USER GUIDE FOR TRAMONTO v2.1: A Density Functional Theory code for inhomogeneous fluids at equilibrium or steady state

February 9, 2007

Contents

1 Abstract

This users guide is meant to introduce the new user to the input files needed to run Tramonto as well as the output files produced by the code. For help on installation of Tramonto and necessary libraries, see the Installation Guide found on the Tramonto home page: www.software.sandia.gov/tramonto.

2 *dft_globals_const.h*

All shared variable, defined constants, and static array sizes for Tramonto are found in *Tramonto/src/ dft_globals_const.h*. Editing this file to increase static variables can be done. However, it will require corresponding adjustment of the file specific header files. This can be done either manually or using the makeheaders utility (located in *Tramonto/Utilities*. For further instructions on manipulating header files see instructions on the Tramonto Documentation web pages. The current presets for static array variables are summarized here so that input files will be designed within the allowed bounds.

- NCOMP_MAX=5: Maximum number of fluid components.
- NDIM_MAX=3: Maximum number of dimensions.
- NWALL_MAX=600: Maximum number of surfaces (or fixed atoms) in the calculation.
- NWALL_MAX_TYPE=50: Maximum number of different kinds of surfaces (or fixed atoms) in the calculation.
- NBOND_MAX=4: Maximum number of bonds for any given atom.
- NSTEPS_MAX=10: Maximum number of segments in an initial guess built from a stepped density function.
- MAX_ROUGH_BLOCK=100: Maximum number of rough patches on any given surface.
- NZONE_MAX=10: Maximum number of numerical integration stencil zones.
- N_NZCR_MAX=200: Maximum number of non-zeros in a direct correlation function for CMS polymer DFT calculations
- NBLOCK_MAX=5: Maximum number of polymer block types.
- NMER_MAX=100: Maximum number of polymer segments.

3 *dft_input.dat*

The default primary input file for Tramonto is called *dft_input.dat*.[1] It contains most of the parameters that define a given run. However, depending on the problem of interest other input files (described below) that may be needed are:

- *dft_surfaces.dat*
- *poly_file*
- *Cr_file*

In *dft_input.dat*, all lines containing data begin with an @ symbol signaling the code to begin reading data. Arbitrary numbers of blank lines or comments may be inserted following the data since everything after the needed data is treated as a comment. However, note that in most cases TRAMONTO needs to read in arrays in their entirety. Numerous comments are included in the *dft_input.dat* file. These comments are meant to assist the user in setting up the input file.

The input data is split into different logical categories described below. After the *dft_input.dat* file is read in, it is echoed in the file *dft_out.lis*. When the code isn't behaving properly, first check that the code has read the input file as expected.[2] Some degree of error checking for conflicting input is built into the code; however, a complete cross check of parameters is not guaranteed. Therefore, the user is encouraged to double check input files before submitting jobs.

[1] Note that the user can give this file any name, but the job must then be launched by typing: dft input_filename.
[2] Note that the user does not need to make any modifications to the input file to run in parallel.

3.1 DIMENSION PARAMETERS

This section defines what kind of dimensions will be set in the input file as well as a maximum allowable external field. All calculations in Tramonto are performed in reduced units. All input parameters are reduced by the reference values entered here as summarized in Table 1.

In the first example shown below, all variables are to be entered in reduced units as indicated by setting parameters < 0. In the second example, the reference length is 3 Angstroms, the reference density is 18.456995 (units of $(mol/L)/\sigma^{-3}$), the temperature is 298 K, and the reference dielectric constant is 80.

==========

Prototype

Length_ref(real) Density_ref(real) Temp(real) Dielec_ref(real) VEXT_MAX(real)

==========

Example 1

```
*************** DIMENSION PARAMETERS ***********************
@  -1. -1. -1. -1. 10.     Length_ref Density_ref Temp Dielec_ref VEXT_MAX
************************************************************
```

Example 2

```
*************** DIMENSION PARAMETERS ***********************
@  3.0 18.456995 298. 80. 10.     Length_ref Density_ref Temp Dielec_ref VEXT_MAX
************************************************************
```

==========

PARAMETER DEFINITION

Length_ref : The characteristic length in the problem of interest. Typical options are:

- reduced units to be entered ($Length\_ref < 0.0$).
- size of one atomic fluid species in the problem.

Density_ref: The reference density entered here should be the ratio of #[units of interest]/1[σ^{-3} units] where options are:

- reduced units entered for all density parameters ($Density\_ref < 0.0$).
- real units of g/cc, Molar, etc.

Temp: Temperature options:

- reduced units for energy parameters ($Temp < 0.0$).
- Temperature in Kelvin units.

Dielec_ref: The reference dielectric constant. Options are:

- use standard assumption of 78.5 for the dielectric constant ($Dielec\_ref < 0.0$).
- real dielectric constant ($Dielec\_ref > 0.0$).

VEXT_MAX: The maximum external field to be allowed in the problem. If $V^{ext} > VEXT\_MAX$ at some nodes in the mesh then the residual equations are replaced with $\rho = 0$ at those nodes. This parameter should be entered relative to the temperature as V/kT. This parameter improves numerical stability of the code by eliminating the need to converge very small densities (note that in an ideal gas $\rho(\mathbf{r})/\rho_b = e^{-V/kT}$).

Ref parameter	Normalization	Affected parameters
Length_ref	x/*Length_ref*	(*Size_x, Esize_x, WallPos, WallParam, Sigma_ff, Cut_ff, Dielec_x, X_1D_bc, X_const_mu, Pore_rad_R_IC, Lseg_IC*, and *Thickness*
Length_ref	$\sigma(Length\_ref)^2$	*Elec_param_w* if $Type\_bc\_elec = 2$ (charge per area)
Density_ref	$\rho/Density\_ref$	*Rho_w, Rho_b, Rho_b_LBB, Rho_b_RTF, Scale_fac*
Temp	$(\epsilon/k)/Temp$	*Eps_ff, Eps_ww, Eps_wf, Vext_membrane*
Temp	$\phi e/k(Temp)$ where ϕ is entered in mV units	*Elec_param_w, Elec_pot_LBB, Elec_pot_RTF*

Table 1: Parameters that are adjusted by the various reference parameters as well as details of how various parameters are reduced.

3.2 MESH PARAMETERS

These parameters are used to set up the variables controlling the size dimension and mesh spacing of the computational domains. The example below shows a 3-Dimensional (3D) problem where the computational domain is $4\sigma \times 4\sigma \times 1\sigma$ in size and the mesh spacing is 0.2σ in each dimension. The system is a square channel with wall-boundary conditions on both sides in both the x and y coordinate and periodic boundary conditions in the z coordinate.

==========

Prototype

Ndim (int)
Size_x[idim] (real array)
Esize_x[idim] (real array)
Type_bc[idim=0][iside=0,1] (int array)
Type_bc[idim=1][iside=0,1] (int array)
Type_bc[idim=2][iside=0,1] (int array)

==========

Example

```
*************** MESH PARAMETERS ************************
@  3      Ndim
@  4.0 4.0 1.0    Size_x(idim); idim=0,Ndim-1
@  0.2 0.2 0.2  Esize_x(idim): idim=0,Ndim-1
@   -1   -1    Type_bc(x0: left,right) (-1=wall,0=bulk,1=pbc,2=ref; 3=cont)
@   -1   -1    Type_bc(x1: down,up) (-1=wall,0=bulk,1=pbc,2=ref; 3=cont)
@    1    1    Type_bc(x2: back,front) (-1=wall,0=bulk,1=pbc,2=ref; 3=cont)

*******************************************************
```

==========

PARAMETER DEFINITION

Ndim: The number of spatial dimensions in the problem of interest.

Size_x[idim]: An array that stores the size of the computational domain in each dimension.

Esize_x[idim]: An array that stores the mesh spacing of the computational domain in each dimension.[3,4]

Type_bc: An array that stores the types of boundary conditions in each dimension. Computation of residual and sometimes Jacobian entries requires computation of integrals that extend beyond the computational domain. Assuming that the mesh points in any given dimension run from 0 to N, and k is an integration point beyond the domain boundary, options for the boundary conditions are:

- -1: IN_WALL: semi-infinite surface : set $\rho = 0$ when integrating beyond boundary.
- 0: IN_BULK: constant bulk fluid: set $\rho_{N+k} = \rho_b$.
- 1: PERIODIC: periodic boundary: set $\rho_{N+k} = \rho_k$.
- 2: REFLECT: reflective boundary: set $\rho_{N+k} = \rho_{N-k}$.
- 3: LAST_NODE: continuation boundary: set $\rho_{N+k} = \rho_N$.

[3]Note: Esize_x need not be the same in all dimensions. However, it should be an integer divisor of 1σ. Thus, possible values are 0.5, 0.3333, 0.25, 0.2,0.16666, 0.125, 0.1,0.05 etc.

[4]Note: Practical values for the Esize_x range are 0.05σ (1D and 2D atomic fluids DFT), $0.1\sigma - 0.25\sigma$ (2D-3D atomic fluids DFT and polymer-DFT where molecular structure is of interest), or $0.25\sigma - 0.5\sigma$ (for debugging and for 2D & 3D polymer-DFT calculations where mesoscopic structures are of interest).

3.3 FUNCTIONAL SWITCH PARAMETERS

These are switches that control the density functional equations being studied for a given case. The functional switches are separated into hard sphere, attractive, Coulombic, and polymer functionals. The switches that are currently available in the code are detailed below. For details on the notation and functionals, read the cited references. The example shows a case where the updated Rosenfeld functional is combined with second order short range corrections to an electrolyte fluid.

==========

Prototype

Type_func (int)
Type_attr(int) *Type_pairPot(int)*
Type_coul(int)
Type_poly(int)

==========

Example

```
************ FUNCTIONAL SWITCHES *******************************
@ 1             Type_func (-1=No HS functional, 0=FMT1, 1=FMT2, 2=FMT3)
@ -1 0         Type_attr    Type_pairPot
               (Type_attr options: -1=No attractions, 0=strict MF, 1=Barker-Henderson MF)
              (Type_pairPot options: 0=PAIR_LJ12_6_CS, 1=PAIR_COULOMB)
@ 1             Type_coul (-1=No coulomb, 0=strict MF, 1=include 2nd order corrections)
@ -1            Type_poly (-1=No polymer, 0=CMS, 1=CMS_SCFT, 2=WTC)

*****************************************************************
```

==========

PARAMETER DEFINITION

Type_func: The type of hard sphere functional desired for the calculation. Note that these are used in perturbation calculations where the reference system is a hard sphere fluid. Options are:

- -1: NONE: No hard sphere functional ($\Phi = 0$). Do this for ideal gas, Poisson-Boltzmann, or CMS polymer calculations.
- 0: FMT1: original FMT functional developed by Rosenfeld [1].
- 1: FMT2: an updated FMT functional with corrected zero dimensional cross-over behavior [2, 3].
- 2: FMT3: the White Bear FMT functional [4].

Free energy densities of each of these hard-sphere functionals are summarized in Table 2. Note that for a complete description (including nonlocal densities, n and $\mathbf{n_V}$, as well as an analysis of the functionals), the original references should be studied.

Label	Energy Density
FMT1	$\Phi = -n_0 \ln(1-n_3) + \frac{n_1 n_2 - \mathbf{n}_{V1}\cdot\mathbf{n}_{V2}}{1-n_3} + \frac{1}{24\pi}\frac{n_2^3}{(1-n_3)^2} - \frac{1}{8\pi}\frac{n_2(\mathbf{n}_{V2}\cdot\mathbf{n}_{V2})}{(1-n_3)^2}.$
FMT2	$\Phi = -n_0 \ln(1-n_3) + \frac{n_1 n_2 - \mathbf{n}_{V1}\cdot\mathbf{n}_{V2}}{1-n_3} + \frac{\left(n_2^2 - \mathbf{n}_{V2}\cdot\mathbf{n}_{V2}\right)^3}{24\pi n_2^3 (1-n_3)^2}.$
FMT3	$\Phi = -n_0 \ln(1-n_3) + \frac{n_1 n_2 - \mathbf{n}_{V1}\cdot\mathbf{n}_{V2}}{1-n_3} + \left(n_2^3 - 3n_2\mathbf{n}_{V2}\cdot\mathbf{n}_{V2}\right)\frac{n_3 + (1-n_3)^2\ln(1-n_3)}{36\pi n_3^2(1-n_3)^2}.$

Table 2: Description of FMT HS functionals

Note that it is relatively straightforward to implement new FMT based hard sphere functionals. For directions, see the developer pages for Tramonto at http://software.sandia.gov/tramonto/developer_physics.html.

