FORTRAN 90 Ornztein-Zernike solver — version 1.7.1

Patrick B Warren, Unilever R&D Port Sunlight (July 2016)

 If you end up publishing results based on this, please cite

> P. B. Warren, A. Vlasov, L. Anton and A. J. Masters "Screening properties of Gaussian electrolyte models, with application to dissipative particle dynamics", J. Chem. Phys. **138**, 204907 (2013).

Features of version 1.7:

- FORTRAN 90 based, with example python driver scripts ;
- fast Ng solver, and Ng decomposition for electrostatics ;
- multicomponent (arbitrary number of components) ;
- hard and soft core (DPD) potentials ;
- full structural thermodynamics ;
- fully open source.

Added in version 1.7.1:

- exact and approximate pair potentials for Slater smearing.

Contents

1 Introduction

The FORTRAN 90 module `oz_mod.f90` implements an Ornstein-Zernike solver for multicomponent mixtures of possibly charged particles, returning both structural and thermodynamic information. The closures currently implemented are the RPA, MSA, EXP, and HNC. Though we say it ourselves, the code is *fast* since it is written in native FORTRAN and implements the Ng acceleration schemes [2]. The interaction potentials are specified in their own routines and the code could be used for other potentials with minor modifications. Further technical background to liquid state integral equation methods can be found in Hansen and McDonald [3], Kelley and Montgomery Pettitt [4], and Vrbka *et al.* [5]. We note that Vrbka provides an independent package called 'pyOZ', implemented entirely in `python` [5]. There are some differences with the present approach: the Ornstein-Zernike relation is normalised slightly differently, a conjugate gradient method is used to accelerate convergence, and pyOZ provides other closures in addition to the ones provided here.

The present code is based on an HNC code developed by Lucian Anton, modified for dissipative particle dynamics (DPD) by Andrey Vlasov with help from Lucian and Andrew Masters. The code was later converted by Patrick Warren to use the open source libraries FFTW and LAPACK, and rewritten to be compatible with the FORTRAN-`python` interface generator `f2py`.

2 Theoretical background

2.1 Single component HNC

We suppose that the interaction between a pair of particles is given by the potential $v(r)$. We introduce the following quantities [3]: the pair distribution function $g(r)$, the total correlation function $h(r) \equiv g(r) - 1$, the direct correlation function $c(r)$ defined by the Ornstein-Zernike relation below, and the indirect correlation function $e(r) \equiv h(r) - c(r)$. Note that the indirect correlation function is called $\gamma(r)$ by Vrbka *et al.* [5] and $b(r)$ by Hansen and McDonald [3]. We also introduce the Fourier transforms, *e. g.*

$$\tilde{h}(k) = \int d^3\mathbf{r}\, e^{-i\mathbf{k}\cdot\mathbf{r}}\, h(r)\,, \tag{1}$$

which simplifies to the Fourier-Bessel transform

$$\tilde{h}(k) = \frac{4\pi}{k} \int_0^\infty dr\, \sin(kr)\, r\, h(r)\,, \tag{2}$$

and

$$h(r) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\, e^{i\mathbf{k}\cdot\mathbf{r}}\, \tilde{h}(k)\,, \tag{3}$$

which simplifies to

$$h(r) = \frac{1}{2\pi^2 r} \int_0^\infty dk\, \sin(kr)\, k\, \tilde{h}(k)\,. \tag{4}$$

In terms of the Fourier transforms the Ornstein-Zernike (OZ) relation is

$$\tilde{h}(k) = \tilde{c}(k) + \rho\, \tilde{h}(k)\, \tilde{c}(k) \tag{5}$$

where ρ is the density. This confirms the relevance of the standard choice of normalisation of the 3d Fourier transform pair which puts the factor $1/(2\pi)^3$ into the back transform. The OZ relation can be rearranged to

$$\tilde{h}(k) = \frac{\tilde{c}(k)}{1 - \rho\tilde{c}(k)} \tag{6}$$

(*c. f.* Eq. (3.5.13) in Ref. [3]). The hyper-netted chain (HNC) closure is defined in real space and is

$$g(r) = \exp[-\beta v(r) + h(r) - c(r)] \tag{7}$$

(*c. f.* Eq. (4.3.19) in Ref. [3]) where $\beta = 1/k_{\mathrm{B}}T$. It amounts the neglect of the bridge diagrams. For hard spheres HNC is known to be inferior to Percus-Yevick, but for soft potentials like DPD it generally gives excellent results. In the presence of hard cores, we note that $\exp[-\beta v]$ is still well defined (and zero, in fact) where βv is infinite. We can accommodate hard cores therefore by making sure that we always work with $\exp[-\beta v]$ instead of βv in the numerics. The function $\exp[-\beta v]$ can be pre-computed for a given potential.

To obtain the algorithm implemented in the code we rewrite Eqs. (6) and (7) in terms of $c(r)$ and $e(r)$. The OZ relation becomes

$$\tilde{e} = \frac{\tilde{c}}{1 - \rho\tilde{c}} - \tilde{c} \tag{8}$$

and the HNC closure becomes

$$c = \exp[-\beta v + e] - e - 1\,. \tag{9}$$

The algorithm is as follows. We start with some guess for the direct correlation function $c(r)$. We Fourier transform this to $\tilde{c}(k)$ and solve the OZ

relation in Eq. (8) to get the indirect correlation function (in reciprocal space) $\tilde{e}(k)$. We Fourier back-transform to get $e(r)$ and plug this into the HNC closure in the form of Eq. (9) to get a new direct correlation function $c(r)$. This cycle is iterated until $c(r)$ and $e(r)$ converge to a self-consistent solution pair. A direct approach like this is usually numerically unstable so in the Picard scheme we mix a little of the new $c(r)$ into the old $c(r)$, and iterate until we converge to a solution. In the Ng acceleration scheme we keep track of the convergence trajectory and use this to accelerate convergence to the solution. The details of the Ng scheme will not be described here, rather we point the interested reader to Ref. [2] and the code itself. The initial trajectory for the Ng scheme is generated by a few Picard iterations. Some possible initial choices for the direct correlation function $c(r)$ are zero, $-\beta v(r)$ and $\exp[-\beta v(r)] - 1$. For soft potentials any of these will do in principle but the initial rate of convergence may differ. For hard core potentials, the middle option cannot be used.

