Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 190 (2003) 184–200
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A new efficient method for density functional theory calculations of inhomogeneous fluids

Mark P. Sears *, Laura J.D. Frink

Sandia National Laboratories, Albuquerque, NM 87185, USA

Received 29 May 2002; received in revised form 21 April 2003; accepted 16 May 2003

Abstract

The accurate computation of the effects of solvation on chemical systems can be done using density functional theories (DFT) for inhomogeneous multicomponent fluids. The DFT models of interest are non-local theories which accurately treat hard-sphere fluid mixtures; attractive inter-particle potentials (Lennard–Jones) are added as perturbations. In this paper, we develop and demonstrate a new efficient method for an accurate non-local DFT. The method described here differs from previous work in the use of fast fourier transform (FFT) methods to carry out the convolutions. As with our previous real space work (J. Comput. Phys. 159(2) (2000) 407, 425), we demonstrate that the Fourier space approach can be solved with a Newton–GMRES approach; however, we now employ a very efficient matrix-free algorithm. A simple but effective preconditioner is presented. The method is demonstrated with calculations performed for one-, two-, and three-dimensional systems, including problems with single and multicomponent fluids. Timing comparisons with previous implementations are given.

1. Introduction

Density functional theories (DFTs) have been widely applied to study the behavior of confined fluids and fluids near interfaces [6]. Until very recently most of these studies have used very simple surfaces with substantial symmetry and uniform properties. Examples of simple confining geometries include smooth planar walls, cylindrical pores, and spherical cavities, which can all be reduced to one-dimensional problems. More recently, studies on patterned surfaces have begun to appear in the literature; however, most of these use simplified functionals or have been limited to two-dimensional problems [9–11]. We are interested in applying accurate non-local DFTs to complex systems such as solvated proteins. In this case the chemistry and geometry of the surface of interest is non-uniform. As a result, a full three-dimensional solution is required.

* Corresponding author.
E-mail addresses: mpsears@sandia.gov (M.P. Sears), ljfrink@sandia.gov (L.J.D. Frink).

0021-9991/$ - see front matter
doi:10.1016/S0021-9991(03)00270-5

We have previously developed a real space code [1,2] that is based on a form of DFT originally derived by Rosenfeld [12]. That code used massively parallel computers and a variety of specialized algorithms to speed up computation of a Jacobian matrix based on the rather long-range integration stencils that result from the non-local density functionals of interest [1,2]. While a variety of novel algorithms were implemented to make three-dimensional protein systems accessible, the computational requirements are nevertheless substantial.

In this paper, we detail a new Fourier space based approach we have developed for solving the DFTs of interest. As with our previous work, we use a Newton solver for robustness and quadratic convergence guarantees. In contrast to our previous work, we have implemented a matrix-free GMRES approach. Below, we discuss the numerical implementation of the Fourier space method, compare Fourier space and real space methods, and demonstrate the power of the Fourier space approach for solving large three-dimensional problems.

2. The free energy functional

We briefly review the systems of equations solved in the DFT model of a fluid. The basic approach of the model is to consider a functional which accurately represents a fluid composed of a mixture of hard-sphere components and then to add attractive potentials as perturbations. The solution to the model is given as a variational minimum of the grand potential Ω, which is a functional of the density ρ, temperature T, and chemical potential μ

$$\Omega[\rho_i(\mathbf{r}), T, \mu] = F_{\mathrm{id}}[\rho_i(\mathbf{r}), T, \mu] + F_{\mathrm{hs}}[\rho_i(\mathbf{r})] + F_{\mathrm{u}}[\rho_i(\mathbf{r})] + \int \mathrm{d}^3\mathbf{r} \sum_i \rho_i(\mathbf{r})\left(V_i^{\mathrm{ext}}(\mathbf{r}) - \mu_i\right), \tag{1}$$

where the subscript i defines the fluid component and $V_i^{\mathrm{ext}}(\mathbf{r})$ is a one-body external field. The inhomogeneity of the fluid density is a direct result of the spatial variation of the external field.

The first three terms in Eq. (1) are the the ideal gas, hard sphere, and two-body perturbative contributions to the free energy, respectively. The ideal gas contribution is

$$F_{\mathrm{id}} = k_{\mathrm{B}}T \int \mathrm{d}^3\mathbf{r} \sum_i \rho_i\left(\ln\left(\rho_i \Lambda_i^3\right) - 1\right), \tag{2}$$

where Λ is the de Broglie length. This expression is invariant under a simultaneous shift in the chemical potential and a rescaling of Λ, and by doing this we can drop Λ out of the problem. Either the chemical potential or the bulk density can be used to define the bulk fluid state point.

Rosenfeld [12] form for the hard-sphere excess free energy is given as a local functional of non-local (weighted) densities

$$F_{\mathrm{hs}} = \int \mathrm{d}^3\mathbf{r}\, \Phi(n_\gamma), \tag{3}$$

where the weighted densities n_γ are given by the convolutions

$$n_\gamma(\mathbf{r}) = \sum_i \int \mathrm{d}^3\mathbf{r}'\, \omega_{i\gamma}(\mathbf{r} - \mathbf{r}')\rho_i(\mathbf{r}'). \tag{4}$$

The weight functions $\omega_{i\gamma}$ can be either scalar or vector. They depend only on so-called fundamental measures of the hard-sphere particles in the fluid: the volume, surface area, and radius of the particles. The vector weight functions probe the density gradient as well. Note that, in this paper we use lower case Italic

indices for density components and lower case Greek indices to identify weight functions. The detailed form for Φ and the weight functions are presented in Appendix A, as is the fast fourier transform (FFT) method for performing convolutions.

The two-body perturbative contribution is taken to be a strict mean field approximation

$$F_{\mathrm{u}} = \frac{1}{2}\int\int \mathrm{d}^3\mathbf{r}\,\mathrm{d}^3\mathbf{r}' \sum_{ij} \rho_i(\mathbf{r})\rho_j(\mathbf{r}')U_{ij}(\mathbf{r}-\mathbf{r}'), \tag{5}$$

where the attractive potential U is described in more detail in Appendix C.

3. A matrix-free Newtons method

We now describe a solution scheme to find the desired stationary point of the grand potential functional. We define for each independent field f (i.e., density component) a *residual* field r_f given by the functional equation

$$r_f = \frac{\delta\Omega}{\delta f}. \tag{6}$$

Clearly, a stationary point is defined by $r_f = 0$ at each point in space for all the fields. In addition to solving these residual equations, boundary conditions must be applied. In contrast to our real space method [1], where implementation of a variety of boundary conditions (periodic, reflective, homogeneous, and inhomogeneous) was straightforward, the FFT based convolution method presented here is inherently periodic, and without modification can only handle problems with periodic boundary conditions.