Type_attr: The type of perturbation functional to use for potential interactions longer range than hard spheres as defined in Table 3. Note that the *attr* reflects the origins of this perturbation approach in attractive potentials of relatively short range (e.g. Lennard-Jones fluids), but the approach may be applied to other types of pair potentials as well (see description of *Type_pairPot* below). Options for *Type_attr* are:

- -1: NONE: No extended potentials to consider
- 0: MFPAIR1: A simple strict Mean Field treatment of perturbation potential. Use this option for CMS polymer calculations.
- 1: MFPAIR2: A second strict Mean Field treatment of perturbation potential using a Barker-Henderson adjustment to the hard core reference fluid diameters.

LABEL	Excess Free Energy / or other use in functional	HS diameter
MFPAIR1	$F_{att} = \sum_i \sum_j \int d\mathbf{r} \int d\mathbf{r}' \ \rho_i(\mathbf{r}) \rho_j(\mathbf{r}') u_{ij}^{att}(\lvert r - r' \rvert)$	$d_{ij} = \sigma_{ij}$
MFPAIR1 ($Type\_poly = CMS$)	$c(r) = c_{ref} + u_{att}$ where c_{ref} is a repulsive contribution that may be obtained from PRISM theory or molecular simulation.	N/A
MFPAIR2	$F_{att} = \sum_i \sum_j \int d\mathbf{r} \int d\mathbf{r}' \ \rho_i(\mathbf{r}) \rho_j(\mathbf{r}') u_{ij}^{att}(\lvert r - r' \rvert)$	$d_{ij} = \int_0^{\sigma_{ij}} (1 - e^{u_{ij}(r)}) dr$

Table 3: Description of options for including extended potentials in DFT using either a hard sphere reference fluid or a hard chain reference fluid ($Type\_poly = CMS$). Note that u^{att} is the attractive part of the pair potential of interest. At short range u^{att} is set to some simple constant with the understanding that the hard sphere reference fluid captures the repulsive part of the interaction potential. In other words, u_{ij} is replaced with $u_{ij}^{att} + u_{ij}^{hs}$. A cut and shift of the potential is always required because integration stencils in the calculations must be of finite range.

Type_pairPot: The pair potential that will be used in conjunction with the mean field functionals defined by *Type_attr*. Table 4 defines the interaction potential as well as u^{attr} for each case. Options are:

- 0: PAIR_LJ12_6_CS: A cut and shifted 12-6 Lennard-Jones fluid where cutoff distances are typically set to $r_{cut} = 2.5\sigma - 10\sigma$
- 1: PAIR_COULOMB: A cut and shifted Coulomb potential. Note that while Coulomb fluids may be treated in this manner, this approach requires a cut and shift of the interaction potential. The better approach for charged systems replaces these perturbations with the electrostatic potential and couples the solve of the DFT to a solve of Poisson's equation. To perform that type of calculation, turn off $Type\_attr$ and turn on $Type\_coul$.

Note that it is relatively straightforward to add new pair potentials for use with the attractive mean-field functionals. For directions, see the developer pages for Tramonto at http://software.sandia.gov/tramonto/developer_physics.html.

Label	Pair Potential: $u_{ij}(r)$	u^{att}
PAIR_LJ12_6_CS	$u_{ij}^{LJ}(r) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right]$	$u^{att}(r) = u_{ij}(r_{min})$ for $r < r_{min}$ where r_{min} is the minimum well depth of u_{ij} This is the WCA approach (WCA=Weeks-Chandler-Anderson).
PAIR_COULOMB	$u(r) = q_i q_j/(T_{elec} r)$ $q_i = z_i e$ where z_i is the valence of species i $T_{elec} = 4\pi k T \kappa \epsilon_0 \sigma / e^2$ κ =dimensionless dielectric constant $\epsilon_0 = 8.85419e-12C^2J^{-1}m^{-1}$ $e = 1.60219e^{-19}C$ $k = 1.3807e^{-23}J/K$ σ =diameter of reference fluid atom	$u^{att}(r) = u_{ij}$ for $r < d_{ij}$ where d_{ij} is the diameter of the hard sphere reference fluid

Table 4: Description of pair potentials

Type_coul: The type of functional used when the DFT Euler-Lagrange equations are coupled to a Poisson's equation solve for the electrostatics. The contribution of the electrostatics to the DFT free energy in the perturbation treatment is detailed in Table 5. Options are:

- -1: NONE: turn off Poisson Terms. Do this for neutral systems or for cases where $Type\_pairPot = 1$.
- 0: BARE: Mean field electrostatics based on point charges only.
- 1: DELTAC: Mean field electrostatics for point charges plus 2nd order correction of Tang and Davis. This only applies to a restricted primitive model (RPM) where all charged species have identical sizes, σ_{ij}. It is based on an analytical solution for the RPM using the mean spherical approximation (MSA).
- 2: POLARIZE: Electrostatics for a polarizeable fluid. Implemented only in 1D as of the February 2007 release of Tramonto v2.1.

Label	Contribution to Euler-Lagrangeeqn (or other residual eqn)	Electrostatics
BARE	$q_i\psi(\mathbf{r})$	$\nabla^2\psi(\mathbf{r}) = -\sum_i z_i e \rho_i$
BARE ($Type\_poly = CMS$)	$q_i\psi(\mathbf{r})$ (added in R_1 residual equation)	$\nabla^2\psi(\mathbf{r}) = -\sum_i z_i e \rho_i$
DELTAC	$q_i\rho_i(\mathbf{r})\psi(\mathbf{r}) + \sum_j \int d\mathbf{r}' \rho_i(\mathbf{r}') \Delta c_{ij}^{MSA}(\lvert r - r' \rvert)$	$\nabla^2\psi(\mathbf{r}) = -\sum_i z_i e \rho_i$
POLARIZE	$q_i\psi(\mathbf{r})$	$\nabla^2\psi(\mathbf{r}) = -\sum_i z_i e \rho_i$

Table 5: Description of options for electrostatics

Type_poly: The type of polymer functional to be used if the fluid is composed of bonded systems. Table 6 describes the unknown fields solved for these cases as well as the residual equations (or contributions to the free energy) associated with these functionals. Table 7 describes the various unknown fields, the number of unknowns per node associated with each type of field, and the label used in Tramonto to describe these unknowns. Extended potentials and charges can also be added to these polymer functionals by turning on $Type\_attr$ and/or $Type\_coul$ parameters. For the CMS functionals, one must set $Type\_attr = 0$, even for purely repulsive chains, so that the code calculates the mean-field part of the potential correctly. Options for $Type\_poly$ are:

- -1: NONE: No polymer functionals. No bonds.

- 0: CMS: Chandler-McCoy-Singer DFT [5, 6, 7] based on freely-jointed chains where the single chain part of the functional is evaluated numerically as described by Donley et al. [8]. Also see our previous work [9]. The notation here has been generalized for branched chains. If using the CMS functionals, one must set $Type\_attr = 0$, even for purely repulsive chains, so that the code calculates the mean-field part of the potential correctly. Finally, note that this functional is _NOT_ based on a perturbation to hard spheres, but rather on a second order perturbation to ideal chains.
- 1: CMS_SCFT: This option simplifies the CMS theory to reproduce polymer Self-Consistent Field Theory. Note that the algorithms in Tramonto are not optimal for SCFT. Rather this approach serves as a test and a point of comparison with work in the SCFT community. This option is not fully implemented as of the February 2007 release of Tramonto v2.1.
- 2: WTC: Tripathi-Chapman functionals [10, 11] based on a Wertheim's theory approach in the limit of infinitely strong associations. This approach is a perturbation theory based on a hard sphere reference fluid. As such it requires one of the FMT types for $Type\_func$ to be turned on.

Label	Unknown fields	Residual equation
CMS	$x_1^\alpha(\mathbf{r}) = e^{-U_\alpha(\mathbf{r})/kT}$	$R_1 = 0 = \ln(x_1^\alpha(\mathbf{r})) + V^\alpha(\mathbf{r})/kT - \sum_\beta \int_V d\mathbf{r}' c_{\alpha\beta}(\lvert\mathbf{r}-\mathbf{r}'\rvert)(x_2^\beta(\mathbf{r}') - \rho_{bulk}^\beta)$
	$x_2^\alpha(\mathbf{r}) = \rho_\alpha(\mathbf{r})$	$R_2 = 0 = x_2^\alpha(\mathbf{r}) * x_1^\alpha(\mathbf{r}) - \frac{\rho^\alpha}{N_\alpha}\sum_i^{\{N_\alpha\}}\left(\prod_{\beta=1}^{N_i} x_3^{i,j(\beta)}(\mathbf{r})\right)(x_1^\alpha(\mathbf{r}))^{-(N_i-2)}$
	$x_3^{i,j}(\mathbf{r}) = G_i^j(\mathbf{r})$	$R_3^{i,j}(\mathbf{r}) = 0 = x_3^{i,j}(\mathbf{r}) - x_1^{\alpha(i)}(\mathbf{r})\times$ $\int d\mathbf{r}'\left(\prod_{\beta=1}^{N_j:k(\beta)\neq i} x_3^{j,k(\beta)}(\mathbf{r}')\right)\left(x_1^{\alpha(j)}(\mathbf{r}')\right)^{-(N_j-2)} w_{i,j}(\mathbf{r}-\mathbf{r}')$
WTC	$x_1^{iseg}(\mathbf{r}) = \rho_{iseg}(\mathbf{r})$	$\Delta A^{assoc}/kT = \frac{1}{2}\int d\mathbf{r}\sum_\alpha \rho_\alpha(\mathbf{r})\sum_{\alpha'\{A\}_\alpha}(1-\ln\{y_{\alpha\alpha'}(\{\xi_i(\mathbf{r}\})n_{\alpha\alpha'}(\mathbf{r})\})$ where $y_{\alpha\alpha'}(\{\xi_i\}) = \frac{1}{1-\xi_3} + \frac{3\sigma_\alpha\sigma_{\alpha'}}{\sigma_\alpha+\sigma_{\alpha'}}\frac{\xi_2}{(1-\xi_3)^2} + 2\left(\frac{\sigma_\alpha\sigma'_\alpha}{\sigma_\alpha+\sigma'_\alpha}\right)^2\frac{\xi_2^2}{(1-\xi_3)^3}$ (chain contribution to perturbation DFT free energy functional)
	$x_2(\mathbf{r}) = \xi_{[2,3]}$	
	$x_3(\mathbf{r}) = n_{\alpha\alpha'}(\mathbf{r})$	$n_{\alpha\alpha'}(\mathbf{r}) = \int d\mathbf{r}'\omega^{\alpha\alpha'}(\lvert\mathbf{r}-\mathbf{r}'\rvert)\rho_{\alpha'}(\mathbf{r}')$

Table 6: Description of systems of equations for polymer functionals. Note that in the CMS functional $\omega_{ij}(\mathbf{r}-\mathbf{r}') = \delta(\lvert\mathbf{r}-\mathbf{r}'\rvert - b_{ij})/(4\pi b_{ij}^2)$ and in the WTC functional $\omega^{\alpha\alpha'}(\mathbf{r}-\mathbf{r}') = \delta(\lvert\mathbf{r}-\mathbf{r}'\rvert - b_{\alpha\alpha'})/(4\pi b_{\alpha\alpha'}^2)$ where b is a bond length, and the ij (or $\alpha\alpha'$) pairs refer to bonded segments on the chains. The δ function sets the chain type to freely-jointed chains. Liquid state information enters the CMS functional through $c_{\alpha\beta}(\mathbf{r})$ which is the direct correlation function between sites of type α and type β.