Now we describe Ng's splitting method for handling long range forces. We partition the interaction potential into a short range part and a long range part,

$$v(r) = v'(r) + v^{\mathrm{L}}(r)\,. \tag{10}$$

It is generally accepted that $c(r) \sim -\beta v^{\mathrm{L}}(r)$ for $r \to \infty$ so we make this explicit by partitioning both the direct and indirect correlation functions,

$$c(r) = c'(r) - \beta v^{\mathrm{L}}(r)\,, \quad e(r) = e'(r) + \beta v^{\mathrm{L}}(r)\,. \tag{11}$$

These are taken to provide the definitions of $c'(r)$ and $e'(r)$. We rewrite the OZ relation and the HNC closure in terms of these,

$$\tilde{e}' = \frac{\tilde{c}' - \beta\tilde{v}^{\mathrm{L}}}{1 - \rho(\tilde{c}' - \beta\tilde{v}^{\mathrm{L}})} - \tilde{c}' \tag{12}$$

and

$$c' = \exp[-\beta v' + e'] - e' - 1\,. \tag{13}$$

The first of these requires that we know the Fourier transform of the long range part of the potential. The main advantage of the Ng split is that $c'(r)$ and $e'(r)$ are now genuinely short-ranged so the numerical behaviour is much better, certain thermodynamic integrals are guaranteed convergence, and the expected asymptotic behaviour of $c(r)$ is explicitly incorporated. The numerical solution scheme based on Eqs. (12) and (13) goes through as before. The initial choices for $c'(r)$ are replicated from above with v replaced by v'. If there is a hard core part of the potential, it should be regarded as belonging to $v'(r)$, and we make sure we always work with $\exp[-\beta v'(r)]$.

2.2 Multicomponent HNC

Now we turn to the multicomponent problem. The pair functions become for example $g_{\mu\nu}(r)$, *et c.*, and the problem is specified by the interaction potential $v_{\mu\nu}(r)$ and the species densities ρ_μ. The total density is $\rho = \sum_\mu \rho_\mu$ and the species mole fractions are $x_\mu = \rho_\mu/\rho$. We again split the potential into short and long range contributions,

$$v_{\mu\nu}(r) = v'_{\mu\nu}(r) + v^{\rm L}_{\mu\nu}(r) \tag{14}$$

where typically $v^{\rm L}_{\mu\nu}(r)$ incorporates the long range part of the charge interactions. Often the long range part satisfies a symmetry condition (SYM) where

$$v^{\rm L}_{\mu\nu}(r) = z_\mu z_\nu v^{\rm L}(r) \quad \text{(SYM)} \tag{15}$$

in which z_μ is the valency and $v^{\rm L}(r)$ is the interaction potential between unit charges of the same sign. However we will not assume that $v^{\rm L}_{\mu\nu}(r)$ necessarily meets the SYM condition. As above we define $c'_{\mu\nu} = c_{\mu\nu} + \beta v^{\rm L}_{\mu\nu}$ and $e'_{\mu\nu} = e_{\mu\nu} - \beta v^{\rm L}_{\mu\nu}$. The multicomponent HNC closure becomes

$$c'_{\mu\nu} = \exp[-\beta v'_{\mu\nu} + e'_{\mu\nu}] - e'_{\mu\nu} - 1\,. \tag{16}$$

To derive the OZ relation in a usable form we start from the multicomponent analogue of Eq. (5),

$$\tilde{h}_{\mu\nu} = \tilde{c}_{\mu\nu} + \rho \textstyle\sum_\lambda x_\lambda\, \tilde{c}_{\mu\lambda}\, \tilde{h}_{\lambda\nu}\,. \tag{17}$$

We write this as a matrix relation, introducing R as a diagonal matrix with entries $\rho_\mu = \rho x_\mu$,

$$\tilde{H} = \tilde{C} + \tilde{C}\cdot R\cdot \tilde{H}\,. \tag{18}$$

Note that all the matrices in this are symmetric, so we can equally well write this as

$$\tilde{H} = \tilde{C} + \tilde{H}\cdot R\cdot \tilde{C}\,. \tag{19}$$

Introducing next $\tilde{C}' = \tilde{C} + \beta\tilde{U}^{\rm L}$ and $\tilde{E}' = \tilde{E} - \beta\tilde{U}^{\rm L}$ in Eq. (18), and rearranging, gives eventually

$$\tilde{E}' = [I - (\tilde{C}' - \beta\tilde{U}^{\rm L})\cdot R]^{-1}\cdot[(\tilde{C}' - \beta\tilde{U}^{\rm L})\cdot R\cdot\tilde{C}' - \beta\tilde{U}^{\rm L}]\,. \tag{20}$$

The solution scheme is essentially as before. Given an initial guess for $c'_{\mu\nu}(r)$ we Fourier transform, evaluate the matrix expression in Eq. (20), and Fourier back-transform to obtain $e'_{\mu\nu}(r)$. We substitute this into the HNC closure in Eq. (16) to obtain a new guess for $c'_{\mu\nu}(r)$. The cycle is iterated to convergence and in practice a few Picard iterations are used to pump-prime a

multicomponent version of the Ng acceleration scheme. As before, any hard core part of the potential should be regarded as belonging to $\beta v'_{\mu\nu}$ and the numerical scheme should be constructed to use only $\exp[-\beta v'_{\mu\nu}]$ (which can be pre-computed).

Given a converged solution pair $(c'_{\mu\nu}, e'_{\mu\nu})$, the total correlation functions can be evaluated from

$$h_{\mu\nu}(r) = c'_{\mu\nu}(r) + e'_{\mu\nu}(r) \tag{21}$$

(note that $\beta v^{\rm L}_{\mu\nu}$ cancels). The pair functions are given by $g_{\mu\nu} = \delta_{\mu\nu} + h_{\mu\nu}$.

For the partial structure factors, there is a choice of normalisation. In the present code the structure factors are perhaps unusually defined to be

$$S_{\mu\nu}(k) = \rho_\mu \delta_{\mu\nu} + \rho_\mu \rho_\nu \tilde{h}_{\mu\nu} \tag{22}$$

where $\tilde{h}_{\mu\nu} = \tilde{c}'_{\mu\nu} + \tilde{e}'_{\mu\nu}$ (again, $\beta v^{\rm L}_{\mu\nu}$ cancels); the Fourier transforms $\tilde{c}'_{\mu\nu}$ and $\tilde{e}'_{\mu\nu}$ are available as a byproduct of solving the OZ relation. The unusual choice of normalisation is made to facilitate the calculation of the charge-charge and density-density structure factors. With this choice the structure factors obey $S_{\mu\nu}(k) \to \rho_\mu \delta_{\mu\nu}$ as $k \to \infty$. An alternative (and more popular) normalisation, for which $S_{\mu\nu}(k) \to \delta_{\mu\nu}$ as $k \to \infty$, is obtained from the above expression by multiplying by $(\rho_\mu \rho_\nu)^{-1/2}$.