A Picard method could be implemented directly from the residual defined above. The residual for component i can be separated into ideal gas and excess parts

$$r_i = r_i^{\mathrm{id}} + r_i^{\mathrm{ex}}, \tag{7}$$

and at the solution where $r_i = 0$, we have

$$\rho_i(\mathbf{r}) = \exp\left(-\frac{V_i^{\mathrm{ext}}(\mathbf{r}) - \mu_i + r_i^{\mathrm{ex}}(\mathbf{r})}{kT}\right). \tag{8}$$

To implement the Picard scheme we simply iterate this equation until self consistency is achieived. In practice the convergence rate can be very slow, depending on the specific physics included in the functional and the state point of interest. It has been demonstrated that the convergence of a Newton's method is much faster, but this approach is costly due to the need for computing the Jacobian matrix [1].

The Jacobian of the system of equations defined by Eq. (6) is given by the functional derivatives

$$J_{ff'} = \frac{\delta r_f}{\delta f'} = \frac{\delta^2\Omega}{\delta f\,\delta f'}. \tag{9}$$

This quantity is defined for each pair of fields f, f' and each pair of points in space x, x'. In the following we treat the spatial coordinates implicitly.

Suppose we are close to a solution, then we can write each field f as the solution $f^{\star}$ plus a small error ϵ_f and then to second order, we have

$$r_f[f] = r_f[f^{\star}] + \sum_{f'} J_{ff'}\epsilon_{f'}, \tag{10}$$

but of course $r_f[f^{\star}] = 0$ and so ϵ_f is the solution to

$$r_f[f] = \sum_{f'} J_{ff'}\epsilon_{f'}, \tag{11}$$

and the Newton iteration procedure may be written as the replacement of f by $f - \epsilon_f$.

We describe Newton's iteration in this way to point out that the objective is to solve the above equation for the change in the fields. The inverse Jacobian operator is not needed (may not even be well defined), and the Jacobian itself is not required. All that is required is a method for solving the above linear system. In this paper, we show that a matrix-free method works quite well: such a method solves the equation using an operational definition of the Jacobian *without storing any part of J*. In other words we assume only the ability to compute the effect of the Jacobian operating on a set of fields. The Jacobian operator can be used directly in an iterative scheme such as GMRES.

As we show in detail below, a powerful result emerges from this matrix-free approach: the computation of the operational definition of the Jacobian involves the *same amount of work as the computation of the residual*, *within a small constant factor*! This means that Newton's method can be implemented with essentially the same computational effort as a Picard iteration with the potential of converging much more quickly.

The method implemented in our real space code [1] contrasts with the current algorithm primarily in that the real space code computes and saves elements of the Jacobian matrix in a sparse format. The computation is done with finite element techniques, i.e., using finite element shape functions, and the assembled matrix elements therefore form a nodal finite difference stencil. The Jacobian equation is then solved by a sparse matrix package [7] which has a number of solver and preconditioning options. At this level both codes solve the same Newton iteration and as we show later the convergence of the codes with respect to Newton iteration is essentially the same for both. At the lower level the real space code computes and saves a sparse matrix and the matrix vector multiplications are performed by the sparse matrix package, while the matrix-free FFT method implements the matrix–vector multiplications needed for a sparse solve directly in terms of the physical operators. For a true comparison of either method with the Picard approach, we should compare total the number of matrix–vector multiplications (however performed) with the number of Picard iterations.

3.1. Overview of the algorithm

The algorithm basically consists of an outer loop which is the Newton's iteration. Inside this loop there are three steps: computing the residual, calling the linear solver, and finally updating the solution; this loop is repeated until the residual norm is below some tolerance or a maximum number of steps is reached. The linear solver in turn computes the Jacobian operator some number of times in order to solve the above equation. In the remainder of this section we outline the method for computing the residual and then show the related techniques used to compute the Jacobian operator.

3.2. Computing the residuals

In this paper, there is one residual field for each density component since we do not include electrostatics or other fields. If we did add electrostatics then there would be a residual field corresponding to the electrostatic potential and perhaps additional fields and residuals corresponding to polarization, but further discussion is beyond the scope of this paper.

Following the development above, we discuss different physical contributions to the residual independently with the understanding that they will be summed. We make the assumption that the physical fields, potentials, and residuals are described on a simple regular real space mesh, which is used because it makes

the Fourier transform algorithms very fast. On the other hand, this choice precludes the use of a mesh which is graded or adaptive in regions where the solution is more or less rapidly varying. The Fourier mesh described below is only present in order to compute convolutions quickly.

The ideal gas contribution to the residual for density component i is

$$kT\ln(\rho_i(\mathbf{r})) + V_i^{\text{ext}}(\mathbf{r}) - \mu_i, \tag{12}$$

which can be evaluated directly point by point on the real-space mesh.

The functional derivative of the Rosenfeld form for the hard-sphere excess free energy is

$$\left.\frac{\delta F_{\text{hs}}}{\delta\rho_i}\right|_{\mathbf{r}} = \int \mathrm{d}^3\mathbf{r}' \sum_\gamma \left.\frac{\partial\Phi}{\partial n_\gamma}\right|_{\mathbf{r}'} \frac{\delta n_\gamma(\mathbf{r}')}{\delta\rho_i(\mathbf{r})} = \int \mathrm{d}^3\mathbf{r}' \sum_\gamma \left.\frac{\partial\Phi}{\partial n_\gamma}\right|_{\mathbf{r}'} \omega_{i\gamma}(\mathbf{r}-\mathbf{r}'). \tag{13}$$

Note the order of $\mathbf{r}$, $\mathbf{r}'$ in this expression compared with Eq. (4), which is just a convolution of $\partial\Phi/\partial n_\alpha$ with the weight functions $\omega_{i\gamma}$. For even (i.e., scalar) weight functions the order is irrelevant, but for the vector weight functions the change in order results in a change of sign. We can take this into account during the process of convolution with the FFT method by taking the complex conjugate of the weight function in Fourier space when the residual is constructed from $\partial\Phi/\partial n_\alpha$. This has no effect on the even weight functions, whose FT is real, and changes the sign of the odd weight functions, whose FT is imaginary. Note that the total amount of work required to compute the residual is $N_{\text{w}} + N_{\text{c}}$ convolutions, where N_{w} is the number of weight functions and N_{c} is the number of density components.

Finally, the contribution to the residual of the attractive perturbations is

$$\left.\frac{\delta F_{\text{u}}}{\delta\rho_i}\right|_{\mathbf{r}} = \int \mathrm{d}^3\mathbf{r}' \sum_j \rho_j(\mathbf{r}')U_{ij}(\mathbf{r}-\mathbf{r}'). \tag{14}$$

Note that this is a convolution of the potential function U with the density, and can easily be treated with the FFT method described in Appendix B.