Func Label	Unknown fields	description	# unks/node	Unknown Label
CMS	$x_1^\alpha(\mathbf{r}) = e^{-U_\alpha(\mathbf{r})/kT}$	U_α =mean fields (α =component)	N_{block}	CMS_FIELD
CMS	$x_2^\alpha(\mathbf{r}) = \rho_\alpha(\mathbf{r})$	component densities	N_{block}	DENSITY
CMS	$x_3^{i,j}(\mathbf{r}) = G_i^j(\mathbf{r})$	propagator funcs for all j bonded to i	$2N_{bond}$	CMS_G
WTC	$x_1^{iseg}(\mathbf{r}) = \rho_{iseg}(\mathbf{r})$	segment densities	N_{seg}	DENSITY
WTC	$x_2(\mathbf{r}) = \xi_{[2,3]}$	cavity equations	2	CAVWTC
WTC	$x_3(\mathbf{r}) = n_{\alpha\alpha'}(\mathbf{r})$	bond densities	$2N_{bond}$	BONDWTC

Table 7: Further description of unknown fields solved in Tramonto.

Note that since the WTC functional does not strictly enforce chain bonding, the functional can exhibit incorrect stoichiometry in an inhomogeneous system. The CMS functionals exhibit smaller stoichiometry errors at hard walls.

3.4 SURFACE PARAMETERS

These parameters are used to set up the geometric parameters defining the surfaces of interest. The example shows the case of a square channel where the channel is created from two types of planar surfaces. The two types have different orientations.

==========

Prototype

Nwall_type Nwall Nlink Lauto_center Lauto_size (int,int,int,int,int)
Xtest_reflect_TF[ilink][idim](int array)
Surf_type[iwall_type](int array)
Orientation[iwall_type](int array)
WallParam[iwall_type](real array)
WallParam2[iwall_type](real array)
WallParam3[iwall_type](real array)
WallParam4[iwall_type](real array)
Lrough_surf[iwall_type](int array)
rough_param_max[iwall_type](real array)
Rough_length[iwall_type](real array)

==========

Example

```
********** SURFACE PARAMETERS **********************
@  2 4 3 0 0             Nwall_type Nwall Nlink Lauto_center Lauto_size
@ 0 0 0 0 0 0 0 0 0 0     Xtest_reflect_TF
@  0  0             Surf_type[iwall_type]; iwall_type=0,Nwall_type-1
@  0  1             Orientation[iwall_type]; iwall_type=0,Nwall_type-1
@  1.0 1.0          WallParam[iwall_type]; iwall_type=0,Nwall_type-1
@  1.0 1.0          WallParam2[iwall_type]; iwall_type=0,Nwall_type-1
@  1.0 1.0          WallParam3[iwall_type]; iwall_type=0,Nwall_type-1
@  0.0 0.0          WallParam4[iwall_type]: iwall_type=0,Nwall_type-1
@  0.0 0.0          Lrough_surf[iwall_type]: iwall_type=0,Nwall_type-1
@  0.0 0.0          rough_param_max[iwall_type]: iwall_type=0,Nwall_type-1
@  0.0 0.0          Rough_length[iwall_type]: iwall_type=0,Nwall_type-1


************************************************************
```

==========

PARAMETER DEFINITION

ilink: Index over the number of independent compound (or linked) surfaces in the system, *Nlink*. *ilink*={*0,Nlink-1*}

iwall_type: Index over the different *types* of surfaces in the system. *iwall_type*={*0,Nwall_type-1*}.

Nwall_type: The number of different surface *types* in the system. Two walls are of the same type if **all** of the parameters defining them are the same.

Nwall: The total number of surfaces in the system.

Nlink: The number of independent compound surfaces in the system.

Lauto_center: Have the code automatically center the surfaces in the computational box. This may facilitate applying bulk boundaries or viewing the results of the simulation when the initial coordinates are not well centered. Taking coordinates from the Protein Data Bank is one case where the surface (atomic) coordinates may not be well centered.

Lauto_size: Automatically determine a box size for coarsened meshes.

Xtest_reflect_TF: A logical array (0=FALSE, 1=TRUE) that indicates how to treat surfaces that exist in adjoining domains across reflective boundaries. TRUE indicates that the surfaces in the next domain are different than those in the computational domain, while FALSE indicates that they are extensions of the surfaces in this domain. For example, consider a cylindrical surface with the long axis parallel to the z-coordinate of the system and assume all boundaries of the 3D domain are reflective. In the x and y directions, there are independent surfaces beyond the boundaries while in the z direction, the same cylinder is extended. These situations must be treated differently when the surface-fluid interactions are hard.

Surf_type: This array stores the basic shapes of the surfaces. Several choices are currently possible as outlined in Table 8, and addition of new surfaces in the code is straightforward. The difference between colloidal cylinder/spheres and atomic cylinder/spheres is that in the former case the WallParam is used to define the radii of the surfaces while in the latter case, the wall σ_w (*Sigma_w*) parameter entered later defines the surface diameter.

Orientation: This parameter identifies the orientation of the surface as detailed in Table 8. This parameter is always a direction (*Orientation={0,2}*). For example it gives the direction of the surface normal in the case of an infinite planar surface. If no orientation is needed, any number may be present in the input file.

WallParam, WallParam2, WallParam3: Arrays that store the parameters needed to describe the geometry of a given surface type. The parameters are given in Table 8. Each surface type must have an entry for each type of WallParam although some wall types only need one parameter. In these cases any number may appear in the list since these numbers are never used. It is important to note that TRAMONTO needs to read in the arrays in their entirety so entries may not be omitted from the input file.

WallParam4, Lrough_surf, rough_param_max, Rough_length: These parameters are used for a special case of surface, a cylindrical wedge defined by a polar angle (either infinite with Ndim=2 or finite length with Ndim=3). For this case, the origin is placed at the center of the cylinder. The surface consists of a wedge beginning at an angle, in degrees, specified by WallParam3 and ending at the angle given by WallParam4, with 0 degrees corresponding to the x-axis. To obtain a full cylinder, set WallParam3 = 0 and WallParam4 = 360. These cylindrical wedges can also be given a rough surface as described next.

Lrough_surf: a logical flag specifying whether the surface should be rough or not: 0 = not rough, 1 = rough. Only implemented for cylinders at this time.

rough_param_max: This parameter specifies the magnitude of the roughness, in units of σ_{ref}. The code then makes the radius of the cylinder = radius $\pm\tau$, where τ is a random number with a maximum magnitude of rough_param_max.

Rough_length: The length scale for the roughness along the long axis of the cylinder, in units of σ_{ref} (i.e. how many different τ values should occur along the cylinder).

Surf_type	flag	Ndim	WallParam	WallParam2	WallParam3	Orientation
Infinite Planar Wall	0	1-3	half thickness	N/A	N/A	Surf Normal
Finite Planar Wall	1	1-3	half thickness	half thickness	half thickness	Surf Normal
Colloids (cyl/sphere)	2	2-3	radius	N/A	0.0	N/A
Atoms (spheres)	3	3	radius	N/A	N/A	N/A
Point Atoms	4	3	radius	N/A	N/A	N/A
Cylinder - finite length	5	3	radius	length	0.0	Long Axis
Cyl/Periodic	6	3	mean radius	amplitude	period length	Long Axis
Pore (cyl-2D/sphere-3D)	7	2-3	radius	N/A	N/A	N/A
Finite Pore (slit-2D/cyl-3D)	8	2-3	pore radius	length	N/A	Long Axis
Tapered Pore	9	2-3	pore radius (lbb)	pore radius (rtf)	length	Long Axis

Table 8: Details of possible surface types. The columns are a description of the surface type, the flag entered in *dft_input.dat* to select a given surface, the number of dimensions possible for a given surface type, the meanings of various wall parameters, and the definition of the orientation parameter. The notation lbb refers to the left(idim=0), bottom (idim=1) or back (idim=2) radius of the tapered pore while rtf refers to the right, top, or front dimension.

3.5 INTERACTION POTENTIAL TYPE PARAMETERS

These switches are used to indicate the type of wall-fluid and wall-wall interaction potential models that are to be used in the calculation. Note that the wall-fluid interactions may be different for different wall types. Here we list the options currently implemented in the code. New interactions may be added by writing an appropriate set of functions; see the Tramonto website for more information. Finally, note that any surface can be treated as a (semi-)permeable object as described in section 2.8.

==========

Prototype

Ipot_wf_n[iwall_type] (int array)
Lhard_surf (int)
Type_vext1D Type_vext3D (int, int)
Ipot_ww_n[iwall_type][jwall_type] (int array)
Type_uwwPot (int)

==========

Example

```
************** INTERACTION POTENTIAL TYPE SELECTIONS ************************
@ 1 2     Ipot_wf_n[iwall_type]
                   0=No_wall-fluid,
                   1=Hard_wall,
                   2=1D potential in 1D calculation,
                   3=1D potential in 2D or 3D calculation - based on Xmin[iwall]
                   4=1D potential in 2D or 3D calculation - based on Xmin[Orientation]
                   5=LJ12_6 integrated,
                   6=ATOMIC potential (LJ 12-6)

@ 1       Lhard_surf (Logical that controls application of integration stencil at the
                   boundaries. If TRUE (1), the step function at the boundary is treated
                   carefully).

@ 0 0     Type_vext1D,  Type_vext3D
                  (Type_vext1D options: 0=LJ9_3_CS,  1=LJ9_3_v2_CS, 2=LJ9_3_noCS,
                      3=LJ9_3_shiftX_CS,4=REPULSIVE9_noCS, 5=EXP_ATT_noCS)
                   (Type_vext3D options: 0=PAIR_LJ12_6_CS, 1=PAIR_COULOMB)

@ -2        Ipot_ww_n[iwall_type][jwall_type]
             -2 : set entire array to 0
             -1 : set entire array to 1
              0 : no interactions
               1: compute interactions of atom centers (LJ+COULOMB)

@ 0          Type_uwwPot
               (Type_uwwPot options: 0=PAIR_LJ12_6_CS, 1=PAIR_COULOMB)
*****************************************************************
```

==========

PARAMETER DEFINITION

Ipot_wf_n: An array of switches indicating the type of neutral interactions between the fluid particles and the surfaces. Table 9 provides a description of several of the options. The options are:

- 0: VEXT_NONE: No external field.
- 1: VEXT_HARD: An infinitely hard wall.

- 2: VEXT_1D: A 1-dimensional (1D) potential used in a 1D calculation. The specific functional form is set by *Type_vext1D* defined below.
- 3: VEXT_1D_XMIN: A 1D potential to be used in a 2D or 3D calculation. The algorithm finds the minimum distance from each surface and uses those distances to compute the external field.
- 4: VEXT_1D_ORIENTATION: A 1D potential used in a 2D or 3D calculation. The algorithm finds the distance to the nearest surfaces in the Orientation direction. Those distances, x are used to compute the external field.
- 5: VEXT_3D_INTEGRATED: Start with a 3D potential (defined by *Type_vext3D* below), and numerically integrate that potential over the surface of interest from every point in the fluid domain. This allows an arbitrary surface geometry to be composed of say Lennard-Jones atoms at a certain density without representing them explicitly. Given a certain class of potentials (e.g. Lennard-Jones 12-6 in 3D and the corresponding analytical surface integration to a 9-3 potential), identical results should be obtained from this option when $r_{cut}^{w} f \to \infty$ as is obtained from VEXT_1D $z_{cut} \to \infty$.
- 6: VEXT_ATOMIC: Surfaces are atoms, and a 3D interaction potential (defined by *Type_vext3D*) is used to calculate the external field.

Label	Mathematical Description of External Field
VEXT_NONE	$V^{ext}(\mathbf{r}) = 0$ for all $\mathbf{r}$.
VEXT_HARD	$V^{ext}(\mathbf{r}) = \infty$ if $\lvert\mathbf{r} - \mathbf{R}_s\rvert < WallParam + \sigma_{wi}/2$ $V^{ext}(\mathbf{r}) = 0$, otherwise where $\mathbf{R_s}$ is the location of the surface
VEXT_1D	$V_{1D}(x)$
VEXT_1D_XMIN	$V_{1D}(x_{min}[idim])$
VEXT_1D_ORIENTATION	$V_{1D}(x_{min}[Orientation[type\_wall])$
VEXT_3D_INTEGRATED	$V(\mathbf{r}) = \int d\mathbf{r_s} \left[u_{3D}(\lvert\mathbf{r} - \mathbf{r_s}\rvert) - u_{3D}(r_{cut})\right]$ where $d\mathbf{r_s}$ indicates an integral over the elements in the surface.
VEXT_ATOMIC	$V(\mathbf{r}) = \sum_{Nwall} u_{3D}(\lvert\mathbf{r} - \mathbf{r_s}\rvert) - u_{3D}(r_{cut})$ where $\mathbf{r_s}$ is the position of a fixed surface atom.