2.3 Multicomponent RPA

The RPA closure of the Ornstein-Zernike equations is $c_{\mu\nu}(r) = -\beta v_{\mu\nu}(r)$. This is in fact one of the choices for initialising the HNC solver. Given the HNC machinery, the implementation of the RPA is almost completely trivial and comprises a single round trip through the OZ solver starting from $c'_{\mu\nu}(r) = -\beta v'_{\mu\nu}(r)$. The RPA does not make sense for hard core potentials and instead the full MSA should be used (see below).

2.4 Multicomponent EXP

The EXP approximation is a straightforward development of the RPA in which the real space total correlation functions are replaced by

$$h_{\mu\nu}(r) \to \exp[h_{\mu\nu}(r)] - 1\,. \tag{23}$$

This breaks unwarranted symmetries such as $h_{++} = -h_{+-}$ and ensures that $h_{\mu\nu}(r) > -1$ which is not always satisfied by the RPA. In practice EXP can be nearly as good as HNC and extends to state points where HNC

doesn't have a solution. Since the EXP approximation is defined as an action on the total correlation functions in real space, a round trip through the OZ relation is required to compute the corresponding direct and indirect correlation functions. To do this we rewrite the OZ relation in Eq. (19) as

$$\tilde{C}' = (I + \tilde{H} \cdot R)^{-1} \cdot \tilde{H} + \beta \tilde{U}^{\mathrm{L}} \,. \tag{24}$$

Once this has been implemented, $e'_{\mu\nu} = h_{\mu\nu} - c'_{\mu\nu}$ follows by subtraction. Since it depends on the RPA, the EXP approximation in this form cannot be used for hard core potentials.

2.5 Multicomponent MSA

The mean spherical approximation (MSA) differs from the RPA only in the presence of hard cores. In this case we require $g_{\mu\nu}(r) = 0$ for $r \leq d_{\mu\nu}$. Together with the RPA-like $c_{\mu\nu}(r) = -\beta v_{\mu\nu}(r)$ for $r > d_{\mu\nu}$, this gives the MSA closure. Here, $d_{\mu\nu}$ is the range of the hard core repulsion between species μ and ν (*i. e.* $v'_{\mu\nu} = \infty$ for $r \leq d_{\mu\nu}$). Rewriting as usual in terms of our offset functions $v'_{\mu\nu} = v_{\mu\nu} - v^{\mathrm{L}}_{\mu\nu}$, $c'_{\mu\nu} = c_{\mu\nu} + \beta v^{\mathrm{L}}_{\mu\nu}$ and $e'_{\mu\nu} = e_{\mu\nu} - \beta v^{\mathrm{L}}_{\mu\nu}$, where $h_{\mu\nu} = g_{\mu\nu} - 1$ and $e_{\mu\nu} = h_{\mu\nu} - c_{\mu\nu}$, the MSA closure is

$$c'_{\mu\nu} = \begin{cases} -e'_{\mu\nu} - 1 & (r < d_{\mu\nu}) \,, \\ -\beta v'_{\mu\nu} & (r > d_{\mu\nu}) \,. \end{cases} \tag{25}$$

This is almost a drop-in replacement for the HNC closure in Eq. (16). For a cold start we can initialise $c'_{\mu\nu} = -1$ for $r \leq d_{\mu\nu}$. In practice we can leave $c'_{\mu\nu} = -\beta v'_{\mu\nu}$ untouched for $r > d_{\mu\nu}$ during the iterative solution. However, we make sure $c'_{\mu\nu}$ is correctly reset for $r > d_{\mu\nu}$ whenever the potential is changed.

2.6 Thermodynamics

The above section completely specifies the HNC closure for the integral equation problem for multicomponent system. We provide here the suite of thermodynamic quantities which can be evaluated from the converged solution pair $(c'_{\mu\nu}, e'_{\mu\nu})$ (or indeed for the RPA or EXP solutions). Some care is needed in the case of hard core potentials, where $v_{\mu\nu} = \infty$ for $r < d_{\mu\nu}$.

The first thermodynamic quantity is the so-called virial route compressibility factor

$$\frac{\beta p}{\rho} = 1 - \frac{2\pi}{3} \sum_{\mu\nu} \rho x_\mu x_\nu \int_0^\infty dr\, r^3 \, \frac{\partial(\beta v_{\mu\nu})}{\partial r} g_{\mu\nu}(r) \tag{26}$$

(*c.f.* Eq. (1.2) in Ref. [5]). To handle the hard core contribution to this we follow the line of argument presented in Hansen and McDonald [3] and introduce the function $y_{\mu\nu} = \exp[\beta v_{\mu\nu}]\, g_{\mu\nu}$, which is continuous through into the hard core region. The above integral can be written

$$\begin{aligned} I_{\mu\nu} &= \int_0^\infty dr\, r^3\, \frac{\partial(\beta v_{\mu\nu})}{\partial r} \exp[-\beta v_{\mu\nu}]\, y_{\mu\nu}(r) \\ &= -\int_0^\infty dr\, r^3\, \frac{\partial(\exp[-\beta v_{\mu\nu}])}{\partial r}\, y_{\mu\nu}(r)\,. \end{aligned} \tag{27}$$