3.3. Computing the operational Jacobian

We now show how to compute the operational definition of the Jacobian, that is the action of the Jacobian on a field. In more detail we are computing the quantities

$$y_i(\mathbf{r}) = \int \mathrm{d}^3\mathbf{r}' \sum_j J_{ij}(\mathbf{r},\mathbf{r}')x_j(\mathbf{r}') = \int \mathrm{d}^3\mathbf{r}' \sum_j \frac{\delta^2\Omega}{\delta\rho_i(\mathbf{r})\,\delta\rho_j(\mathbf{r}')}x_j(\mathbf{r}'). \tag{15}$$

For some input field x, which will come from the iterative scheme that we use to solve $J\epsilon = r$. We do this without constructing or saving any part of J, thus the method is matrix-free.

The ideal gas contribution to $y_i(\mathbf{r})$ is simply evaluated pointwise as

$$\frac{kT}{\rho_i(\mathbf{r})}x_i(\mathbf{r}). \tag{16}$$

In order to write down the hard-sphere contribution to the operational Jacobian we define

$$\Phi_{\alpha\beta}(\mathbf{r}) = \left.\frac{\partial^2\Phi}{\partial n_\alpha \partial n_\beta}\right|_r. \tag{17}$$

Expressions for $\Phi_{\alpha\beta}(\mathbf{r})$ are given in Appendix A. Then the hard-sphere contribution to the Jacobian becomes

$$\left.\frac{\delta^2 F_{\mathrm{hs}}}{\delta\rho_i\delta\rho_j}\right|_{r,r'} = \int \mathrm{d}^3\mathbf{r}'' \sum_{\alpha\beta} \Phi_{\alpha\beta}(\mathbf{r}'') \frac{\delta n_\alpha(\mathbf{r}'')}{\delta\rho_i(\mathbf{r})} \frac{\delta n_\beta(\mathbf{r}'')}{\delta\rho_j(\mathbf{r}')} \tag{18}$$

$$= \int \mathrm{d}^3\mathbf{r}'' \sum_{\alpha\beta} \Phi_{\alpha\beta}(\mathbf{r}'')\omega_{i\alpha}(\mathbf{r}''-\mathbf{r})\omega_{j\beta}(\mathbf{r}''-\mathbf{r}'), \tag{19}$$

where the second derivative of Φ only depends on a single coordinate $\mathbf{r}''$, because Φ is a local function of the weighted densities.

Efficient computation of the hard-sphere contribution to the operational Jacobian relies on noticing that it can be evaluated via two successive sets of convolutions. To this end we define two intermediate quantities. The first is a convolution of the input field x with the weight functions ω

$$t_\beta(\mathbf{r}'') = \int \mathrm{d}^3\mathbf{r}' \sum_j \omega_{j\beta}(\mathbf{r}''-\mathbf{r}')x_j(\mathbf{r}'). \tag{20}$$

We also define

$$z_\alpha(\mathbf{r}'') = \sum_\beta \left.\frac{\partial^2\Phi}{\partial n_\alpha \partial n_\beta}\right|_{\mathbf{r}''} t_\beta(\mathbf{r}''). \tag{21}$$

We now see that then we have the hard-sphere contribution to $y_i(r)$ given as a convolution of z_α

$$\int \mathrm{d}^3\mathbf{r}'' \sum_\alpha \omega_{i\alpha}(\mathbf{r}''-\mathbf{r})z_\alpha(\mathbf{r}''). \tag{22}$$

Thus $y(\mathbf{r})$ is computed by first evaluating the convolutions in Eq. (20) followed by a second set of convolutions in Eq. (22). Each requires $N_c + N_w$ individual convolutions for a total of $2N_c + 2N_w$. Note that the order of the variables in the weight functions is reversed in the two convolutions.

We compute and save the quantities $\Phi_{\alpha\beta}(\mathbf{r})$ at each location in the real-space mesh once at the beginning of each Newton's iteration. The cost of computing the hard-sphere contribution to the Jacobian operator is therefore dominated by the two convolutions described here. The total number of FFT operations is $2N_w + 2N_c$ per Jacobian operator computation, or twice the number of FFT operations as were needed for the residual computation. Note also that when N_c is small the cost is dominated by N_w, so the cost of solving for mixtures with a small number of components is hardly larger than that of solving for a single component.

Finally, the attractive contribution to the operational Jacobian is simply given by the convolution of the field x with the interparticle potential U. This becomes a term of the form

$$y_i(\mathbf{r}) = \int \mathrm{d}^3\mathbf{r}' \sum_j U_{ij}(\mathbf{r}-\mathbf{r}')x_j(\mathbf{r}'). \tag{23}$$

3.4. Iterative solution with GMRES and preconditioning

GMRES [14] is a powerful method for solving large systems of linear equations iteratively. The method only requires an implementation of the matrix–vector product, which we provided above. Since the Jacobian is defined in a way that explicitly reveals the fact that it is symmetric, we might think that a simpler method such as conjugate gradients (CG) would be sufficient. CG requires that the matrix also be positive definite, however, and this is not so easy to demonstrate. GMRES has a great deal of robustness that CG

lacks, and in addition we have intentions to modify this program for situations where the equations are not symmetric, for example, when we treat steady-state transport [4] or add a state-following algorithm that accesses unstable parts of the phase diagram [3].

GMRES and CG are both Krylov-space iterative methods. In GMRES, as vectors are accumulated they are orthogonalized against all the previously accumulated vectors; this provides much of the stability and robustness relative to CG, which uses a three term recursion relation to implicitly maintain orthogonality. The memory required for GMRES is $N(S+2)$, where N is the number of independent variables and S is the maximum number of iterations. The method can be restarted, so a fixed number can be chosen for S. Nevertheless, this storage can dominate the required storage for the problem as a whole. Beyond this description we treat GMRES as a black box and refer the interested reader to the published literature on the method.

Like other Krylov-space iterative methods, GMRES can benefit greatly from a good preconditioner. We consider the development of preconditioners to be an open problem for this application. Of course, many of the preconditioners that are available for finite element methods rely on the existence of explicit matrix elements, which our matrix-free method does not make available. So these preconditioners cannot be used and we are forced to look for physics-based preconditioners.

The approach currently used in the code is very simple. A class of preconditioners is defined by building an approximate inverse to J that is easily computed. If M is such an inverse then we can solve $JMM^{-1}x = y$ instead of $Jx = y$, by solving the modified equation $JMx' = y$ and then computing $x = Mx'$. Here we construct a local function (diagonal matrix) that interpolates between the ideal gas limit and the bulk state point. The preconditioner is

$$\tilde{x}_i = M^{ij}x_j, \tag{24}$$

where for a single component

$$M = \frac{\rho}{J_0 - \rho(kT/\rho_0) + kT} \tag{25}$$

and J_0 is the Jacobian for a constant-density fluid at the bulk state point ρ_0. This preconditioner interpolates between the bulk fluid state point far from any wall to the ideal gas behavior where the density is small.