Table 9: Different options for computing external fields in Tramonto.

Lhard_surf: A logical that controls how hard boundaries are handled. If TRUE (1), then the step function at the boundary of a hard wall is treated carefully to account for the discontinuity. Otherwise, all fields are assumed to vary linearly in elements, even at the boundary of a hard surface.

Type_vext1D: The type of 1D external potential to use for computing the external field. The potentials are a function of the distance x between the surface of interest and a point in the fluid. Table 10 provides specifics. The options are:

- 0: LJ9_3_CS: The cut and shifted Lennard-Jones 9-3 wall for 1D systems of infinite planar walls.
- 1: LJ9_3_v2_CS: Same functional form as LJ9_3_CS, but with different prefactors
- 2: LJ9_3_noCS: The 9-3 Lennard-Jones potential without the cut and shift.
- 3: LJ9_3_shiftX_CS: This option shifts the usual 9-3 LJ wall potential by a factor Δ, given by $\Delta_{ff} = (\sigma_{ff} - 1)/2$.
- 4: REPULSIVE9_noCS: This is a purely repulsive wall. No cut and shift.
- 5: EXP_ATT_noCS: An exponential external field. No cut and shift.

Label	$V^{ext}(x)$	1D external fields - functional form
LJ9_3_CS	$V(x) - V(x_{cut}) \quad x < x_{cut}$ $0 \quad x > x_{cut}$	$V(x) = \frac{2\pi\epsilon_{wf}\rho_w\sigma_{wf}^3}{3}\left[\frac{2}{15}\left(\frac{\sigma_{wf}}{x}\right)^9 - \left(\frac{\sigma_{wf}}{x}\right)^3\right]$
LJ9_3_v2_CS	$V(x) - V(x_{cut}) \quad x < x_{cut}$ $0 \quad x > x_{cut}$	$V(x) = \epsilon_{wf}\rho_w\left[\frac{2}{15}\left(\frac{\sigma_{wf}}{x}\right)^9 - \left(\frac{\sigma_{wf}}{x}\right)^3\right]$
LJ9_3_noCS	$V(x)$ $0 \quad x > x_{cut}$	$V(x) = \frac{2\pi\epsilon_{wf}\rho_w\sigma_{wf}^3}{3}\left[\frac{2}{15}\left(\frac{\sigma_{wf}}{x}\right)^9 - \left(\frac{\sigma_{wf}}{x}\right)^3\right]$
LJ9_3_shiftX_CS	$V(x) - V(x_{cut}) \quad x < x_{cut} + \Delta_{ff}$ $0 \quad x > x_{cut} + \Delta_{ff}$	$V^{LJ}(x) = \epsilon_{wf}\rho_w\left[\frac{2}{15}\left(\frac{\sigma_{wf}}{x-\Delta_{ff}}\right)^9 - \left(\frac{\sigma_{wf}}{x-\Delta_{ff}}\right)^3\right]$
REPULSIVE9_noCS	$V(x)$ $0 \quad x > x_{cut}$	$V^{ext}(x) = \frac{2\pi\epsilon_{wf}\rho_w\sigma_{wf}^3}{3}\left[\frac{2}{15}\left(\frac{\sigma_{wf}}{x}\right)^9\right]$
EXP_ATT_noCS	$V(x)$ $0 \quad x > x_{cut}$	$V(x) = -\epsilon_{wf}\rho_w e^{-x/\sigma_{wf}}$

Table 10: Description of 1D external field options. Note that all cutoff distances are set by the input parameter Cut_wf as $x_{cut} =$ *Cut_wf[icomp][iwall_type]*. See section on Interaction Parameters for more details.

See http://software.sandia.gov/tramonto/developer_physics.html for guidance on implementing a new 1D external field option.

. . . . . .

Type_vext3D: Controls the 3D potential that will be used if *Ipot_wf_n* is VEXT_3D_INTEGRATED or VEXT_ATOMIC. The specific form of the interactions between the wall atoms and fluid species are specified in Table 11. Options are:

- 0: PAIR_LJ12_6_CS: A cut and shifted 12-6 Lennard-Jones potential.
- 1: PAIR_COULOMB: A coulomb potential. Note that in this potential q_w =wall atom charge, q_f =fluid component charge, and that q_w is set in *dft_surfaces.dat* while q_f is set in the Interaction Potential parameters section of *dft_input.dat*. Finally note that T_{elec} was defined in the section on Functional Switch Parameters under *Type_pairPot*.

Label	$V^{ext}(\mathbf{r})$	3D potential functional form
PAIR_LJ12_6_CS	$u_{wf}(r) - u_{wf}(r_c) \quad r < r_c$ $0 \quad r > r_c$	$u_{wf}(r) = 4\epsilon_{wf}\left[\left(\frac{\sigma_{wf}}{r}\right)^{12} - \left(\frac{\sigma_{wf}}{r}\right)^6\right]$
PAIR_COULOMB		$u(r) = \frac{q_w q_f}{T_{elec} r}$. q_w =wall atom charge, q_f =fluid component charge

Table 11: 3D potential options for computing external fields. Note that the distance r is computed from a point in the fluid to the center of the surface as defined by the $WallPos$ variables set in *dft_surfaces.dat*.

. . . . . .

Ipot_ww_n: An array of switches indicating the type of neutral interactions between the surfaces in the system. Often we leave these terms out of the calculation; however they may be needed in some cases. For example, a calculation of potential of mean force as a function of distance between two solvated atoms, molecules, colloidal particles, or surfaces requires the direct interaction as one contribution. To turn off computation of wall-wall interactions, set the first entry of the array to -2. The code automatically fills the arrays with FALSE (0). To turn on surface interactions for all wall-wall pairs, set the first entry in the array to -1 and this array will automatically be set to TRUE (1). To compute some but not all surface interactions, manually set up the array using the integers 0 for FALSE and 1 for TRUE.

Type_uwwPot: This specifies the type of surface-surface interactions. Options are the same as in Tables 4 and 11. However, the interaction potential parameters are all based on wall-wall parameters, σ_{ww}, ϵ_{ww}, q_w, etc. Again the options are:

- 0: PAIR_LJ12_6_CS
- 1: PAIR_COULOMB

3.6 INTERACTION ENERGY PARAMETERS

These parameters describe the properties of the fluid and wall particles with respect to their interactions. The example shows input for a three component 1:1 electroltye solution interacting with a hard charged wall. Each parameter is described in more detail below for both atomic fluids and polymeric fluids.

==========

Prototype

Ncomp Mix_Type
Mass[icomp]
Charge_f[icomp]
Pol[icomp]
Sigma_ff[icomp,jcomp]
Eps_ff[icomp,jcomp]
Cut_ff[icomp,jcomp]
Bond_ff[icomp,jcomp]
Rho_w[iwall_type]
Sigma_ww[iwall_type,jwall_type]
Eps_ww[iwall_type,jwall_type]
Cut_ww[iwall_type,jwall_type]
Sigma_wf[icomp,iwall_type]
Eps_wf[icomp,iwall_type]
Cut_wf[icomp,iwall_type]

==========

Example

```
************ INTERACTION ENERGY PARAMETERS ******************************
@  3  0            Ncomp  Mix_Type
                   Ncomp = (number of components) OR for polymers ....
                            Ncomp=Nblock_tot (number of polymer block groups)
                   Mix_Type = 0 for L-B rules, =1 for manual input

                        --------FLUID-FLUID INTERACTION PARAMETERS---------
@  1. 1. 1.                      Mass(icomp):icomp=0,Ncomp-1
@  0. 1. -1.                      Charge_f(icomp):icomp=0,Ncomp-1; Valence
@  0. 0. 0.                      Pol(icomp):icomp=0,Ncomp-1; Polarization Parameter

@  1. 1. 1.                      Sigma_ff(icomp,jcomp):(i,j)comp=0,Ncomp-1
@  1. 1. 1.                    Eps_ff(icomp,jcomp):(i,j)comp=0,Ncomp-1
@  2.5 2.5 2.5                 Cut_ff(icomp,jcomp):(i,j)comp=0,Ncomp-1
@  1. 1. 1.                    Bond_ff(icomp,jcomp):(i,j)comp=0,Ncomp-1

                        ----------WALL-WALL INTERACTION PARAMETERS---------
@ 1.                          Rho_w[iwall_type]: density of atoms in surface
@ 1.                         Sigma_ww[iwall_type][jwall_type]:
@ 9. 9. 9.                  Eps_ww[iwall_type][jwall_type]:
@ 2.5 2.5 2.5                        Cut_ww[iwall_type][jwall_type]:

                        ----------WALL-FLUID INTERACTION PARAMETERS---------
@  1.0                       Sigma_wf[i][j] [i=0,Ncomp-1][j=0,Nwall_type-1]
@  1.0                       Eps_wf[i][j]
@  1.0                       Cut_wf[i][j]

***********************************************************************
```

==========

PARAMETER DEFINITION

icomp, jcomp: Index over the number of fluid components $Ncomp$ in the system.

iwall_type, jwall_type: Index over the number of wall types $Nwall\_type$ in the system.

Ncomp: Number of fluid components. For polymer fluids enter the total number of different block types in the system. So, for the case of a one component diblock copolymer, there are two segment types, and *Ncomp* should be 2. All the following arrays then have a length of {*Ncomp*,*Ncomp*}.

Mix_Type: Type of mixing rules to be applied for a given problem. If $Mix\_type = 0$ then the Lorentz-Berthlot rules will be applied. In these cases, only the diagonals of the various arrays must be entered in this section. Then the code will calculate the off-diagonal entries. They are $\sigma_{ij} = 0.5(\sigma_{ii} + \sigma_{ij})$, Bond_ff$_{ij}$=0.5(Bond_ff$_{ii}$+Bond_ff$_{ij}$), Cut_ff$_{ij}$ =Cut_ff$_{ii}$+Cut_ff$_{jj}$, and $\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}}$. In addition, the wall-fluid interaction energy parameters will be calculated as $\sigma_{wf} = 0.5(\sigma_{ww} + \sigma_{ff})$ and $\epsilon_{wf} = \sqrt{\epsilon_{ww}\epsilon_{ff}}$. Finally, the cut-off distance for the wall-fluid interaction is taken to be Cut_wf = Cut_ww + Cut_ff for each wall type/fluid component pair. If Mix_type is set to 1, then every element of the arrays must be manually entered for fluid-fluid, wall-wall, and wall-fluid parameters. Note that depending on what types of fluid and surfaces are being studied many of these parameters may be irrelevent so any number may be found in the input file.

Mass: Array containing the mass of each species. Used if one wants to include the deBroglie wavelength term in the chemical potential expressions. Otherwise set to 1.0.

Charge_f: Array containing the valence associated with each component. Charges are only read in if the fluid is Coulombic.

Pol: Array containing the Polarization parameter. Note that one must have Type_coul=3 for this array to be read or used.

Sigma_ff: Array containing interaction diameters of various fluid-fluid interactions. The dimensions depend on the *Length_ref* parameter set earlier. These parameters are irrelevent for the ideal gas and Poisson-Boltzmann electrolyte cases. Note that all arrays, [*i*][*j*], are entered in the following order: [*i*][*j*]: [0][0],...,[0][Nj-1]; [1][0],...,[1][Nj-1]; ...[Ni-1][0],...,[Ni-1][Nj-1].

Eps_ff: Array containing interaction energy parameters associated with Lennard-Jones fluid-fluid interactions. The dimensions depend on the *Temp* parameter set earlier. These parameters are only read in if Lennard-Jones attractions are included in the calculation.

Cut_ff: Array containing the cut-off lengths for the Lennard-Jones fluid-fluid interactions. The units depend on the *Length_ref* parameter discussed earlier. These parameters are only read in if Lennard-Jones attractions are included in the calculation.

Bond_ff: Array containing bond lengths between various species pairs for polymer calculations. Making the bonds shorter than the corresponding σ is an overlapping sphere model.

Rho_w: An array that contains the density of the solid surfaces of interest. The units depend on the *Density_ref* parameter set earlier.