Let us write

$$\exp[-\beta v_{\mu\nu}] = \Theta(r - d_{\mu\nu})\, \exp[-\beta \overline{v}_{\mu\nu}] \tag{28}$$

where $\overline{v}_{\mu\nu} = v_{\mu\nu}$ for $r \geq d_{\mu\nu}$, and we don't care what it does for $r < d_{\mu\nu}$ except that it should be *continuous* through $r = d_{\mu\nu}$. The Heaviside step function $\Theta(x) = 0$ for $x < 0$ and $\Theta(x) = 1$ for $x \geq 0$, with derivative $\Theta'(x) = \delta(x)$. Then

$$\begin{aligned} I_{\mu\nu} &= -\int_0^\infty dr\, r^3 \Big(\delta(r - d_{\mu\nu}) \exp[-\beta \overline{v}_{\mu\nu}]) \\ &\qquad + \Theta(r - d_{\mu\nu}) \frac{\partial(\exp[-\beta \overline{v}_{\mu\nu}])}{\partial r} \Big)\, y_{\mu\nu}(r) \\ &= -g_{\mu\nu}(d_{\mu\nu})\, d_{\mu\nu}^3 + \int_{d_{\mu\nu}}^\infty dr\, r^3\, \frac{\partial(\beta v_{\mu\nu})}{\partial r} g_{\mu\nu}(r) \end{aligned} \tag{29}$$

In the second line we have restored $v_{\mu\nu}$ and $g_{\mu\nu}$. Thus, finally,

$$\frac{\beta p}{\rho} = 1 + \frac{2\pi}{3} \sum_{\mu\nu} \rho x_\mu x_\nu \Big[g_{\mu\nu}(d_{\mu\nu})\, d_{\mu\nu}^3 - \int_{d_{\mu\nu}}^\infty dr\, r^3\, \frac{\partial(\beta v_{\mu\nu})}{\partial r} g_{\mu\nu}(r) \Big] \tag{30}$$

This now contains explicitly the famous contact contribution to the pressure, and the integral is now restricted to the region external to the hard core. In the case of a soft potential (DPD, URPM), the contact contribution vanishes and the integral extends to $r = 0$.

In practice we write the integral contribution to the virial pressure as

$$-\frac{2\pi}{3} \int_{d_{\mu\nu}}^\infty dr\, r^3\, \frac{\partial(\sum_{\mu\nu} \rho x_\mu x_\nu\, \beta v_{\mu\nu})}{\partial r} + \sum_{\mu\nu} \rho x_\mu x_\nu t_{\mu\nu}^{\mathrm{V}} \tag{31}$$

where

$$t_{\mu\nu}^{\mathrm{V}} = -\frac{2\pi}{3} \int_{d_{\mu\nu}}^\infty dr\, r^3\, \frac{\partial(\beta v'_{\mu\nu} + \beta v_{\mu\nu}^{\mathrm{L}})}{\partial r}\, (c'_{\mu\nu} + e'_{\mu\nu}) \tag{32}$$

The first term in Eq. (31) is the result for $g_{\mu\nu} = 1$ and is the mean-field contribution. For many applications it is important to evaluate this analytically, as the individual contributions may diverge. Under the SYM condition the contribution from $v^{\rm L}_{\mu\nu}$ to the mean field term vanishes since charge neutrality implies $\sum_{\mu\nu} \rho x_\mu x_\nu v^{\rm L}_{\mu\nu} \propto v^{\rm L}(\sum_\mu \rho_\mu z_\mu)^2 = 0$. The second term in Eq. (31) is the correlation correction (note $c'_{\mu\nu} + e'_{\mu\nu} = g_{\mu\nu} - 1$).

Next up is the compressibility itself which we write as

$$\frac{\partial(\beta p)}{\partial \rho} = 1 - \textstyle\sum_{\mu\nu} \rho x_\mu x_\nu t^{\rm C}_{\mu\nu} \tag{33}$$

where

$$t^{\rm C}_{\mu\nu} = 4\pi \int_0^\infty dr\, r^2 \, (c'_{\mu\nu}(r) - \beta v^{\rm L}_{\mu\nu}) \tag{34}$$

(*c. f.* Eq. (18) in Ref. [5]; the *compressibility* is not to be confused with the above *compressibility factor*), In terms of this, the isothermal compressibility is $\chi_T = [\rho(\partial p/\partial \rho)]^{-1}$. The contribution from $v^{\rm L}_{\mu\nu}$ can also often be evaluated analytically and vanishes under the SYM condition by the same argument as above. Obviously the compressibility can be integrated along an isotherm to obtain an alternative expression for the pressure. This is the so-called compressibility route to the equation of state. This expression is insensitive to the presence of hard cores since they are supposed to be contained in $v'_{\mu\nu}$ and enter *via* the direct correlation function.

The next quantity of interest is the internal energy U. Per particle, the excess internal energy is

$$\frac{U^{\rm ex}}{N} = 2\pi \textstyle\sum_{\mu\nu} \rho x_\mu x_\nu \displaystyle\int_{d_{\mu\nu}}^\infty dr\, r^2 \, v_{\mu\nu}(r)\, g_{\mu\nu}(r) \tag{35}$$

(*c. f.* Eq. (2.5.20) in Ref. [3]). Again this can be split into a mean field term and a correlation correction,

$$\frac{\beta U^{\rm ex}}{N} = 2\pi \int_{d_{\mu\nu}}^\infty dr\, r^2 \, (\textstyle\sum_{\mu\nu} \rho x_\mu x_\nu \, \beta v_{\mu\nu}) + \sum_{\mu\nu} \rho x_\mu x_\nu t^{\rm E}_{\mu\nu} \tag{36}$$

where

$$t^{\rm E}_{\mu\nu} = 2\pi \int_{d_{\mu\nu}}^\infty dr\, r^2 \, (\beta v'_{\mu\nu} + \beta v^{\rm L}_{\mu\nu}) \, (c'_{\mu\nu} + e'_{\mu\nu}) \, . \tag{37}$$

An integration by parts shows that the mean field term here is related to that for the compressibility factor,

$$2\pi \int_{d_{\mu\nu}}^\infty dr\, r^2 \, \beta v_{\mu\nu} = -\frac{2\pi \beta v_{\mu\nu}(d_{\mu\nu})\, d^3_{\mu\nu}}{3} - \frac{2\pi}{3} \int_{d_{\mu\nu}}^\infty dr\, r^3 \, \frac{\partial(\beta v_{\mu\nu})}{\partial r} \, . \tag{38}$$

The excess internal energy density (per unit volume) is $U^{\rm ex}/V = \rho \times (U^{\rm ex}/N)$. The equation of state can be derived from the internal energy and this is the so-called energy route.

