\sigma$, exact Jacobian, three-zone mesh coarsening with break points at 2σ and 4σ away from a surface, and Jacobian coarsening using 0.2σ mesh everywhere; ($\circ$) same as ($+$) except minimal Jacobian applied. The number of Newton iterations needed for a solution is 6–7 for ($\times$), 10–12 for ($+$), and 13–15 for ($\circ$).

is nearly constant for $N_{\text{proc}} \geq 4$. For smaller N_{proc}, the solution time increases as the number of processors decreases. In contrast, when the exact Jacobian is applied with both mesh and Jacobian coarsening ($+$ in Fig. 9), the time per iteration is found to be nearly constant for all cases. Finally, when the minimal Jacobian is used along with mesh and Jacobian coarsening ($\circ$ in Fig. 9), the time per iteration is nearly constant when $N_{\text{proc}} \geq 16$. In this case, the time per fill increases with *increasing* N_{proc} when $N_{\text{proc}} \leq 16$.

These different behaviors highlight the competing effects that control parallel scaling. In the first case, the Jacobian fill dominates over the linear solver, and so the overall scaling reflects the behavior of the fill. The initial decrease in time with increasing processors results from a decrease in boundary checking on a per processor basis as the domain size increases. In the second case, all boundary condition checking is done in the $\bar{\rho}$ calculation up front. Thus, the initial increase in solve time reflects the increased time needed for the linear solves as the system size is increased. In the third case, these two effects are of similar magnitude but opposite sign, and therefore the code appears to exhibit nearly perfect parallel scaling.

6. SUMMARY

In this paper we have presented the underlying algorithms for a novel DFT code for calculation of the properties of inhomogeneous fluids near complex heterogeneous surfaces that require 2D or 3D treatments. The nonlinear integral equations describing equilibrium are discretized on a uniform, rectangular mesh and the resulting system of coupled nonlinear equations are solved using Newton's method. Algorithms for using inexact Jacobian matrices and power-of-2 mesh coarsening away from the surfaces have been presented and demonstrated to greatly improve the speed and scaling of the algorithms. These algorithmic improvements make most 2D calculations and even small 3D problems feasible on desktop computers.

The code has been written to run on massively parallel computers and is shown to scale well up to 512 processors. By using the computational resources of parallel computers, detailed parametric studies of 2D models and large 3D calculations can now be performed. Further analysis of the method can be found in our companion paper, which addresses issues of precision in the method for the case of solvated polymers.

ACKNOWLEDGMENTS

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

REFERENCES

1. I. K. Snook and W. van Megen, Solvation forces in simple dense fluids, 1, *J. Chem. Phys.* **72**, 2907 (1980).
2. D. Henderson, Ed., *Fundamentals of Inhomogeneous Fluids* (Dekker, New York, 1992).
3. P. Tarazona and R. Evans, A simple density functional theory for inhomogeneous liquids: Wetting by gas at a solid liquid interface, *Mol. Phys.* **52**, 847 (1984).
4. Y. Rosenfeld, Free-energy model for the inhomogeneous hard-sphere fluid mixture and density-functional theory of freezing, *Phys. Rev. Lett.* **63**, 980 (1989).
5. E. Kierlik and M. L. Rosinberg, Free-energy density functional for the inhomogeneous hard-sphere fluid: Application to interfacial adsorption, *Phys. Rev. A* **42**, 3382 (1990); Density-functional theory for inhomogeneous fluids: Adsorption of binary-mixtures, *Phys. Rev. A* **44**, 5025 (1991).
6. R. Ohnesorge, H. Löwen, and H. Wagner, Density-functional theory of crystal fluid interfaces and surface melting, *Phys. Rev. E* **50**, 4801 (1994).
7. H. H. von Grünberg and R. Klein, Density functional theory of nonuniform colloidal suspensions: 3D density distributions and depletion forces, *J. Chem. Phys.* **110**, 5421 (1999).
8. T. K. Vanderlick, L. E. Scriven, and H. T. Davis, Molecular theories of confined fluids, *J. Chem. Phys.* **90**, 2422 (1989).
9. R. Evans, Fluids adsorbed in narrow pores: Phase-equilibria and structure, *J. Phys. Condens. Matter* **2**, 8989 (1990).
10. U. Marini Bettolo Marconi and F. van Swol, Structure effects and phase-equilibria of Lennard–Jones mixtures in a cylindrical pore: A nonlocal density-functional theory, *Mol. Phys.* **5**, 1081 (1991).
11. A. Gonzalez, J. A. White, F. L. Roman, S. Velsco, and R. Evans, Density functional theory for small systems: Hard-spheres in a closed spherical cavity, *Phys. Rev. Lett.* **79**, 2466 (1997).
12. E. Velasco and P. Tarazona, Prewetting at a solid–fluid interface via Monte-Carlo simulation: Comment, *Phys. Rev. A* **42**, 2454 (1990).
13. V. Talanquer and D. W. Oxtoby, Dynamical density-functional theory of gas–liquid nucleation, *J. Chem. Phys.* **100**, 5190 (1994).
14. L. J. Douglas Frink and F. van Swol, A molecular theory for surface forces adhesion measurements, *J. Chem. Phys.* **106**, 3782 (1997).
15. L. J. Douglas Frink and F. van Swol, Solvation forces between rough surfaces, *J. Chem. Phys.* **108**, 5588 (1998).
16. L. J. Douglas Frink and F. van Swol, Stress isotherms of porous thin materials: Theoretical predictions from a nonlocal density functional theory, *Langmuir* **15**, 3296 (1999).
17. L. J. Douglas Frink and A. G. Salinger, Wetting of a chemically heterogeneous surface, *J. Chem. Phys.* **110**, 5969 (1999).
18. J. R. Henderson, Statistical mechanics of patterned inhomogeneous fluid phenomena, *J. Phys. Condens. Matter* **11**, 629 (1999).
19. M. Schoen and D. J. Diestler, Ultrathin fluid films confined to a chemically heterogeneous slit-shaped nanopore, *Phys. Rev. E* **56**, 4427 (1997).

20. P. Röcken, A. Somoza, P. Tarazona, and G. Findenegg, 2-Stage capillary condensation in pores with structured walls: A nonlocal density-functional theory, *J. Chem. Phys.* **108**, 8689 (1998).
21. L. J. Douglas Frink and A. G. Salinger, Two- and three-dimensional nonlocal density functional theory for inhomogeneous fluids. II. Solvated polymers as a benchmark problem, *J. Comput. Phys.* **159**, 425 (2000).
22. J. R. Henderson and Z. A. Sabeur, Liquid-state integral-equations at high-density: On the mathematical origin of infinite-range oscillatory solutions, *J. Chem. Phys.* **97**, 6750 (1992).
23. S. A. Hutchinson, L. V. Prevost, R. S. Tuminaro, and J. N. Shadid, *Aztec User's Guide: Version 2.0*, Technical Report, Sandia National Laboratories, Albuquerque, NM, 1998.