A better preconditioner might involve solving a multilevel problem. In this case we would solve the DFT problem on a coarse mesh and then interpolate that solution to the fine mesh. Note that we can do a very good job of interpolation using the FFT method, and if the coarse mesh has twice the spacing then the effort needed to solve on the coarse mesh is 1/8 that on the fine mesh.

4. Example problems

In this section we give results for several simple one-, two-, and three-dimensional cases. We begin with sample problems where the boundary condition is that of a hard wall, hard cylinder, or hard sphere. A general hard surface (defined as the boundary of some volume) boundary condition can be defined by an external field with

$$V(\mathbf{r}) = \begin{cases} \infty & \text{inside,} \\ 0 & \text{outside.} \end{cases} \tag{26}$$

Note that for all grid points or nodes inside the volume, where $V(\mathbf{r}) = \infty$ the density $\rho(\mathbf{r})$ can be rigorously set to zero. However, when integrating (e.g., computing non-local densities) through the discontinuity

defined by the surface one should include contributions from elements on the fluid side of the interface only in order to predict surface densities that will satisfy known sum rules (e.g., $\sum_i \rho_i = \beta P$). If one allows for interpolation between the density at the surface node and the density for the first node inside the volume then the sum rules will not be satisfied except in the limit that the grid spacing goes to zero. However, all other points in the density distribution will be affected very little. Since we are ultimately interested in systems described by continuous potentials, we have elected not to implement the required code for accurately dealing with true hard potentials in the Fourier algorithm (see [1] for more details). Therefore, the real space results shown below for comparison were done allowing a linear interpolation between the surface nodes of the mesh and the first element inside the volume.

Figs. 1–3 present a direct comparison of the algorithm presented in this paper with our previous finite element real space implementation. These figures show the fluid density distribution in a periodic array of planar walls, parallel cylinders, and spheres, respectively. In all cases the agreement is extremely good. The bulk state of the fluid is defined by a bulk density of $\rho = .8785$ corresponding to a packing fraction of $\eta = .46$, which is the same state point used in [8]. The wall thickness (radius) is 1σ. For the hard cylinder and hard sphere cases we plot the density along a line extending radially from the surface. The problems were run in a periodic volume of size $L = 12.8\sigma$ for the one- and two-dimensional problems and $L = 6.4\sigma$ for the three-dimensional problem. The mesh cell size for the hard wall and hard cylinder cases is $h = .1\sigma$ and is doubled for the sphere case. The points are from the FFT based calculations and the lines from our finite element calculations; the results in all three cases are almost identical. The inset to Fig. 2 shows a contour plot of the density around the cylinder. This contour plot shows that the staircase representation of the cylinder does not induce significant staircase artifacts in the computed density. Rather, uniform cylindrical density bands are observed.

To further illustrate applications of our new algorithm we include three additional one-dimensional cases. The first shows an example of a two component hard-sphere fluid near a hard wall (see Fig. 4). The bulk state is chosen with the bulk densities of (.0260, .0104) for components with hard-sphere radii of (.5, 1.5), which regenerates the results of [13].

Fig. 5 shows a calculation of a hard-sphere fluid at a Lennard–Jones 9-3 wall, which can be compared with [8]. Details of the wall potential are described in Appendix C. Note the large peak next to the wall with

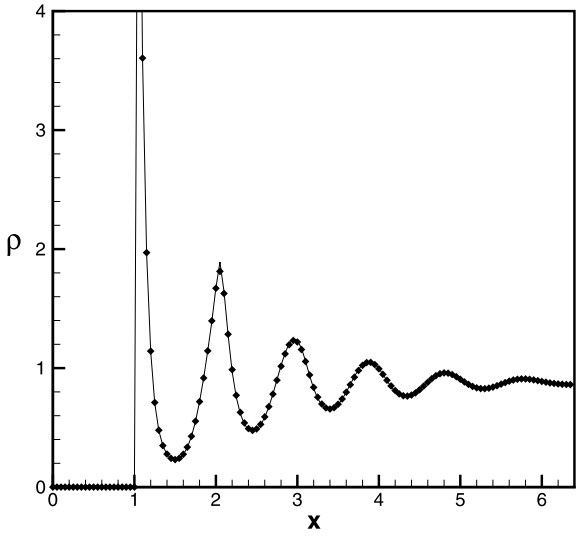

Fig. 1. Density profile at a hard wall. The bulk state is $\rho = .8785$ (packing fraction .46).

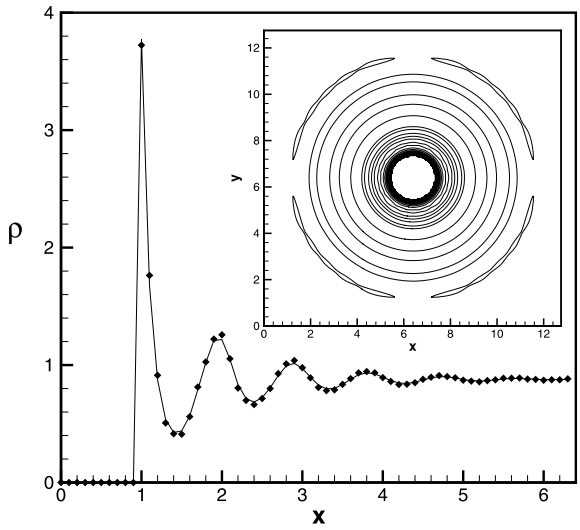

Fig. 2. Density profile at a hard cylinder. State is same as Fig. 1.

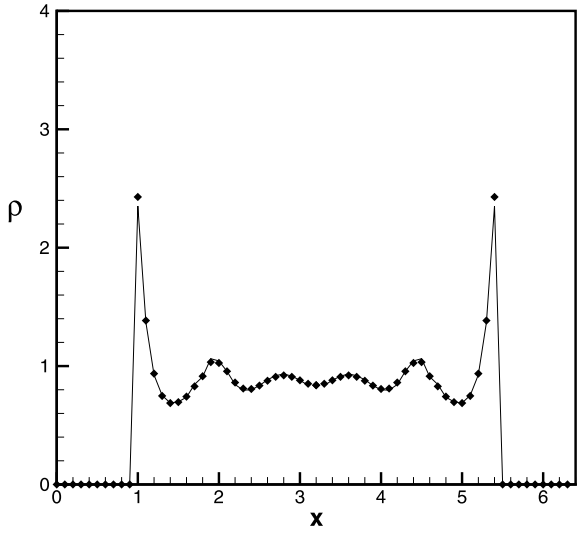

Fig. 3. Density profile around a hard sphere. State is same as Fig. 1.

a very low minimum next to it. The large peak corresponds to a monolayer of fluid lying in the minimum of the 9-3 potential.