Last, and **only valid for the HNC closure**, are thermodynamic integrals for excess chemical potentials. The relevant result is

$$\beta\mu_\mu^{\rm ex} = 4\pi \sum_\nu \rho x_\nu \int_0^\infty dr\, r^2 \left[\tfrac{1}{2}\, h_{\mu\nu}(r)\, e_{\mu\nu}(r) - c_{\mu\nu}(r)\right] \quad \left(\equiv \ln \gamma_\mu \right) \tag{39}$$

(*c. f.* section II C in Ref. [5]). We write this result as $\beta\mu_\mu^{\rm ex} = \sum_\nu \rho x_\nu t_{\mu\nu}^{\rm M}$ where

$$t_{\mu\nu}^{\rm M} = 4\pi \int_0^\infty dr\, r^2 \left[\tfrac{1}{2}\, (c'_{\mu\nu} + e'_{\mu\nu})(e'_{\mu\nu} + \beta v_{\mu\nu}^{\rm L}) - c'_{\mu\nu} + \beta v_{\mu\nu}^{\rm L}\right]. \tag{40}$$

The contribution from $v_{\mu\nu}^{\rm L}$ (the last term) can usually be evaluated analytically and is the same as that appearing in the compressibility (to within an overall sign). These expressions, valid only for HNC, offer a fourth route to the equation of state (the chemical-potential route). These expressions are insensitive to the presence of hard cores.

In HNC both the virial route pressure and the chemical potentials correspond to a certain free energy. Since they are compatible in this way, the excess free energy density follows from

$$\beta F^{\rm ex}/V = \beta \sum_\mu \rho x_\mu \mu_\mu^{\rm ex} - \beta p + \rho \tag{41}$$

where the pressure is calculated from the virial route compressibility factor.

3 Potentials

3.1 DPD potential

The HNC code contains routines which specify a multicomponent potential for dissipative particle dynamics (DPD) with Gaussian charges. The potential in this case consists of short range and long range contributions which can be mapped exactly to the Ng split. The short range part is

$$v'_{\mu\nu}(r) = \begin{cases} \frac{1}{2} A_{\mu\nu} k_{\rm B} T (1 - r/r_c)^2 & r < r_c \\ 0 & r \geq r_c \end{cases}. \tag{42}$$

This is a conventional choice, depending on a dimensionless symmetric repulsion amplitude matrix $A_{\mu\nu}$ and cut off beyond a distance r_c.

Unlike the short range part there is no consensus on the best choice for the long range interaction although the differences can always be adsorbed

into a redefinition of the short range potential. The first choice of Gaussian charges can be supported for a range of practical reasons [1, 6]. The Gaussian charges have size $\sim \sigma$ and a density distribution $(2\pi\sigma^2)^{-3/2}\exp(-r^2/2\sigma^2)$. The corresponding interaction satisfies the SYM condition and is given by

$$v^{\rm L}_{\mu\nu}(r) = z_\mu z_\nu v^{\rm L}(r)\,, \quad v^{\rm L}(r) = \frac{l_{\rm B}k_{\rm B}T}{r}\,{\rm erf}\Big(\frac{r}{2\sigma}\Big) \tag{43}$$

where $l_{\rm B}$ is the coupling strength (the Bjerrum length). The precise definition of σ is made to simplify the Fourier transform (see below).

For use with the above expressions, we need the derivative of both parts of the potential, and the Fourier transform of the long range part. These are

$$\frac{\partial(\beta v'_{\mu\nu})}{\partial r} = \begin{cases} -(A_{\mu\nu}/r_c)(1-r/r_c) & r<r_c \\ 0 & r\geq r_c \end{cases} \tag{44}$$

$$\frac{\partial(\beta v^{\rm L})}{\partial r} = -\frac{l_{\rm B}}{r^2}\,{\rm erf}\Big(\frac{r}{2\sigma}\Big) + \frac{l_{\rm B}}{r\sigma\sqrt{\pi}}\,e^{-r^2/4\sigma^2}\,, \tag{45}$$

and

$$\beta\tilde{v}^{\rm L}(k) = \frac{4\pi l_{\rm B}}{k^2}\times e^{-k^2\sigma^2}\,. \tag{46}$$

The mean field contributions to the virial route pressure and energy are given by (see also Eq. (38))

$$2\pi\int_0^\infty dr\, r^2\,(\textstyle\sum_{\mu\nu}\rho x_\mu x_\nu\,\beta v_{\mu\nu}) = \frac{\pi r_c^3}{30}\textstyle\sum_{\mu\nu}\rho x_\mu x_\nu A_{\mu\nu}\,. \tag{47}$$

Because of the SYM condition, the contribution from $v^{\rm L}$ vanishes from this, and also vanishes from the mean field contributions to the compressibility and chemical potentials (this is true for all the cases in this section).

This DPD model depends on the dimensionless repulsion amplitude matrix $A_{\mu\nu}$, the ratios $l_{\rm B}/\sigma$ and σ/r_c, and the dimensionless density ρr_c^3 [1]. In these terms $r_c = 1$ is often used as the fundamental length scale. The ratio $l_{\rm B}/\sigma$ plays the role of an inverse effective temperature for the Coloumbic part of the potential.

The above is the Gaussian charge case and is the default. If Bessel charges are selected the charge distribution becomes $K_1(r/\sigma)/2\pi^2\sigma^2 r$ and the expressions change to [6]

$$\beta v^{\rm L}(r) = \frac{l_{\rm B}}{r}(1-e^{-r/\sigma})\,, \tag{48}$$

$$\frac{\partial(\beta v^{\rm L})}{\partial r} = -\frac{l_{\rm B}}{r^2}\Big[1-e^{-r/\sigma}\Big(1+\frac{r}{\sigma}\Big)\Big]\,, \tag{49}$$

$$\beta\tilde{v}^{\mathrm{L}}(k) = \frac{4\pi l_{\mathrm{B}}}{k^2} \times \frac{1}{1+k^2\sigma^2}\,. \tag{50}$$

Note that σ here is chosen to match the second moment of the charge distribution for the Gaussian case.

Another alternative is linear charge smearing proposed by Groot [7]. In this case the charge distribution is $(3/\pi R^3)(1-r/R)$ for $r < R$, vanishing for $r > R$. This gives rise to an interaction potential in reciprocal space [6]

$$\beta\tilde{v}^{\mathrm{L}}(k) = \frac{4\pi l_{\mathrm{B}}}{k^2}\Big(\frac{24-24\cos kR-12kR\sin kR}{k^4R^4}\Big)^2\,. \tag{51}$$

A closed form expression in real space is not available, hence is not implemented in the code. This means that the thermodynamics is *not available* in this case, though the HNC solution goes through since this only requires the reciprocal space potential. The term in large brackets is the charge density in reciprocal space. Matching the second moment of the charge distribution [6] shows that the equivalent Gaussian smearing length is $\sigma = R\sqrt{2/15}$.