Sigma_ww: Array containing interaction diameters of various surface types with one another.

Eps_ww: Array containing interaction energy parameters associated with Lennard-Jones wall-wall interactions.

Cut_ww: Array containing the cut-off lengths for the Lennard-Jones wall-wall interactions.

Sigma_wf: Array containing interaction diameters of various fluid-surface interactions. These are only relevent if $Mix\_Type = 1$.

Eps_wf: Array containing interaction energy parameters associated with Lennard-Jones wall-fluid interactions. These are only relevent if $Mix\_Type = 1$.

Cut_wf: Array containing the cut-off lengths for the Lennard-Jones wall-fluid interactions. These are only relevent if $Mix\_Type = 1$.

3.7 POLYMER INPUT PARAMETERS

Here is where necessary information for polymer runs is entered.[5] The example below shows an 8-8 block copolymer. Note that if using the CMS functionals, the c(r) file containing the direct correlation function needs to be generated in some way. This can be done using PRISM as is described later in this document, or by simulation or other means.

==========

Prototype

Npol_comp
Nblock[pol_number]
Block[iblock_tot]
Block_type[iblock_tot]
poly_file
NCrfiles Crfac Cr_file Cr_file2 Cr_file3 Cr_file4
Cr_rad

==========

Example

```
************ POLYMER INPUT PARAMETERS ***************************************
@ 1             Npol_comp:                 Number of (co)polymer components
@ 2             Nblock[pol_num]:        Number of blocks in each copolymer
@ 8 8           block[pol_num][iblock]: Number of segments in each block
@ 0 1           block_type[pol_num][iblock]:    Segment types in each block (start w/0, don't skip)
@ star_4444_sym poly_file:              File containing polymer connectivity
@ 2  0.2  crf8.8_0.7  crf8.8_0.7_xs0.333 crf8.8_0.7_xs0.666 crf8.8_0.7_xs0.999
                                                    NCrfiles  Crfac  Cr_file1  Cr_file2 Cr_file3 Cr_file4
@  0.333 0.666                           Cr_break[i=0;NCr_files-2]
@ 1.0           Cr_rad:                    c(r) radius (units of sigma)

*********************************************************************

==========
```

pol_num: An index that runs over all the polymer components in the system. (e.g. *pol_num=(0,Npol_comp-1)*).

iblock: An index that runs over all the blocks in a given polymer component. (e.g. *iblock=(0,Nblock[pol_num]-1)*).

Npol_comp: Enter the total number of (co)polymer components in the mixture of interest.

Nblock[pol_number]: Enter the number of distinct blocks in each of the polymers of interest.

block[pol_num][iblock]: Enter the number of polymer segments in each block of interest.

block_type[pol_num][iblock]: Enter the type of each block of segments. These must be indexed starting with zero, and must run over all the polymer components and blocks of interest. The order for entering this array is $[0][0], [0][1], ...[0][Nblock-1[0]], [1][0]....[1][Nblock-1[1], ...$

poly_file: The file that contains polymer connectivity and bond symmetries if applicable. See the discussion in section 5.

NCrfiles: The number of direct correlation function files to be read in. Up to 4 are allowed. This is to facilitate careful continuation (or interpolation) between disparate direct correlation functions. (Not relevant to WTC functionals.)

Crfac: The multiplicative factor by which the first direct correlation function will be multiplied. This should be between 0 and 1. The second $c(r)$ will then be multiplied by $(1 - Crfac)$, and the two direct correlation functions will be mixed. Note that this definition should be checked in the source code before computing as the desired mixing may be problem dependent. (Not relevant to WTC functionals.)

[5] Note: If the polymer functionals are not turned on, this section will be skipped.

Cr_file1–Cr_file4: These files contain c(r) data from PRISM calculation. See a further discussion of PRISM in the section 5. Interpolation between the various files will be problem dependent requiring source code modification. (Not relevant to WTC functionals.)

Cr_break$[i = 0; NCr\_files - 2]$: These parameters define the breakpoints between different $c(r)$ files with the understanding that the interpolation is being performed in the solvent fraction parameter space of a two component system. In other words the meaning of this parameter is problem dependent. (Not relevant to WTC functionals.)

Cr_rad: This contains the range of c(r) in units of σ, the segment size. (Not relevant to WTC functionals.)

3.8 SEMI-PERMEABLE SURFACE PARAMETERS

These parameters indicate whether a given surface type is semi-permeable and to what degree. We assume that semi-permeable membranes have a constant (non-infinite) potential in their interior. The lower this potential, the more material can be adsorbed in the surface.

==========

Prototype

Lsemiperm[iwall_type][icomp]
Vext_membrane[iwall_type][icomp]

==========

Example

```
************** SEMI-PERMEABLE SURFACE PARAMETERS *****************
@ -2    Lsemiperm[iwall_type][icomp]; [0][0],[0][1],...[0][Ncomp-1][1][0]...
                 (-2 in first entry will set array to all zeros - FALSE)
                  (-1 in first entry will set array to all ones -  TRUE)
@ 0.0   Vext_membrane[iwall_type][icomp]; [0][0],[0][1],...[0][Ncomp-1][1][0]...
                   (array automatically zeroed if Lsemiperm first entry is -2

*****************************************************************

==========
```

PARAMETER DEFINITION

Lsemiperm: An array of logicals (0=FALSE,1=TRUE) that indicates if any wall type is permeable to any of the fluid components in the system. The example shows the order in which the array must be entered in the file. The entire array is set to FALSE if the first entry in the input file is -2. The entire array is set to TRUE if the first entry in the input file is -1.

Vext_membrane: The non-infinite external field associated with each of the semi-permeable surfaces. Note that if the first entry in Lsemiperm is -2 then this array will all be set to zero, and will not be used by the code.

3.9 STATE POINT PARAMETERS

This section sets the state point (the density) of the bulk fluid.

==========

Prototype

Rho_b[icomp]

==========

Example

```
************** STATE POINT PARAMETERS ****************************************
@ 0.001 0.001   Rho_b[icomp], icomp = 0,Ncomp-1  (or Npol_comp-1 for polymers)
******************************************************************
```

==========

PARAMETER DEFINITION

Rho_b: The bulk number density (units of $\rho\sigma_{ref}^3$) for each fluid component in the system. For polymers, enter the density of each polymer component (Npol_comp). The code will automatically determine the density per segment type.

3.10 CHARGED SURFACE BOUNDARY CONDITION PARAMETERS

This section details the properties of charged surfaces in the system. A variety of options are possible from uniform constant surface charge boundaries to locating specific charged sites on a boundary. The example shows a case where there are two types of surfaces, each with constant surface charge (note that the magnitude of the surface charge is read in *dft_surfacess.dat*). In addition, there is one local charge spread over a diameter of $0.5\sigma_{ref}$ located at the center of the domain.

==========

Prototype

Type_bc_elec[iwall_type]
Nlocal_charge
Charge_loc[icharge]
Charge_Diam[icharge]
Charge_x[icharge][idim]
Charge_type_atoms Charge_type_local

==========

Example

```
************** CHARGED SURFACE BOUNDARY CONDITIONS ***************************
@ 2  2             Type_bc_elec[iwall_type]
@  1               Nlocal_charge, # of local charges
                  (-1 for linear profile of charge between
                      two points aligned with principle axes.  !!!)
@  1.0             Charge_loc[0,Nlocal_charge-1]
@  0.5             Charge_Diam[0,Nlocal_charge-1]
@  0.0 0.0 0.0     Charge_x[0,Nlocal_charge-1][idim];
                           [0][0];[0][1];[0][2];[1][0]..
@  0 0             Charge_type_atoms  Charge_type_local :: values 0,1,2

*******************************************************************

==========
```

PARAMETER DEFINITION

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

Nlocal_charge: The number of local volumetric charges that exist within the computational domain. These charges may or may not be located within boundaries of surfaces from which solvent is precluded. They just represent source terms in the system of equations. The flag of Nlocal_charge= -1 indicates that two charges at two locations in the domain will be entered and there should be a linear distribution of charge between these two points.

Charge_loc: This array (of length Nlocal_charge) stores the total charge associated with each site of local charge.

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

Charge_x: This array (of length [Nlocal_charge][Ndim]) stores the positions of the centers of the local volumetric sources of charge. The example shows the order in which this array is read into the input file.

Charge_type_atoms and Charge_type_local: This parameter indicates how atomic or local charges are to be smeared in the domain. The options are:

- *Charge_type_xxx*=0 : Smear the charges over the indicated diameter. For a local charge this would be given by *Charge_diam*. For an atomic charge, this would be indicated by $\Sigma_w w$.

- *Charge_type_xxx*=1 : Approximate point charges by putting all the charge in one element at the center of the atom (given by *WallPos[]*) or local charge (given by *Charge_x*).
- *Charge_type_local*=2 : Smear the charge over every element in the domain. This is a uniform background charge. It is only available for the local charges - not the atoms.

3.11 DIELECTRIC CONSTANT PARAMETERS

This section defines how dielectric constants will be treated in the system.[6]

==========

Prototype

Type_dielec
Dielec_bulk *Dielec_pore* *Dielec_X*
Dielec_wall[iwall_type]

==========

Example

```
************** DIELECTRIC CONSTANTS ***********************************
@ 0                Type_dielec
@ 1.0  0.5  2.0    Dielec_bulk  Dielec_pore  Dielec_X
@ 1.0  1.0 1.0     Dielec_wall[i]  i=1,Nwall_type
**********************************************************************
```

==========

PARAMETER DEFINITION

Type_dielec: This flags sets how dielectric constants will be treated for a given run. Options are: set ϵ constant everywhere in domain including surfaces (option 0), give surfaces and fluid different dielectric constants (option 1), give *pore* fluid and *bulk* fluid different dielectric constants (option 2), or set constant ϵ in walls, but make it a function of density in the fluid (option 3). All dielectric constants are read in as a function of the reference value, ϵ_{ref}. For water, $\epsilon_{ref} = 78$.

Delec_bulk: The ratio ϵ/ϵ_{ref} in the bulk fluid. This should be 1.0 provided the bulk is used as the reference fluid.

Dielec_pore: The ratio ϵ/ϵ_{ref} in the pore fluid.

Dielec_X: The distance from the surfaces that will be considered *pore* fluid.

Dielec_wall[]: An array (of length Nwall_type) that stores the ratio ϵ/ϵ_{ref} of each surface in the system.

[6] This section is only read in if $Type\_coul > -1$.

3.12 DIFFUSION PARAMETERS

This section sets boundary conditions for steady-state transport calculations. In these cases, there is a chemical potential gradient in the system, so one bulk density state point is not sufficient to define the system.

==========

Prototype

Lsteady_state
Grad_dim L1D_bc X_1D_bc
x_const_mu
Geom_Flag Nseg
Radius_L Radius_R Length
Rho_b_Left[icomp]
Rho_b_Right[icomp]
D_coef[icomp]
Elec_pot_L Elec_pot_R
Velocity

==========

Example

```
************* DIFFUSION PARAMETERS ******************************
@   0  0    Lsteady_state  (0=equilibrium problem, 1=steady state problem)
@   0       Grad_dim direction of gradient (0=x, 1=y, 2=z)
@   5.0     x_const_mu (on both sides of domain).
@   0  1    Geom_Flag; (0=unit area; 1=cyl pore; 2=vary pore) Nseg (# pore segments)
@   2.5 0.75 4.    Radius_L, Radius_R, Length
@   0.141  0.312   Rho_b_Left[Icomp]  B.C. on left or bottom or back
@   0.312 0.141    Rho_b_Right[Icomp] B.C. on right or top or front
@   0.32 0.32 4.e-6  4.e-7   D_coef[icomp] Diff Coeff per component (cm^2/sec)
@  0. 0.0   Elec_pot_L,      Elec_pot_R B.C. on elec. potential lbb and rtf
@   -0.05  -0.035            Velocity
*****************************************************************
```

==========

PARAMETER DEFINITION

Lsteady_state: Logical (0 =FALSE, 1 =TRUE) that indicates whether this run is a steady state calculation with inhomogeneous boundary conditions. If Lsteady_state = 0, the remainder of the lines in this section are not read in.