Fig. 6 is the most interesting case. We show what happens when a relatively high density fluid is in contact with a Lennard–Jones wall. In this case the fluid is described by the hard-sphere terms as well as a perturbative Lennard–Jones interaction and the state point is chosen to lie in the liquid regime. The bulk density is set to .75 in units where the hard-sphere diameter is 1. The fluid-wall interaction and the fluid–fluid interaction potential are described in Appendix C. In this case large amplitude density oscillations are seen that persist a long distance into the fluid, and again we see that the two codes yield identical results.

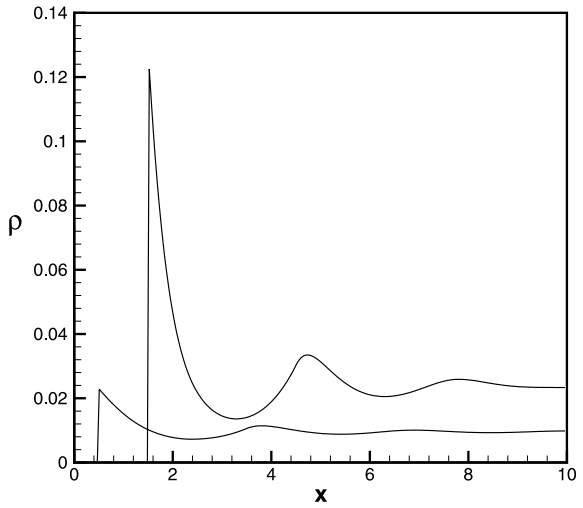

Fig. 4. Density profiles for hard-sphere mixture. Compare with [13] Fig. 2a. Bulk density of component 1 (hard-sphere radius .5) is .0260 and bulk density of component 2 (hard-sphere radius 1.5) is .0104.

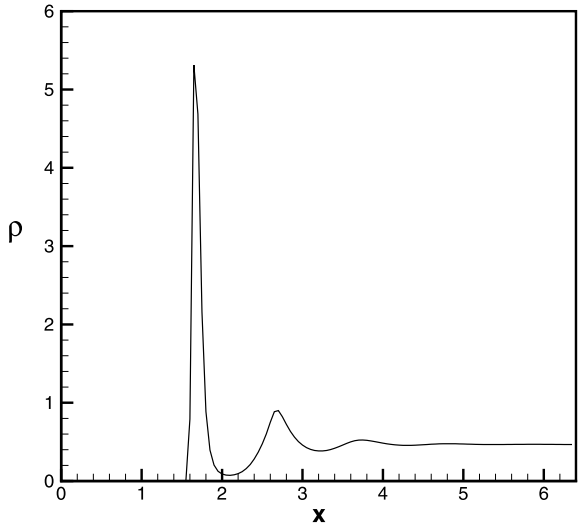

Fig. 5. Density profile for hard-sphere fluid at LJ 9-3 wall. Compare with [8] Fig. 3.

5. Convergence and performance

In this section we compare the convergence and performance of an implementation of the algorithm described in this paper with that of our real space finite element code. Above we showed that the results of the two codes are essentially identical, here we note that there are three main issues of convergence and performance:

- Comparison of convergence of the two codes.
- Comparison of the speed (time to solution) of the two codes.
- Comparison of memory required by the two codes.

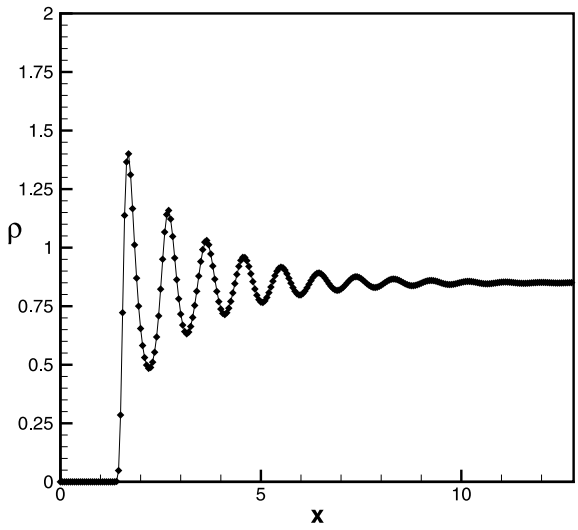

Fig. 6. Density profile for LJ fluid at LJ 9-3 wall. See Appendix C for details on wall and fluid interaction potentials.

Convergence of the two codes is very similar, as is shown in Fig. 7. This plot shows how the residual norm changes as a function of Newton–GMRES iteration, showing the quadratic convergence expected for this algorithm. The residual norm is defined by the expression

$$\epsilon = \left(\int \mathrm{d}V |r|^2 \right)^{1/2}, \tag{27}$$

where r is the residual field. One interesting feature is that the convergence rate improves with increasing dimension. The apparent reason for this is that the curved walls lower the magnitude of the density oscillations in the solution; the effect can be seen by comparing the peak heights in Figs. 1–3. This means that the fluid in these cases is everywhere closer to the bulk state and therefore convergence is improved. For the

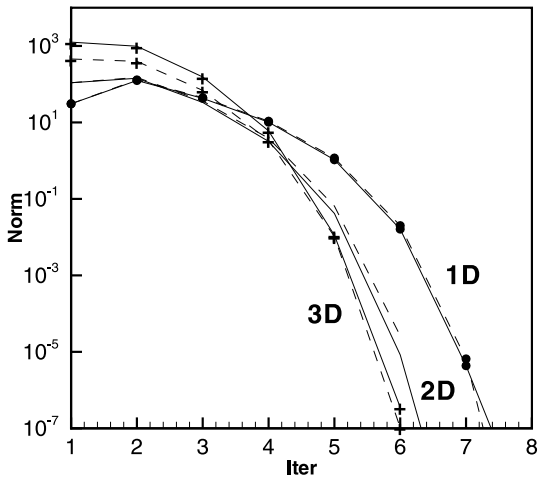

Fig. 7. Convergence vs Newton–GMRES iteration for one-, two-, three-dimensional problems. See Figs. 1–3.

one-dimensional case convergence requires eight iterations, in two dimensions seven, and in the three-dimensional case the algorithm reduces the residual norm by 10 orders of magnitude in only six iterations. In all cases the initial guess for the solution is just a constant density equal to the bulk density outside the wall and zero inside.

Although both codes use approximately the same number of Newton–GMRES iterations to solve the problem, the speed of the two codes is markedly different. The primary difference in performance is due to the following. In the finite element code the weight function convolutions are implemented as finite difference operators. Considering such an operator, we note that each weight function has a finite radius R, so there are a number $O(R^3/h^3)$ points in the finite difference operator in three dimensions, where h is the mesh spacing. The work required to perform a convolution is therefore $O(L^3/h^3)$ since there are L^3 mesh points. In comparison, the FFT method for performing the convolution is almost independent of the radius of the weight functions, so the work in this case is $O(L^3)$. Mesh refinement for the finite element code therefore has a cost $O(1/h^3)$ relative to the FFT method. For one dimension this relative cost is $O(1/h)$ and for two dimensions the relative cost is $O(1/h^2)$.