Another popular alternative is Slater smearing proposed by González-Melchor *et al.* [8]. They use an approximate expression for the interaction potential, for which we have

$$\beta v^{\mathrm{L}}(r) = \frac{l_{\mathrm{B}}}{r}\Big[1-e^{-2\beta r}(1+\beta r)\Big]\,. \tag{52}$$

$$\frac{\partial(\beta v^{\mathrm{L}})}{\partial r} = -\frac{l_{\mathrm{B}}}{r^2}\Big[1-e^{-2\beta r}(1+2\beta r(1+\beta r))\Big]\,, \tag{53}$$

$$\beta\tilde{v}^{\mathrm{L}}(k) = \frac{4\pi l_{\mathrm{B}}}{k^2} \times \frac{1}{(1+k^2/4\beta^2)^2}\,. \tag{54}$$

These actually corresponds to a charge density $\beta^2 e^{-2\beta r}/\pi r$ (β is a parameter here, not to be confused with $1/k_{\mathrm{B}}T$). The equivalent Gaussian smearing length is here $\sigma = 1/\beta\sqrt{2}$. The intended charge density in the Slater smearing proposal is $(1/\pi\lambda^3)e^{-2r/\lambda}$. It turns out that there are exact expressions for the interaction between such charges [6], namely

$$\beta v^{\mathrm{L}}(r) = \frac{l_{\mathrm{B}}}{r}\Big[1-e^{-2r/\lambda}\Big(1+\frac{11r}{8\lambda}+\frac{3r^2}{4\lambda^2}+\frac{r^3}{6\lambda^3}\Big)\Big]\,, \tag{55}$$

$$\frac{\partial(\beta v^{\mathrm{L}})}{\partial r} = -\frac{l_{\mathrm{B}}}{r^2}\Big[1-e^{-2r/\lambda}\Big(1+\frac{2r}{\lambda}+\frac{2r^2}{\lambda^2}+\frac{7r^3}{6\lambda^3}+\frac{r^4}{3\lambda^4}\Big)\Big]\,, \tag{56}$$

$$\beta\tilde{v}^{\mathrm{L}}(k) = \frac{4\pi l_{\mathrm{B}}}{k^2} \times \frac{1}{(1+k^2\lambda^2/4)^4}\,. \tag{57}$$

The equivalent Gaussian smearing length for these exact expressions is $\sigma = \lambda$. Originally, González-Melchor *et al.* [8] had $\beta = 1/\lambda$ in Eq. (52), but choosing $\beta = 5/(8\lambda)$ makes Eqs. (52) and (55) have the same limiting value as $r \to 0$ and brings the approximate interaction much closer to the exact result. Note that there is a natural hierarchy going from Bessel ($n = 1$), to approximate Slater ($n = 2$), to exact Slater ($n = 4$), to Gaussian ($n \to \infty$), in terms of the reciprocal space potential $\beta\tilde{v}^{\mathrm{L}}(k) = 4\pi l_{\mathrm{B}} k^{-2}(1 + \sigma^2 k^2/n)^{-n}$.

3.2 Softened URPM potential

Another potential provided by the code at present is for the URPM in which the unlike pair potential is artificially softened. The standard URPM (an equimolar mixture of Gaussian charges) is given by the DPD potential above with $z_\mu = \pm 1$ and $A_{\mu\nu} = 0$. There are two ways the softened version can be implemented. The first way is to work purely in reciprocal space. It is important to note that this takes the model away from the SYM condition expressed in Eq. (15). In this approach the short range part $v'_{\mu\nu} = 0$, and the long range part is

$$v_{11}^{\mathrm{L}} = v_{22}^{\mathrm{L}} = \frac{l_{\mathrm{B}} k_{\mathrm{B}} T}{r} \operatorname{erf}\Big(\frac{r}{2\sigma}\Big) \quad v_{12}^{\mathrm{L}} = -\frac{l_{\mathrm{B}} k_{\mathrm{B}} T}{r} \operatorname{erf}\Big(\frac{r}{2\sigma'}\Big) \tag{58}$$

where typically $\sigma' > \sigma$ (in the case $\sigma' = \sigma$ we have the original URPM which is also contained in the DPD potential described above). The potential in reciprocal space and the derivatives follow *mutatis mutandis* from the DPD case. In the alternative approach we retain the SYM condition so that the long range part is given by Eq. (43) and the short range part is

$$v'_{12} \equiv -\Delta v_{12} = -\frac{l_{\mathrm{B}} k_{\mathrm{B}} T}{r} \operatorname{erf}\Big(\frac{r}{2\sigma'}\Big) + \frac{l_{\mathrm{B}} k_{\mathrm{B}} T}{r} \operatorname{erf}\Big(\frac{r}{2\sigma}\Big) \tag{59}$$

(obviously, $v'_{11} = v'_{22} = 0$). We define Δv_{12} as the potential that should be *added* to the softened URPM, to recover the standard URPM. This is so that we can use the softened version as a reference fluid.

For both routes the mean field contributions to the virial route pressure and energy are non-zero only for the unlike pairs. We can evaluate them with v'_{12} given by Eq. (59)

$$-\frac{2\pi}{3} \int_0^\infty dr\, r^3 \, \frac{\partial(\beta v'_{12})}{\partial r} = 2\pi \int_0^\infty dr\, r^2 \, \beta v'_{12} = 2\pi l_{\mathrm{B}} (\sigma'^2 - \sigma^2)\,. \tag{60}$$

If we take the first, purely reciprocal space route, the failure to satisfy the SYM condition means the compressibility and chemical potentials need to

take account of $v^{\rm L}_{\mu\nu}$. This leads to the long range contributions

$$4\pi \int_0^\infty dr\, r^2\, \beta v'_{12} = 4\pi l_{\rm B}(\sigma'^2 - \sigma^2)\,. \tag{61}$$

These should not be included if we follow the second route.