Grad_dim: This constant indicates in which dimension the chemical potential gradient exists. Currently, the code is only set up to allow inhomogeneous boundary conditions in one dimension. The options are $0 = x$, $1 = y$, and $2 = z$.

L1D_bc: A logical (0=FALSE : 1=TRUE) indicating whether or not a 1D boundary condition should be applied at the ends of a 2D or 3D box in the Grad_dim direction. This is useful when studying electrolytes and the Debye length is significantly longer than the local structue near a surface at the center of the box. Then much of the computational cost of increasing box size to account for the proper decay can be mitigated.

X_1D_bc: The distance over which the 1D boundary should be applied.

x_const_ mu: This constant indicates the distance in the *Grad_dim* dimension where the chemical potential is held constant. This constant region will be a bulk fluid far enough away from any surfaces in the transport region.

Geom_Flag: This flag indicates if the area of a pore varies in the *Grad_dim* dimension. This switch is only active for 1D calculations.

Nseg: This constant sets how many pore segments of different geometry there are for a given pore. For example if there is a cylindrical pore with tapered ends, there would be three segments.

Radius_L: An array of length *Nseg* that stores the radius of the left (or bottom or back) side of the pore segment.

Radius_R: An array of length *Nseg* that stores the radius of the right (or top or front) side of the pore segment.

Length: An array of length *Nseg* that stores the length of each pore segment.

Rho_b_Left: An array of length *Ncomp* that stores the constant bulk densities on the left (or bottom or back) boundary condition. Note that for these inhomogeneous problems, only constant bulk boundaries (with different values) are allowed.

Rho_b_Right: An array of length *Ncomp* that stores the constant bulk densities on the right (or top or front) boundary condition. Note that for these inhomogeneous problems, only constant bulk boundaries (with different values) are allowed.

D_coeff: An array of length *Ncomp* that stores the diffusion coefficient for each species in the problem of interest.

Elec_pot_L: This constant stores the electric potential on the left (or bottom or back) side of the domain.

Elec_pot_R: This constant stores the electric potential on the right (or top or front) side of the domain.

Velocity: This constant accounts for center of mass motion in transport in these steady state problems.

3.13 STARTUP CONTROL PARAMETERS

These settings determine how a given run will be started. The variety of options is optimized for a variety of problems from studying wetting to studying steady state transport.

==========

Prototype

Iliq_vap
Iguess
Nsteps
Orientation_step[istep]
Xstart_step[istep]
Xend_step[istep]
Rho_step[icomp][istep]
Restart
Rho_max

==========

Example

```
************** STARTUP CONTROL PARAMETERS ******************************
@  -1            Iliq_vap (-2=don't compute coexistence
                                 -1=compute coexistence,
                                   1=compute a Wall-Vapor profile,
                                   2=compute a Wall-Liquid profile
                                   3=compute a Liquid-Vapor profile)
@  -3          Iguess
                     -3:    Constant Bulk Density
                     -2:    Constant liquid coexistence density
                     -1:    Constant vapor coexistence density
                      0:     rho_bulk*exp(-Vext/kT)
                      1:     rho_liq*exp(-Vext/kT)
                      2:     rho_vap*exp(-Vext/kT
                      3:     step function
                      4:     chopped profile: to rho_bulk
                      5:     chopped profile: to rho_liq
                      6:     chopped profile: to rho_vap
                      7:     chopped profile: to rho_step
                      8:     linear profile

@  1                 Nsteps
@  0                Orientation_step[istep]
@  -5.0  -5.0    Xstart_step[istep]
@   5.0  5.0      Xend_step[istep]
@  0.1 0.1        Rho_step[icomp][istep]

@ 0              Restart
                         0=no - use Iguess to determine initial guess
                         1=yes - use guess from dft_dens.dat
                         2=yes, but w/ step function at Xstart_step[0]
                         3=yes for densities but not elec.pot or chem.pot.
                         4=yes also restart external field from file
                         5=use 1D profile as initial guess for 2 or 3D calculation
@  1000.     Rho_max
************************************************************************

==========
```

PARAMETER DEFINITION

Iliq_vap: Indicates whether this run is to use the entered state point parameters *Rho_b* or whether the profile should be set up for a fluid at liquid-vapor coexistence. In the latter case, the *Temp* parameter is used to first find coexistence, and then the bulk densities/chemical potentials are set to coexistence values. The choices are: off coexistence and don't calculate coexistence properties (-2); off coexistence, but want to calculate the bulk coexistence properties (-1), at coexistence on the vapor side (1), at coexistence on the liquid side (2), doing a liquid-vapor interface (3). This parameter is particularly useful when performing wetting studies or calculating contact angles. Liquid-vapor coexistence can only be calculated at this time for Lennard-Jones fluids of one component. So for all other systems, set Iliq_vap to -2.

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

- -3: CONST_RHO: a constant density with $\rho = \rho_b$.
- -2: CONST_RHO_L: a constant density with $\rho = \rho_{coex}^{liq}$.
- -1: CONST_RHO_V: a constant density with $\rho = \rho_{coex}^{vap}$.
- 0: EXP_RHO: an ideal gas profile with $\rho = \rho_b \exp(-V_{ext}/kT)$.
- 1: EXP_RHO_L: an ideal gas profile with $\rho = \rho_{coex}^{liq} \exp(-V_{ext}/kT)$.
- 2: EXP_RHO_V: an ideal gas profile with $\rho = \rho_{coex}^{vap} \exp(-V_{ext}/kT)$.
- 3: STEP_PROFILE: a stepped profile defined in more detail by the parameters *Nsteps*, *Orientation_step*, *Xstep_start*, *Xstep_end*, and *Rho_step*.
- 4: CHOP_RHO: a chopped profile where an old density profile is read in, and then the profile is chopped off at a distance *Xstep_start[0]* from the surfaces. The remainder of the profile is replaced with ρ_b.
- 5: CHOP_RHO_L: Same as CHOP_RHO except the remainder of the profile is replaced with ρ_{coex}^{liq}.
- 6: CHOP_RHO_V: Same as CHOP_RHO except the remainder of the profile is replaced with ρ_{coex}^{vap}.
- 7: CHOP_RHO_STEP: Same as CHOP_RHO except the remainder of the profile is replaced with *Rho_step*[0].
- 8: LINEAR: linear profile between ρ_{Left} and ρ_{Right} defined in the steady-state diffusion section. Use this option for steady-state problems with inhomogeneous boundary conditions.

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

- 0: No restart file: use guess indicated by *Iguess*.
- 1: Restart from default files. The files that are needed to perform a restart are: dft_dens.dat for all systems, and in addition dft_dens.datg for CMS polymer calculations. The former contains the density and field variables while the latter stores the chain information in the G functions. In order to do binodal calculations, the files dft_dens2.dat (and dft_dens2.datg for CMS polymers) will be required, which give the second profile on which binodal calculations are performed.
- 2: Restart from files, but step the profile as indicated by *Iguess* parameter
- 3: Restart density unknowns only. Set nonlocal densities, $n(\mathbf{r})$, electric potential, $\psi(\mathbf{r})e/kT$, or $G(\mathbf{r})$ variables to simple initial guesses.
- 4: Restart solution fields from files. Also restart external field from the files *dft_vext.dat* and *dft_zeroTF.dat*.
- 5: Use a 1-dimensional solution field as an initial guess for a 2 or 3 dimensional calculation. The 1-dimensional solution is simply replicated in the y and/or z directions.

Rho_max: This is the maximum density allowed for continuation from an old profile when MESH continuation is performed. Again very large densities can be difficult to converge.

3.14 OUTPUT FORMAT PARAMETERS

These parameters just set the format for various output files and parameters in the code. The optimal output format usually depends on what one is trying to study.

==========

Prototype

Lper_area Lcount_reflect Lprint_gofr Lprint_uww
Print_rho_type
Print_rho_switch Print_mesh_switch
IWRITE

==========

Example

```
************ OUTPUT FORMAT PARAMETERS ************************************
             ************************************************************
             **** set how you would like all of the output to print ******
             ************************************************************
@  0 1 0 0        Lper_area Lcount_reflect  Lprint_gofr  Lprint_uww
@  0         Print_rho_type
@  1   0     Print_rho_switch    Print_mesh_switch:
@  1         IWRITE (0=MINIMAL, 1=DENSITY_PROF, 2=NO_SCREEN, 3=VERBOSE)
**********************************************************************
```

==========

PARAMETER DEFINITION

Lper_area: Switch to indicate how the user would like principle output parameters (Adsorption, Free Energy, Charge in the fluid, and Force) to be printed. Note that this is a tricky quantity to define for any complex 3-dimensional system. However, it can be useful to report per unit area results for simple surface geometries. Options include:

- Give extensive result - don't divide by area (option 0). It should be noted that 1D problems are inherently reported per unit area, and 2D problem results are reported per unit length.
- Report based on the total exposed surface area in the system (option 1).

Lcount_reflect: If TRUE (1) compute energies and adsorptions based on the total domain accounting for reflections. If FALSE (0) just compute for the computational box ignorning reflections.

Lprint_gofr: For the special case of one fixed atom or molecule in a fluid, it is of interest to compute g(r). In this case the radial distance from every mesh point is calculated and the normalized density, $\rho(r)/\rho_b$ is reported as a function of r from the central atom of interest.

Lprint_uww: Print out the direct surface-surface interactions as well as the excess surface free energy.

Print_rho_type: This switch sets the way the solution data ($\rho(\mathbf{r})$,$n(\mathbf{r})$, $\psi(\mathbf{r})$, $\mu(\mathbf{r})$, $G(\mathbf{r})$) will be written to files. The options are:

- 0: Put all output in dft_dens.dat, dft_dens2.dat (Lbinodal=TRUE), or dft_dens.datg (CMS_POLYMER). This option overwrites the file as continuation proceeds.
- 1: Put the output from each run in a different file numbered as dft_dens.0, dft_dens.1, etc. (given continuation in one field only).

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

- 0: Include only densities as $\rho_b\sigma^3$.
- 1: Include the value of p/p_{sat} where p_{sat} is the saturation pressure of the fluid at the given temperature. This is applicable only to a 1 component LJ fluid.
- 2: Include the Debye screening length k^{-1}: This is applicable only to electrolyte fluids.
- 3: Include chemical potentials, μ/kT.

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

IWRITE: This switch controls how much output will be printed. For minimal output (no density profiles) enter 0; for minimal output plus density profiles enter 1; to eliminate all screen output enter 2, or for verbose printing enter 3. The files printed in each case are detailed below in the section on output files.

3.15 COARSENING SWITCHES

These parameters provide a variety of ways to coarsen the integrals in the residual and/or Jacobians in order to save computational cost.

==========

Prototype

Nzone
Rmax_zone[izone]
Coarsen_resid
Coarser_jac Esize_jacobian
Ljac_cut Jac_threshold
Matrix_fill_flag

==========

Example

```
********** COARSENING SWITCHES **********************************************
@  1             Nzone (Coarsens Mesh/Jacobian by factor of 2)
@  0.0           Rmax_zone[Nzone-1] [0.0 for complete coarsening]
@  0             Coarsen_Residual ? (0=NO, 1=YES)
@  0  0.25       Coarser_jac; Esize_jacobian
@  0   100.      Ljac_cut   Jac_threshold
*****************************************************************
```

==========

PARAMETER DEFINITION

Nzone: The number of zones the domain will be split into based on the geometry of the surfaces. These zones will control where residual or Jacobian coarsening is applied.

Rmax_zone[]: An array containing the distances from the surfaces where the various zones are active. The length of the array is *Nzone-1*. This zoning works for any surface geometry of interest.

Coarsen_resid: Logical (0 =FALSE : 1 =TRUE) to indicate if the residual equations will be coarsened according to the zones. In this case, the coarsening occurs in powers of two for each zone away from the surface, and the residual equations at the coarsened nodes are changed from the euler-lagrange equations (or other equations such as nonlocal density equations) to a linear average of the surrounding nodes. These averaging equations are practically free of computational cost compared with all other equations in the system.