Fig. 8 shows the relative performance of the two methods running on the same platform (Intel Pentium 4 at 1.3 GHz), for a scaled set of three-dimensional problems. We have plotted time per Newton iteration as a function of the number of points in the mesh, N, and the scaling of the graph is logarithmic. Note that all the algorithms would scale linearly with number of mesh points if the size of the system were increased. However, when the number of points is increased by making the mesh more dense as in Fig. 8, there are considerable differences between the algorithms. The performance of the FFT method is approximately $N^{1.2}$, where the performance of the real space method is between $N^{2.2}$ and $N^{1.6}$ depending on the specific algorithm applied. More specifically, the $N^{2.2}$ scaling is found for algorithms dominated by the Jacobian computation while the $N^{1.6}$ scaling is found for algorithms dominated by the GMRES solve [5]. For comparison, a scaling curve is also shown for the faster real space algorithm on 100 processors of the CPlant commodity cluster at Sandia National Labs. Note that the CPlant processors are slower than the Intel processors used for single processor timings by about a factor of 4. Fig. 8 clearly demonstrates that our new matrix-free algorithms can be used to compute large $O(10^6)$ *unknowns* calculations on a single processor workstation in a modest O(1 h) amount of time. When these new algorithms are parallelized and further optimized we expect a further two orders of magnitude speedup.

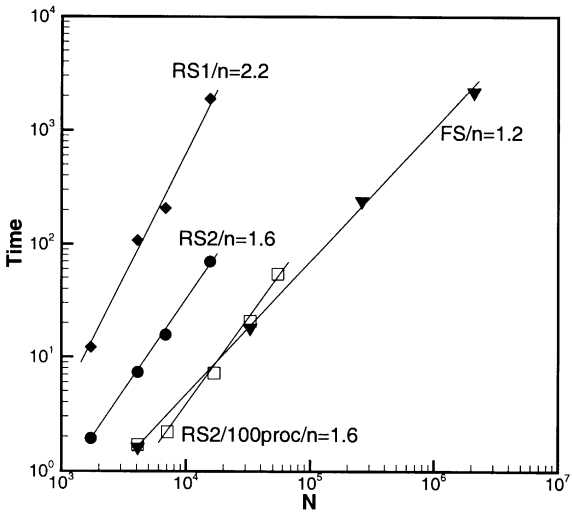

Fig. 8. Performance scaling with respect to mesh density (fixed unit cell size) for a three-dimensional problem.

Table 1
Memory usage in gigabytes (GB)

h	$N \times 10^6$	RS1 (GB)	FS (GB)
.2	0.26	1	.25
.1	2.10	70	2
.05	16.7	4500	16

Finally, Table 1 shows a comparison of the approximate memory in gigabytes (GB) required by the two codes for a typical one-component fluid in a cubical box with side $L = 12.8\sigma$ at various mesh densities (h in units of σ). This table compares the algorithm presented here (FS) with the RS1 algorithm in Fig. 8 for the case of a hard-sphere fluid. Clearly, there is a very significant difference between the two codes with respect to required memory.

6. Conclusions

We have presented an efficient algorithm for treating DFT in complex three-dimensional geometries. The Newton–GMRES solution scheme is capable of solving the highly non-linear optimization problem presented by the minimization of complex DFT model functionals in remarkably few iterations and with quadratic convergence, as shown above. The development of good preconditioners for the solve is still an open question. The efficient FFT based method of computing the residual and the Jacobian operator is based on analysis of the physical operators. The method works without storing a matrix and is both faster than the finite element method and uses significantly less memory.

We have shown that the new matrix-free algorithm accurately reproduces the results of our previous real space finite element code as well as results from the literature. The new algorithm is thus a powerful technique for the treatment of complex fluid mixtures in one-, two-, and three-dimensional inhomogeneous systems.

Appendix A. Rosenfeld functional

Here we give for reference the Rosenfeld functional for the excess hard-sphere free energy, its first and second derivatives, the weight functions and their Fourier transforms.

For a fluid component with hard-sphere radius R there are four scalar and two vector weight functions, or 10 weight functions altogether for each fluid component. These are

$$\omega_0(r) = \frac{\delta(r-R)}{4\pi R^2}, \quad \omega_1(r) = R\omega_0(r), \quad \omega_2(r) = 4\pi R^2\,\omega_0(r), \quad \omega_3(r) = \theta(r-R),$$
$$\omega_{1\mathrm{v}}(\mathbf{r}) = \frac{\hat{\mathbf{r}}}{4\pi R}\,\delta(r-R), \quad \omega_{2\mathrm{v}}(\mathbf{r}) = 4\pi R\omega_{1\mathrm{v}}(\mathbf{r}), \tag{A.1}$$

where R is the hard-sphere radius of the component. The weight functions $\omega_{1\mathrm{v}}$, $\omega_{2\mathrm{v}}$ and their associated densities $n_{1\mathrm{v}}$, $n_{2\mathrm{v}}$ are treated as vector quantities. The Fourier transforms of these functions are:

$$\tilde{\omega}_0(k) = \frac{\sin(kR)}{kR}, \quad \tilde{\omega}_1(k) = R\tilde{\omega}_0(k), \quad \tilde{\omega}_2(k) = 4\pi R^2\tilde{\omega}_0(k), \quad \tilde{\omega}_3(k) = 4\pi R^3\,\frac{\sin(kR) - kR\cos(kR)}{(kR)^3},$$
$$\tilde{\omega}_{1\mathrm{v}}(\mathbf{k}) = -\mathrm{i}\mathbf{k}\,\frac{\tilde{\omega}_3(k)}{4\pi R}, \quad \tilde{\omega}_{2\mathrm{v}}(\mathbf{k}) = 4\pi R\tilde{\omega}_{1\mathrm{v}}(\mathbf{k}), \tag{A.2}$$

where $k = |\mathbf{k}|$. Note that $\omega_{1\mathrm{v}}(\mathbf{r})$ is $-\frac{1}{4\pi}\nabla\omega_3(r)$.