Finally if we use the softened URPM as a reference fluid in the Wertheim approach we need to evaluate the integral

$$\Delta_{12} = 4\pi \int_0^\infty dr\, r^2\, g_{12}(r)\, [\exp(-\beta\Delta v_{12}) - 1] \tag{62}$$

where the pair distribution function is that of the reference fluid. A related integral is in the first order perturbation theory $F = F_0 + \langle \Delta U \rangle_0$ where

$$\frac{\beta\langle \Delta U\rangle_0}{V} = \frac{\pi\rho^2}{2} \int_0^\infty dr\, r^2\, g_{12}(r)\, \beta\Delta v_{12}\,. \tag{63}$$

As of version 1.7 these integrals have been moved out of the FORTRAN module and into the `python` scripts.

3.3 Softened RPM potential

Another potential provided by the code at present is for the RPM in which the unlike pair potential is artificially softened. The standard RPM is an equimolar mixture of charged hard spheres. The interaction potential $v'_{\mu\nu}(r) = \infty$ for $r < \sigma$ where σ is the hard core diameter. This is regarded as belonging to the short range interaction potential, for which we are always careful to use the exponentiated version. Thus hard cores are actually implemented by calculating $\exp[-\beta v'_{\mu\nu}]$ using one of the choices below, and setting this to zero for $r < \sigma$. The valencies are $z_\mu = \pm 1$.

As in the softened URPM, there are two ways the softened RPM can be implemented. The first way is to work purely in reciprocal space which takes the model away from the SYM condition expressed in Eq. (15). In this approach the short range part $v'_{\mu\nu} = 0$ (outside the hard cores), and the long range part is

$$v^{\rm L}_{11} = v^{\rm L}_{22} = \frac{l_{\rm B}k_{\rm B}T}{r} \quad v^{\rm L}_{12} = -\frac{l_{\rm B}k_{\rm B}T}{r}\,{\rm erf}(\kappa r)\,. \tag{64}$$

The potential in reciprocal space and the derivatives follow *mutatis mutandis* from the softened URPM case (*i. e.* URPM $\sigma \to 0$ and $\sigma' \to 1/(2\kappa)$). We note that the pure RPM case obtains in the limit $\kappa \to \infty$.

In the alternative approach we retain the SYM condition so that the long range part is given by

$$v^{\mathrm{L}}_{\mu\nu} = z_\mu z_\nu \frac{l_{\mathrm{B}} k_{\mathrm{B}} T}{r} \tag{65}$$

and the short range part is

$$v'_{12} \equiv -\Delta v_{12} = \frac{l_{\mathrm{B}} k_{\mathrm{B}} T}{r} \operatorname{erfc}(\kappa r) \tag{66}$$

(obviously, $v'_{11} = v'_{22} = 0$). As in the URPM, we define Δv_{12} as the potential that should be *added* to the softened RPM, to recover the standard RPM. This is so that we can use the softened version as a reference fluid.

For both routes the mean field contributions to the virial route pressure and energy are non-zero only for the unlike pairs. We can evaluate them with v'_{12} given by Eq. (59)

$$-\frac{2\pi}{3} \int_\sigma^\infty dr\, r^3\, \frac{\partial(\beta v'_{12})}{\partial r} = \pi l_{\mathrm{B}} \Big(\frac{\sigma \exp[-\kappa^2\sigma^2]}{\kappa\sqrt{\pi}} + \Big(\frac{1}{2\kappa^2} - \frac{\sigma^2}{3} \Big) \operatorname{erfc}(\kappa\sigma) \Big) . \tag{67}$$

$$2\pi \int_\sigma^\infty dr\, r^3\, \beta v'_{12} = \pi l_{\mathrm{B}} \Big(\frac{\sigma \exp[-\kappa^2\sigma^2]}{\kappa\sqrt{\pi}} + \Big(\frac{1}{2\kappa^2} - \sigma^2 \Big) \operatorname{erfc}(\kappa\sigma) \Big) . \tag{68}$$

In the limit $\sigma \to 0$ these both become $\pi l_{\mathrm{B}}/2\kappa^2$ (*c. f.* softened URPM). In the limit $\kappa \to \infty$ (RPM case) both vanish.

If we take the first, purely reciprocal space route, the failure to satisfy the SYM condition means the compressibility and chemical potentials need to take account of $v^{\mathrm{L}}_{\mu\nu}$. This leads to the long range contributions

$$4\pi \int_0^\infty dr\, r^2\, \beta v'_{12} = \frac{\pi l_{\mathrm{B}}}{\kappa^2} . \tag{69}$$

These should not be included if we follow the second route, and vanishes in any case for the RPM case ($\kappa \to \infty$).

4 Implementation notes

Most of the code is self-expanatory, given the above mathematical background. Correlation functions in the real space domain are discretised in an array of size $i = 1 \ldots n-1$ with a spacing δ_r so that $r_i = \delta_r \times i$. Usually one choses δ_r to be a small fraction of the relevant length scale (*e. g.* $\delta_r = 0.01\, r_c$) and $n\delta_r$ to be some large multiple of the same (*e. g.* $n\delta_r \approx 40\, r_c$). The actual value of n can be chosen to optimise the fast Fourier transform algorithm,

for instance $n = 4096$. The concomitant functions in the reciprocal space domain are discretised in an array of size $j = 1 \ldots n-1$ with a spacing $\delta_k \equiv \pi/(n\delta_r)$ so that $k_j = \delta_k \times j$. The array size $n-1$ and the value δ_k are fixed by the demands of the fast Fourier transform algorithm.

The implementation of the Fourier transforms is as follows. A discrete version of Eq. (2) is

$$\tilde{h}_j = \frac{2\pi\delta_r}{k_j} \times 2\sum_{i=1}^{n-1} r_i h_i \sin(\pi ij/n) \tag{70}$$

where $j = 1 \ldots n-1$. The quantity $2\sum_{i=1}^{n-1} r_i h_i \sin(\pi ij/n)$ is computed by calling the FFTW routine `RODFT00` on $r_i h_i$. From the FFTW documenation, this specific routine works best when the array length is of the form $2^a 3^b 5^c 7^d 11^e 13^f - 1$ where $e + f$ is either 0 or 1 and the other exponents are arbitrary (this means n should be of the form of the first term since the array length is $n-1$). For the back-transform the corresponding discrete version of Eq. (4) is

$$h_i = \frac{\delta_k}{(2\pi)^2 r_i} \times 2\sum_{j=1}^{n-1} k_j \tilde{h}_j \sin(\pi ij/n) \tag{71}$$

The quantity $2\sum_{j=1}^{n-1} k_j h_j \sin(\pi ij/n)$ is computed by calling `RODFT00` on $k_j h_j$.