Coarser_jac: This flag sets the type of Jacobian coarsening that will be performed for a given calculation. The options are:

- 0: No Jacobian coarsening beyond the coarsening of the zones and residuals.
- 1: Coarsen the Jacobian integrals in the finest zone (nearest the surfaces) by an extra power of two.
- 2: Coarsen all but most coarse zone by an extra factor of two.
- 3: Use the mesh spacing in the coarsest zone for Jacobian integrals in every zone.
- 4: Use the mesh spacing in the second to the coarsest zone for Jacobian interals everywhere except in the coarsest zone.
- 5: Use a constant mesh spacing (*Esize_jacobian*) to define Jacobian integrals everywhere in the domain.

Esize_jacobian: See previous definition (option 5 only).

Ljac_cut: A logical that indicates whether the Jacobian integrals will be cut off at some threshold value (see next definition). This may be useful for attractive systems with rather long cutoffs.

Jac_threshold: The threshold value for whether or not to include a given point in the integration stencil. A value of 100 indicates that you will reject all points smaller than 100 times less than the maximum value in the stencil.

3.16 NONLINEAR SOLVER PARAMETERS

These parameters control the Newton's method constraints and the load balancing options.

==========

Prototype

Max_Newton_iter
Newton_rel_tol *Newton_abs_tol* *Min_update_frac*
Load_Bal_Flag

==========

Example

```
************ NONLINEAR SOLVER PARAMETERS **********************************
@ 50               Maximum # of Newton Iterations
@ 1.0e-5 1.0e-10 0.2  Relative conv. tol,  Absolute conv. tol., min update fraction
@ 2                Load balance switch (0=linear,1=box,2=weights,3=timings)
***************************************************************
```

==========

PARAMETER DEFINITION

Max_Newton_iter: The maximum number of Newton iterations the code will peform before exiting.

Newton_rel_tol: Relative convergence tolerance for the nonlinear solver.

Newton_abs_tol: Absolute convergence tolerance for the nonlinear solver.

Min_update_frac: During the Newton-Raphson solve, the code will mix some fraction of the new solution in with the old solution at each iteration. This parameter sets the minimum fraction of the new solution used. The code will actually use a fraction between Min_update_frac and 100% (of the new solution).

Load_Bal_Flag:

3.17 LINEAR SOLVER PARAMETERS

These parameters define how the iterative linear solves (Aztec) will be performed.

==========

Prototype

L_Schur Az_solver Az_kspace
Az_scaling
Az_preconditioner Az_ilut_fill_param
Max_gmres_iter Az_tolerance

==========

Example

```
************ LINEAR SOLVER PARAMETERS ***************************************
@  0 0 100    L_Schur  Az_solver (0=gmres, 1=cg, 2=tfqmr, 3=cg2, 4=bicgstab)  Az_kspace
@  0           Scaling (-1=none, 0=row_sum, 1=Jacobi, 2=symrow_sum)
@  -1    4     Preconditioner (-1=none, 0=ilu, 1=Jacobi, 2=symGS, 3=LSpoly3, ilut), Az_ilut_param
@ 100 1.0e-6     Max iterations and Convergence Tolerance for Linear Solver
*****************************************************************
```

==========

PARAMETER DEFINITION

L_Schur: Flag for using the Schur solvers, as described in [12]. Options are: 0: don't use Schur solver; 1: do use Schur solver. For many problems the Schur solvers result in faster convergence.

Az_solver: If *L_Schur* = 0, then the code uses the linear solver specified here.

Az_kspace:

Az_scaling:

Preconditioner: Only used for *L_Schur* = 0. Some amount of preconditioning is usually helpful for convergence.

Az_ilut_param: Sets the level of preconditioning to use.

Max_gmres_iter: Maximum number of iterations for the linear solver.

Az_tolerance: Convergence tolerance for the linear solver.

3.18 MESH CONTINUATION PARAMETERS

This section sets parameters for performing mesh continuation runs. This type of continuation is not allowed by the LOCA libraries because the mesh cannot be changed continuously.

==========

Prototype

N_runs
Del_1[idim=0 to Ndim-1]
Plane_new_nodes Pos_new_nodes
Guess_range[0] Guess_range[1]

==========

Example

```
************ MESH CONTINUATION PARAMETERS ***********************************
              Here you enter information for mesh continuation.
              All other types are handled by LOCA
@  10      N_runs
@  -.2 0.0 0.0        Del_1[idim=0;Ndim]  How much to change parameter.
@  0   0      Plane_new_nodes     Pos_new_nodes
                 (0=yz,1=xz,2=xy)   (-1=lbb,0=center,1=rtf)
@ 0 0  Guess_range[]
*****************************************************************
```

==========

PARAMETER DEFINITION

N_runs: The number of mesh continuation steps to be performed. To not perform mesh continuation (i.e. just do 1 run), set N_runs = 1.

Del_1[]: This array stores the amount the mesh size will be varied for each run. Note that all the mesh dimensions (x, y, and z) may be changed simultaneously.

Plane_new_nodes: Mesh continuation involves changing the mesh size by insertion or deletion of plane(s) of nodes. The orientation of the new plane(s) of nodes must be indicated. The options are: yz-plane (0), xz-plane (1), and xy-plane (2).

Pos_new_nodes: The position of the new plane(s) of nodes must also be indicated. The options are to add/delete the plane(s) of nodes from one of three positions: left-bottom-back (-1), center of box (0), and right-top-front (1).

Guess_range[]: This array stores two surface separations, and is used when performing mesh continuation specifically for the case of two interacting surfaces. Usually this type of calculation corresponds to potential of mean force calculations. Guess_range[0] stores the maximum surface separation where Rho_b will be used as an initial guess while Guess_range[1] stores the minimum surface separation where the previous solution will be used as an initial guess. This heuristic parameter has been implemented because there are times when very steep density profiles between two surfaces will not allow for easy convergence, and it is better to start over from a bulk density initial guess. Between Guess_range[0] and Guess_range[1], a mixing of the old solution and the bulk density is performed.

3.19 ARC-LENGTH CONTINUATION PARAMETERS

These parameters define how arc-length continuation will be performed. The algorithms are described in [13].

==========

Prototype

Continuation Method
Continuation parameter Scale_fac
Step_size
N_steps step control aggressiveness
Second continuation parameter

==========

Example

```
************ ARC-LENGTH CONTINUATION PARAMETERS**********************
@  2      Continuation Method (-1=None; 0,1,2=0th, 1st, arc-length
                          3=Spinodal (Turning Point); 4=Binodal (Phase Eq))
@  2   1.0    Continuation parameter :  Scale_fac (for CONT_SCALE cases only)
                         CONT_TEMP        1   /* State Parameters */
                         CONT_RHO_0       2
                         CONT_RHO_ALL     3
                         CONT_LOG_RHO_0   4
                         CONT_LOG_RHO_ALL 5
                         CONT_SCALE_RHO   6

                         CONT_EPSW_0      7    /* Wall-Wall Energy Params */
                         CONT_EPSW_ALL    8
                         CONT_SCALE_EPSW  9

                         CONT_EPSWF00     10    /* Wall-Fluid Energy Params */
                         CONT_EPSWF_ALL_0 11
                         CONT_SCALE_EPSWF 12

                         CONT_EPSFF_00    13   /* Fluid-Fluid Energy Params */
                         CONT_EPSFF_ALL   14
                         CONT_SCALE_EPSFF 15

                         CONT_SCALE_CHG   16  /* Charged surface params */
                         CONT_SEMIPERM    17  /* Vext_memebrane */
                         CONT_WALLPARAM   18  /* WallParam */

                         CONT_CRFAC        19      /* Mixing parameter for 2 cr files */

@  0.05   Parameter initial step size
@  10  0.25  N Steps,  Step Control Aggressiveness  (0.0 = constant step)
@  2       Second parameter for Spinodal and Binoadal Calculations (Same list).
*********************************************************************
```

==========

PARAMETER DEFINITION

Continuation method: This flag specifies which continuation method to use. Options are:

- -1: none
- 0: 0th order continuation using a constant step (non-adaptive).

- 1: 1st order continuation.
- 2: 2nd order, i.e. arc-length, continuation.
- 3: Track spinodal (turning) points. If restarting, this requires two density files as input (dft_dens.dat and dft_dens2.dat).
- 4: Track binodal points, i.e. two different state points with the same free energy. Use for following phase transitions. If restarting, this option also requires two density files as input, containing the two different density profiles at the desired free energy.

Continuation parameter: The continuation is a function of this parameter, which must be a continuous variable in the system. Various options are provided, and more can be added by adding extra cases into dft_continuation.c. The options shown in the example include continuation in temperature, various combinations of bulk densities, various energetic ϵ parameters, charges, wall permeabilities, and scale factors for CMS-polymer $c(r)$ functions. It is advisable to check the source code in dft_continuation.c to determine exactly which variable is being changed in each option.

Scale_fac: Some of the continuation variables can be changed by a multiplicative scaling factor given by this value, as opposed to increasing or decreasing the parameter. In these cases, *Scale_fac* itself is the continuation parameter. These cases are identified by the word SCALE in the name.

Step_size Initial value for the change in the parameter. This will be adjusted by the algorithm as continuation proceeds. Large values will sometimes fail to converge, while small values can lead to very small step sizes along a continuation curve.

N_steps: The number of continuation steps to perform.

step control aggressiveness: A measure of how fast the algorithm will step along the continuation curve.

Second continuation parameter: For binodal calculations (method 4), this is the second parameter that is varied in order to keep the two systems at the same free energy. It can be any of the implemented continuation parameters in the code. For example, one might have a phase transition at a given chemical potential (Rho_b) and temperature, and the goal is to track the transition as a function of temperature. Then the temperature should be the first continuation variable, that will be increased or decreased as specified by *Step_size*, and the chemical potential variable will be the second parameter and this will be adjusted by the code as necessary in order to maintain the system at the phase transition.

4 *dft_surfaces.dat*

The second required input file (except in the case where there are no surfaces in the domain) contains the locations of the centers of the surfaces of interest. This data must be located in a file called *dft_surfaces.dat*. The first column should indicate the type of the surface of interest, the second column indicates the linkage of the surface with other surfaces, the next *Ndim* columns give the position of the center of the surface of interest, and finally the last column gives the charge on each surface. The number of position coordinates that should be found in this file is equal to the number of dimensions in the calculation. Unique surface types are distinguished by any property from surface shape to interaction strengths. The array of surface types starts with zero (C convention). An example of the *dft_surfaces.dat* file for a case with 4 surfaces, 2 surface types, and 3 independent surfaces is shown below.

The *Link* array is used to denote which of the individual surfaces are grouped into compound surfaces. Thus every surface with the same *Link* parameter is considered to be part of the same compound surface. It is desirable to link surfaces together in a variety of circumstances. One example is if all the surfaces are individual atoms of one macromolecule. Another example is if one wants to study a chemically heterogeneous surface. In the latter case two surface types are needed, and the walls are positioned so they touch each other. The linkage array then indicates that they are really part of the same surface. Linkage is particularly important when one wants to calculate the force on a compound surface.

==========

Prototype for *dft_surfaces.dat* **file**

WallType[iwall] Link[iwall] WallPos[iwall][idim=0] WallPos[iwall][idim=1] WallPos[iwall][idim=2] Elec_param_w[iwall]

==========

Example for *dft_surfaces.dat* **file**

```
0  0   0.0  0.0  0.0   0.0
1  0   1.0  0.0  0.0   0.0
0  1   0.0  1.0  0.0   0.0

1  2   0.0  0.0  1.0   0.0
```

==========

PARAMETER DEFINITION

iwall: The index over the number of surfaces, *Nwall*, in the system *iwall*={*0,Nwall-1*}.

idim: The index over the number of dimensions, *Ndim* in the system *idim*={*0,Ndim-1*}.

WallType: An array containing the type of each surface in the system.

Link: An array containing the linkage of each surface to a larger compound surface.

WallPos: An array containing the position of the center of each surface in the system. Note that if the flag -9999. is entered in any of the wall positions, the surface will be randomly placed in the domain. If conditions on the placement are desired (e.g. no overlaps) additional code must be added to the file *dft_input.c*. Also note that the origin is located at the center of the computational domain in the code. So, the surface positions should lie between *-0.5Size_x[idim]* and *0.5size_x[idim]*.

Elec_param_w: An array containing the electrostatics parameter of every surface. This can be a surface charge (given as $q\sigma^2/e$), surface potential (given as $\psi_s e/kT$), or partial charge (given as Q/e) as defined in the array Type_elec_bc. If a surface is neutral, enter 0.0 in this array.