The Rosenfeld functional F_{hs} is given by the following expression:

$$F_{\text{hs}} = \int \mathrm{d}^3\mathbf{r}\,(\Phi_{\text{s}} + \Phi_{\text{v}}), \tag{A.3}$$

where the scalar and vector contributions Φ_{s} and Φ_{v} are local functions of the weighted densities n_α:

$$\begin{aligned}
\Phi_{\text{s}} &= -n_0 \ln(1-n_3) + \frac{n_1 n_2}{1-n_3} + \frac{1}{24\pi}\frac{n_2^3}{(1-n_3)^2}, \\
\Phi_{\text{v}} &= -\frac{n_{1\text{v}}\cdot n_{2\text{v}}}{1-n_3} - \frac{1}{8\pi} n_2 \frac{n_{2\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^2}.
\end{aligned} \tag{A.4}$$

The first derivatives of Φ are:

$$\begin{aligned}
&\frac{\partial\Phi}{\partial n_0} = -\ln(1-n_3), \quad \frac{\partial\Phi}{\partial n_1} = \frac{n_2}{1-n_3}, \\
&\frac{\partial\Phi}{\partial n_2} = \frac{n_1}{1-n_3} + \frac{1}{8\pi}\frac{n_2^2}{(1-n_3)^2} - \frac{1}{8\pi}\frac{n_{2\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^2}, \\
&\frac{\partial\Phi}{\partial n_3} = \frac{n_0}{1-n_3} + \frac{n_1 n_2}{(1-n_3)^2} + \frac{1}{12\pi}\frac{n_2^3}{(1-n_3)^3} - \frac{n_{1\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^2} - \frac{1}{4\pi} n_2 \frac{n_{2\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^3}, \\
&\frac{\partial\Phi}{\partial n_{1\text{v}}} = -\frac{n_{2\text{v}}}{1-n_3}, \quad \frac{\partial\Phi}{\partial n_{2\text{v}}} = -\frac{n_{1\text{v}}}{1-n_3} - \frac{1}{4\pi} n_2 \frac{n_{2\text{v}}}{(1-n_3)^2}.
\end{aligned} \tag{A.5}$$

The 21 non-vanishing second derivatives (modulo symmetry) are:

$$\begin{aligned}
&\frac{\partial^2\Phi}{\partial n_0 \partial n_3} = \frac{1}{1-n_3}, \quad \frac{\partial^2\Phi}{\partial n_1 \partial n_2} = \frac{1}{1-n_3}, \quad \frac{\partial^2\Phi}{\partial n_1 \partial n_3} = \frac{n_2}{(1-n_3)^2}, \quad \frac{\partial^2\Phi}{\partial n_2 \partial n_2} = \frac{1}{4\pi}\frac{n_2}{(1-n_3)^2}, \\
&\frac{\partial^2\Phi}{\partial n_2 \partial n_3} = \frac{n_1}{(1-n_3)^2} + \frac{1}{4\pi}\frac{n_2^2}{(1-n_3)^3} - \frac{n_{2\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^3}, \quad \frac{\partial^2\Phi}{\partial n_2 \partial n_{2\text{v}}} = -\frac{1}{4\pi}\frac{n_{2\text{v}}}{(1-n_3)^2}, \\
&\frac{\partial^2\Phi}{\partial n_3 \partial n_3} = \frac{n_0}{(1-n_3)^2} + \frac{n_1 n_2}{(1-n_3)^3} + \frac{1}{4\pi}\frac{n_2^3}{(1-n_3)^4} - 2\frac{n_{1\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^3} - \frac{3}{4\pi} n_2 \frac{n_{2\text{v}}\cdot n_{2\text{v}}}{(1-n_3)^4}, \\
&\frac{\partial^2\Phi}{\partial n_3 \partial n_{1\text{v}}} = -\frac{n_{2\text{v}}}{(1-n_3)^2}, \quad \frac{\partial^2\Phi}{\partial n_3 \partial n_{2\text{v}}} = -\frac{n_{1\text{v}}}{(1-n_3)^2} - \frac{1}{2\pi} n_2 \frac{n_{2\text{v}}}{(1-n_3)^3}, \\
&\frac{\partial^2\Phi}{\partial n_{1\text{v}} \partial n_{2\text{v}}} = -\frac{1}{1-n_3}\mathbf{I}, \quad \frac{\partial^2\Phi}{\partial n_{1\text{v}} \partial n_{2\text{v}}} = -\frac{1}{4\pi}\frac{n_2}{(1-n_3)^2}\mathbf{I},
\end{aligned} \tag{A.6}$$

where $\mathbf{I}$ is the unit tensor.

Appendix B. FFT method for convolutions

If the density components are periodic then the convolutions can be computed rapidly with FFT operations. Let f be a periodic function in a (possibly non-orthogonal) unit cell defined by lattice vectors $\mathbf{l}_x$, $\mathbf{l}_y$, $\mathbf{l}_z$ then f can be represented with a Fourier series

$$f(\mathbf{x}) = \sum_{\mathbf{k}} \hat{f}_{\mathbf{k}}\, \mathrm{e}^{\mathrm{i}\mathbf{x}\cdot\mathbf{k}}, \tag{B.1}$$

where $\mathbf{k}$ is summed over points of the reciprocal space grid

$$\mathbf{k}(i_x, i_y, i_z) = i_x\mathbf{g}_x + i_y\mathbf{g}_y + i_z\mathbf{g}_z. \tag{B.2}$$

Here the reciprocal lattice vectors are defined as $\mathbf{g}_x = \frac{2\pi}{V}\mathbf{l}_y \times \mathbf{l}_z$, etc., where V is the volume of the unit cell $\mathbf{l}_x \cdot \mathbf{l}_y \times \mathbf{l}_z$, and the sum occurs over all integers i_x, i_y, i_z. If we truncate the representation for $|i_x| \leqslant N/2$, etc. then the discrete Fourier transform (implemented with the FFT algorithm) can be used to transform to and from the representation on a real space grid

$$\mathbf{x}(q_x, q_y, q_z) = \frac{q_x}{N_x}\mathbf{l}_x + \frac{q_y}{N_y}\mathbf{l}_y + \frac{q_z}{N_z}\mathbf{l}_z, \tag{B.3}$$

where q_x goes from 0 to $N_x - 1$, etc.

Inserting the representation of $f(x)$ into the convolution

$$\tilde{f}(r') = \int \mathrm{d}^3\mathbf{r}\, \omega(r - r') f(r), \tag{B.4}$$

we obtain the relation

$$\hat{\tilde{f}}_{\mathbf{k}} = \hat{\omega}(\mathbf{k})\hat{f}_{\mathbf{k}}, \tag{B.5}$$

where $\hat{\omega}_{\mathbf{k}}$ is the Fourier *transform* of ω

$$\hat{\omega}(\mathbf{k}) = \int \mathrm{d}^3\mathbf{r}\, \omega(r)\, \mathrm{e}^{-\mathrm{i}\mathbf{r}\cdot\mathbf{k}}. \tag{B.6}$$

Analytic expressions for $\hat{\omega}(\mathbf{k})$ are given above. We can use the FFT algorithm to compute $\hat{f}_{\mathbf{k}}$ even for a non-orthogonal unit cell, and the inverse algorithm can be used to convert the product back to the real-space mesh.