5 *poly_file*

We now briefly describe the polymer connectivity file. This file simply identifies the architecture of the chains by defining the number of bonds at each segment as well as the bead numbers to which a given segment is bonded. This file also identifies symmetries in the chain so that redundant equations may be removed from the system of equations. The example shows the case of a star polymer where a central bead is connected to 4 identical arms. Each of the arms is four beads in length, and the symmetries are enumerated.

==========

Prototype for *poly_file* **file**

Nbond[pol_number][iseg] Bond[pol_number][iseg][ibond] Pol_Sym[iunk_bond]

==========

Example for *star_4444_sym* **file**

```
2 -1 -1 1 -1
2  0 -1 2 -1
2  1 -1 3 -1
2  2 -1 4 -1
4  3 -1 5 -1 9 -1 13 -1
2 4  7  6  6
2 5  5  7  4
2 6  3  8  2
2 7  1 -1  0
2 4   7 10  6
2 9   5 11  4
2 10  3 12  2
2 11  1 -1  0
2 4   7 14   6
2 13  5 15  4
2 14  3 16  2
2 15  1 -1  0
```

==========

PARAMETER DEFINITION

pol_number: The index over the polymer component of interest.

iseg: The index over the polymer segments (beads).

Note that each line in the file belongs to a particular polymer component and segment, with the order being that first the segments 0–N_0 are listed on polymer 0, then the segments 0–N_1 on polymer 1, etc.

ibond: An index over the bonds connected to a given segment.

iunk_bond: An index that runs from 0 to $\sum_{pol\_comp=1}^{Npol\_comp} \sum_{iseg=1}^{Nseg[pol\_comp]} Nbond[pol\_comp][iseg]$ that is used as an index into the bond equation (G) unknowns. The ith bond entry in the *poly_file* represents the unknown $iunk\_bond = i-1$.

NOTE: For mixtures, the polymer components must come in the same order as defined in the polymer section of the file *dft_input.dat*.

Nbond[pol_number][iseg]: The total number of bonds connected to a given polymer segment. Note that for end groups, *Nbond+1* is entered because there are still two G equations associated with that segment.

Bond[pol_number][iseg][ibond]: An array containing the segments to which a given [pol_number][iseg] are bonded. The segments of each polymer component should be numbered starting with 0, and every end segment should have -1 as one of the bond entries to identify that segment as an end group to the code.

Pol_Sym[iunk_bond]: An array that identifies symmetries in the polymer(s) of interest. This entry should be $iunk\_bond = -1$ if there are no symmetries. Otherwise enter another bond number ($iunk\_bond = junk\_bond$) which is identical to the current entry. For symmetries, the G equation defined above will be replaced with

$$G[iunk\_bond] = G[junk\_bond] \tag{1}$$

6 *Cr_file.dat*

Finally, when using the CMS polymer functionals, an input of the direct correlation function is required. This input could come from PRISM calculations, from simulations, or from experiments. Note that $c(r)$ does not need to be defined on the same mesh as the rest of the DFT problem, so e.g. in the example, the spacing between r values is 0.04, while the mesh size could be anything, e.g. 0.05 or 0.1, or.... The code currently takes the $c(r)$ for a repulsive chain with the same bond patterns as the polymer of interest to the DFT calculations. Attractions are added using the random phase approximation by assuming $c(r) = -u^{att}(r)$ for $r > d$ as described earlier. An example of the input that will be used for a study of an 8-8 diblock copolymer is shown below.

==========

Prototype for *Cr_file.dat* **file**

r Cr[ipol_comp][jpol_comp]

==========

Example for *crf8.8_0.7* **file**

```
0.40000E-01    -0.58756E+01    -0.58756E+01    -0.58756E+01
0.80000E-01    -0.56529E+01    -0.56529E+01    -0.56529E+01
0.12000E+00    -0.54115E+01    -0.54115E+01    -0.54115E+01
0.16000E+00    -0.51665E+01    -0.51665E+01    -0.51665E+01
0.20000E+00    -0.49236E+01    -0.49236E+01    -0.49236E+01
0.24000E+00    -0.46829E+01    -0.46829E+01    -0.46829E+01
0.28000E+00    -0.44461E+01    -0.44461E+01    -0.44461E+01
0.32000E+00    -0.42130E+01    -0.42130E+01    -0.42130E+01
0.36000E+00    -0.39847E+01    -0.39847E+01    -0.39847E+01
0.40000E+00    -0.37611E+01    -0.37611E+01    -0.37611E+01
0.44000E+00    -0.35430E+01    -0.35430E+01    -0.35430E+01
0.48000E+00    -0.33305E+01    -0.33305E+01    -0.33305E+01
0.52000E+00    -0.31242E+01    -0.31242E+01    -0.31242E+01
0.56000E+00    -0.29242E+01    -0.29242E+01    -0.29242E+01
0.60000E+00    -0.27313E+01    -0.27313E+01    -0.27313E+01
0.64000E+00    -0.25455E+01    -0.25455E+01    -0.25455E+01
0.68000E+00    -0.23676E+01    -0.23676E+01    -0.23676E+01
0.72000E+00    -0.21977E+01    -0.21977E+01    -0.21977E+01
0.76000E+00    -0.20365E+01    -0.20365E+01    -0.20365E+01
0.80000E+00    -0.18843E+01    -0.18843E+01    -0.18843E+01
0.84000E+00    -0.17417E+01    -0.17417E+01    -0.17417E+01
0.88000E+00    -0.16091E+01    -0.16091E+01    -0.16091E+01
0.92000E+00    -0.14871E+01    -0.14871E+01    -0.14871E+01
0.96000E+00    -0.13762E+01    -0.13762E+01    -0.13762E+01
0.10000E+01    -0.63836E+00    -0.63836E+00    -0.63836E+00
```

==========

PARAMETER DEFINITION

r: The first column of numbers contains the distance r in $c(r)$.

C(r): The 2nd-nth columns contain direct correlation function data in the following order:
- the self-terms, *ii*, are listed first.
- the cross-terms, *ij* follow.

In the example given above, there are three columns because the calculation of interest will have an 8-8 diblock copolymer. The first column is read in to correspond to AA interactions, the second corresponds to BB interactions, and the third corresponds to AB interactions.

7 Introduction to output files

Here we briefly discuss output generated by the Tramonto code. There are a variety of files that are generated only if the code is operating in VERBOSE mode. The primary output is always printed with the exception that the printing of density files can be turned off with $Iwrite = 0$. In addition, we note that all screen output is turned off if $Iwrite = 2$. Both of these options can be useful to prevent needless I/O, and can be particularly important when performing coupled CBMC-DFT calculations.

7.1 Primary Output

7.1.1 *dft_dens.dat*

This file contains the solution vector fields as a function of the position in the mesh. The first three columns are x, y, and z. The following columns are the solution vector. The first few lines in the file summarize the variables that are printed. Depending on the calculation performed, these include the densities (for each fluid component), electrostatic field, chemical potentials, CMS-polymer mean fields, hard sphere nonlocal densities, cavity correlation functions for WTC polymer functionals, and bonding nonlocal densities for WTC polymer functionals. Note that for binodal tracking calculations, a second density file is printed in *dft_dens2.dat*. Also note that the output can be printed to sequentially numbered files (e.g. *dft_dens.0*, *dft_dens.1*, etc.) using the *Print_rho_type* parameter described earlier.

7.1.2 *dft_dens.datg*

This file contains the results for the propagator equations on the mesh for the case of CMS polymer fluids.

7.1.3 *dft_output.dat*

This file contains the principle output of the code. Currently it prints any continuation variables, the number of nonlinear iterations, time to solve, adsorptions, charges, forces, and free energies. Note that the first line of output for any continuation run will enumerate the parameter names as well as the values to facilitate identification of the various columns in the file.

7.1.4 *dft_out.lis*

This file echos the input file and provides details of some of the critical setup parameters including number of boundary nodes and surface elements.

7.1.5 *dft_time.out*

Some timing information on the solves.

7.2 Debugging Output

These files are only produced when the Iwrite switch is set to VERBOSE (3). They are useful for a variety of debugging tasks.

7.2.1 *cr.out*

For CMS polymers, this file prints back out the direct correlation function, as given by the *Cr_file* (see Sec. 6).

7.2.2 *cr.lj.out*

For CMS polymers, this file contains the direct correlation function after the LJ attractions have been added to it.

7.2.3 *dens_iter.dat*

Stores the solution vector as a function of position for each Newton iteration.

7.2.4 *dft_freen_prof.dat*

For one dimensional systems, $Ndim = 1$, this file contains the free energy density as a function of position. This is related to the surface tension profile of an interface with planar symmetry.

7.2.5 *dft_vext.dat*

This file contains the neutral part of the external field acting on each species as a function of position on the mesh.

7.2.6 *dft_vext_c.dat*

This file contains the Coulombic part of the external field acting on each species as a function of position on the mesh.

7.2.7 *dft_zeroTF.dat*

This file contains an array of 0s and 1s (False and True respectively) as a function of position on the mesh. This array shows where the molecular theory density functional equations are replaced with the condition density=0 (wherever this array is True).

7.2.8 *dft_zones.dat*

This file contains the zone assignment of every node on the mesh.

7.2.9 *proc_mesh.dat*

This file contains data on the node to processor map obtained by the load balancing process.

7.2.10 *Resid2.dat*

This file should contain information on the residuals (not currently working).

7.2.11 *rho_init.dat*

This file echoes the initial guess used for a given run.

7.2.12 *rho_init.datg*

For CMS polymers, this file echoes the initial guess for the propagator G functions used for a given run.

7.2.13 *stencil.out*

This file contains the integration stencils used for residual and jacobian calculations. The columns are an index, then the Ndim offsets, then the weight.

References

[1] Y. Rosenfeld. Free-energy model for the inhomogeneous hard-sphere fluid mixture and density-functional theory of freezing. *Phys. Rev. Lett.*, 63:980, 1989.

[2] Y. Rosenfeld, M. Schmidt, H. Löwen, and P. Tarazona. Fundamental-measure free-energy density functional for hard spheres: Dimensional crossover and freezing. *Phys. Rev. E*, 55:4245, 1997.

[3] Y. Rosenfeld, M. Schmidt, H. Löwen, and P. Tarazona. Dimensional crossover and the freezing transition in density functional theory. *J. Phys.: Condens. Matter*, 8:L577, 1996.

[4] R. Roth, R. Evans, A. Lang, and G. Kahl. Fundamental measure theory for hard-sphere mixtures revisited: the white bear version. *J. Phys. Condens. Matter*, 14:12063, 2002.

[5] D. Chandler, J. D. McCoy, and S. J. Singer. *J. Chem. Phys.*, 85:5971, 1986.

[6] D. Chandler, J. D. McCoy, and S. J. Singer. *J. Chem. Phys.*, 85:5977, 1986.

[7] J. D. McCoy, S. J. Singer, and D. Chandler. *J. Chem. Phys.*, 87:4853, 1987.

[8] J. P. Donley, J. J. Rajasekaran, J. D. McCoy, and J. D. Curro. Microscopic approach to inhomogeneous polymeric liquids. *J. Chem. Phys.*, 103(12):5061, 1995.

[9] A. L. Frischknecht, J. D. Weinhold, A. G. Salinger, J. G. Curro, L. J. D. Frink, and J. D. McCoy. Density functional theory for inhomogeneous polymer systems. i. numerical methods. *J. Chem. Phys.*, 117:10385–10397, 2002.

[10] S. Tripathi and W. G. Chapman. Microstructure and thermodynamics of inhomogeneous polymer blends and solutions. *Phys. Rev. Lett.*, 94:087801, 2005.

[11] S. Tripathi and W. G. Chapman. Microstructure of inhomogeneous polyatomic mixtures from a density functonal formalism for atomic mixtures. *J. Chem. Phys.*, 122:094506, 2005.

[12] M. A. Heroux, L. J. D. Frink, and A. G. Salinger. Schur complement based approach to solving density functional theories for inhomogeneous fluids on parallel computers. *submitted to SIAM J. Sci.Comput.*, 2006.

[13] A. G. Salinger and L. J. D. Frink. Rapid analysis of phase behavior with density functional theory. i. novel numerical methods. *J. Chem. Phys.*, 118:7457–65., 2003.