The evaluation of the weighted densities can therefore be accomplished with FFT operations using the real space grid representation of the densities together with the analytic expressions for $\hat{\omega}(\mathbf{k})$. There is one more important point to make here. Suppose we have N_w weight functions (10 for the Rosenfeld functional) and there are N_c density components. Then naively we might suppose that the evaluation of the weighted densities would require $N_\mathrm{w}N_\mathrm{c}$ transform operations. In fact they can be constructed with $N_\mathrm{w} + N_\mathrm{c}$ transforms.

This is accomplished by looping over the components. For each density component ρ_i we compute its FFT $\hat{\rho}_i$ and then add $\hat{\omega}^\mathrm{c}_\alpha\hat{\rho}_i$ to $\hat{n}_\alpha$. After the loop over components is complete we apply the inverse FFT to each $\hat{n}_\alpha$ to obtain the weighted densities on the real-space mesh.

As noted above, some care must be taken to perform convolutions with vector valued weight functions.

Appendix C. Interaction and wall potentials

The most commonly used interaction potential between two components i, j is the 12-6 Lennard–Jones (LJ) interaction potential given by

$$U^{\mathrm{lj}}_{ij}(r) = 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right). \tag{C.1}$$

The interaction energy and diameter can be specified (symmetrically) for each pair of components or by means of Lorentz–Berthelot mixing rules which determine the interaction energy and diameter in terms of

parameters specified for each component, thus $\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j)$ and $\epsilon_{ij} = \sqrt{(\epsilon_i \epsilon_j)}$ where ϵ_i and σ_i are the energy and diameter parameters associated with component i.

In order to use this interparticle potential as a mean field perturbation to the hard-sphere fluid, we must somehow cutoff the repulsive part of the potential at small r. In addition, to accurately treat the long-range part of the potential at large r we must cut and shift the potential. We therefore take as the attractive potential the Weeks–Chandler–Anderson [15] split of a cut-and-shifted version of Eq. (C.1). We then have for the potential U_{ij}

$$U_{ij}(r) = \begin{cases} U_{ij}^{\mathrm{lj}}(r_{\min}) - \Delta_{ij} & r \leqslant r_{\min}, \\ U_{ij}^{\mathrm{lj}}(r) - \Delta_{ij} & r_{\min} < r < r_{\mathrm{cut}}, \\ 0 & r \geqslant r_{\mathrm{cut}}, \end{cases} \tag{C.2}$$

where $r_{\min}$ is defined by the minimum of U_{ij}^{lj} and r_{cut} is some suitable cutoff distance. The shift Δ_{ij} is given by $U_{ij}^{\mathrm{lj}}(r_{\mathrm{cut}})$ and should be small. The resulting function is continuous and has a finite range, but there is a small jump in the derivative at r_{cut}. The Fourier transform of this function is required for the computation of the residual and Jacobian operator; this is computed numerically on a fine one-dimensional mesh and then interpolated onto the problem mesh.

A Lennard–Jones potential can also be used to build a realistic representation of some environment that the fluid interacts with, by smearing the potential over a volume of interest. If this volume is taken to be a slab then we get a 9-3 potential

$$U(z) = 8\pi\epsilon\left(\frac{1}{90}\left(\frac{\sigma}{z}\right)^9 - \frac{1}{12}\left(\frac{\sigma}{z}\right)^3\right), \tag{C.3}$$

where z here is the distance to the wall and ϵ, σ are defined as before in terms of parameters for a wall material and a fluid material. The potential inside the wall is infinite and very large just outside the wall; we handle this in the code by clamping the density to zero for $z < z_{\min}$.

This approach can be generalized to define boundary conditions of almost arbitrary complexity. If the potential defined by Eq. (C.1) is cutoff by setting it to a constant for $r < r_{\mathrm{cut}}$ then we can convolve it with a body shape defined by simple geometric primitives (e.g., slabs, cylinders, and spheres) using the FFT convolution method to define the external field for the problem. Note that cutoff is essential, otherwise the convolution can not be defined.

For the calculations performed above with a 9-3 wall (see Fig. 5) we only have a single component. In scaled units, where $kT = 1$ and the hard-sphere diameter is 1 we have $\epsilon = 4.7933$ and $\sigma = .562$. The parameters for the wall are $\epsilon_{\mathrm{wall}} = 19.17$ and $\sigma_{\mathrm{wall}} = .562$, chosen to match the results in [8].

References

[1] L.J.D. Frink, A.G. Salinger, Two- and three-dimensional nonlocal density functional theory for inhomogeneous fluids i. Algorithms and parallelization, J. Comput. Phys. 159 (2) (2000) 407–424.
[2] L.J.D. Frink, 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 (2) (2000) 425–439.
[3] A.G. Salinger, L.J.D. Frink, Rapid analysis of phase behavio with density functional theory I, J. Chem. Phys. 16 (118) (2003) 7457–7465.
[4] L.J.D. Frink, A. Thompson, A.G. Salinger, Applying molecular theory to steady-state diffusing systems, J. Chem. Phys. 112 (17) (2000) 7564–7571.
[5] L.J.D. Frink, A.G. Salinger, M.P. Sears, J.D. Weinhold, A.L. Frischknecht, Numerical challenges in the application of density functional theory to biology and nanotechnology, J. Phys. Cond. Matter 46 (14) (2002) 12167–12187.
[6] D. Henderson (Ed.), Fundamentals of Inhomogeneous Fluids, Marcel Dekker, New York, 1992.
[7] S.A. Hutchinson, L.B. Prevost, R.S. Tuminaro, J.N. Shadid, Aztec User's Guide: Version 2.0. Technical Report, Sandia National Laboratories, 1998.

[8] E. Kierlik, M.L. Rosinberg, Free energy density functional for the inhomogeneous hard-sphere fluid: application to interfacial adsorption, Phys. Rev. A 42 (6) (1990) 3382–3387.
[9] L.J.D. Frink, A.G. Salinger, J. Chem. Phys. 110 (1999) 5969–5977.
[10] P. Rocken, A. Somoza, P. Tarazona, G. Findenegg, J. Chem. Phys. 108 (1998) 8689–8697.
[11] P. Rocken, P. Tarazona, J. Chem. Phys. 105 (1996) 2034–2043.
[12] Y. Rosenfeld, Free energy model for the inhomogeneous hard-sphere fluid mixture and density functional theory of freezing, Phys. Rev. Lett. 63 (9) (1989) 980–983.
[13] Y. Rosenfeld, Free energy model for inhomogeneous fluid mixtures: Yukawa-charged hard spheres, general interactions, and plasmas, J. Chem. Phys. 98 (10) (1993) 8126–8148.
[14] Y. Saad, M.H. Schultz, Gmres: a generalized minimal residual algorithm for solving nonlinear linear systems, SIAM J. Sci. Stat. Comp. 7 (3) (1986) 856–869.
[15] J.D. Weeks, H.C. Anderson, J. Chem. Phys. 54 (1971).