Version 1.7 of the code can handle an arbitrary number of components (versions prior to this could only handle at most three components). A species pair array index (i, j) is mapped to a 'function' index n according to the following rule

$$n = \begin{cases} i + j(j-1)/2 & \text{if } i \le j\,, \\ j + i(i-1)/2 & \text{if } i > j\,. \end{cases} \tag{72}$$

This is implemented in various forms throughout the code. The mapping essentially labels the entries in the upper triangular form of a symmetric matrix as shown here (where i labels rows, and j labels columns)

	1	2	3	4	$\cdots$
1	1	2	4	7	$\cdots$
2		3	5	8	$\cdots$
3			6	9	$\cdots$
4				10	$\cdots$
$\cdots$					$\cdots$

(73)

If necessary, one can recover the array indices (i, j) from the the index n by the following rule, where $\lfloor \cdots \rfloor$ is the 'floor' function,

$$j = \left\lfloor \frac{1 + \sqrt{8n-7}}{2} \right\rfloor, \quad i = n - j(j-1)/2\,. \tag{74}$$

This is not used in the code though. The proof is left as an exercise.

The matrix operations in Eqs. (20) and (24) are implemented using the standard `MATMUL` function in FORTRAN 90. The matrix inversions are not implemented *per se*, but are rather done by solving the matrix equation $A \cdot X = B$ using Gauss-Jordan elimination with pivoting [9].

The Ng acceleration scheme is implemented by keeping track of a number of previous iterations (typically 6), recycling the storage. Thus the main functions $c'_{\mu\nu}$ and $e'_{\mu\nu}$ are multidimensional: the first axis is the spatial discretisation, the second axis is the species pair index, and the third axis is the Ng trajectory index.

The thermodynamic quantities in section 2.6 are all evaluated by standard trapezium rules, applied to the integrals given therein.

If scripting with `python` note that the `python` array index is one less than the FORTRAN array index. This conversion is seamlessly and invisibly implemented in `f2py`.

The solver routines require only $\beta\tilde{v}^{\mathrm{L}}_{\mu\nu}$ and $\exp[-\beta v'_{\mu\nu}]$, which are prepared in the routines which initialise the interaction potential. In the case of hard cores, to facilitate the calculation of the thermodynamic integrals with lower bound $d_{\mu\nu}$, we subsequently set $v^{\mathrm{L}}_{\mu\nu} = v'_{\mu\nu} = 0$ for $r < d_{\mu\nu}$.

5 Usage notes

The code is presented as a suite of functions in a FORTRAN 90 module. Thus it typically needs a driver code, which can be written in FORTRAN 90 or in `python` with the use of `f2py` (see examples below). Thermodynamic integration schemes, for example the compressibility route to the equation of state, are not implemented here. Rather it seems better to put these into the driver routines, as the philosophy has been to make the solution of the basic integral equation problem as fast and robust as possible.

5.1 Routines

- `initialise` – Allocate all arrays given the number of components `ncomp`, the number of real space points `ng`, and the length of the Ng trajectory `nps`. Initialise the FFTW plan for fast Fourier transforms. This `initialise` routine is typically called once, at the start, after the values of `ncomp`, `ng`, `nps`, and `deltar` have been settled.

- `dpd_potential(charge_type)` – Sets up the DPD potential described in section 3.1. This routine can be called multiple times. The parameter indicates Gaussian charges (`charge_type = 1`), Bessel charges

(`charge_type = 2`), linear (Groot) charges (`charge_type = 3`), Slater charges with exact interaction (`charge_type = 4`), or Slater charges with approximate interaction (`charge_type = 4`). The linear (Groot) case (`charge_type = 3`) is incomplete in the sense that the thermodynamics is not calculated correctly.

- `soft_urpm_potential(use_ushort)` – Sets up the softened URPM potential described in section 3.2. This routine can be called multiple times. It checks that `ncomp = 2` and forces $z_1 = 1$ and $z_2 = -1$. The parameter indicates whether the potential should be a pure long range reciprocal space part which then no longer satisfies SYM in Eq. (15) (`use_ushort = 0`), or a long range reciprocal space part which satisfies Eq. (15) plus a short range real space correction (`use_ushort = 1`).

- `soft_rpm_potential(use_ushort)` – Sets up the softened RPM potential described in section 3.3. This routine can be called multiple times. It checks that `ncomp = 2` and forces $z_1 = 1$ and $z_2 = -1$. The control parameter `use_ushort` works the same way as described for the softened URPM potential.

- `hnc_solve` – Calculate the HNC solution. Specifically, initialise $c'_{\mu\nu}$ if the flag `cold_start` is set, take enough Picard steps to pump-prime the Ng acceleration method, then attempt to converge to a solution. This is the basic HNC solver routine and contains appropriate calls to `oz_solve`, `hnc_picard`, `hnc_ng` and `conv_test` below. A warning is issued if the convergence criterion is not met. If the flag `auto_fns` is set the three `make_*` functions are also called, resulting in a complete solution to the problem. The function `hnc_solve` should be called only after `initialise` and the potential has been initialised. It may be called multiple times with different densities for a given potential, or the potential can be reset between calls. In later calls, the existing $c'_{\mu\nu}$ is used as a starting point unless the flag `cold_start` is reset.

- `rpa_solve` – Calculate the RPA solution. This involves only a single round trip through `oz_solve` from a specified start and does not require initialisation or a convergence criterion. Provided the flag `auto_fns` is set, the three `make_*` functions are also called. The function `rpa_solve` should be called only after `initialise` and the potential has been initialised. It may be called multiple times with different densities for a given potential, or the potential can be reset between calls. The RPA closure cannot be used with hard cores (see section 2.3).

- `exp_solve` – Calculate the EXP solution. This involves first solving the RPA then implementing Eqs. (23) and (24) (the latter involves a call to `oz_solve2`). Provided the flag `auto_fns` is set, the three `make_*` functions are also called. Like the RPA, the function `exp_solve` should be called only after `initialise` and the potential has been initialised. It may be called multiple times with different densities for a given potential, or the potential can be reset between calls. The EXP closure cannot be used with hard cores (see section 2.4).

- `msa_solve` – Calculate the MSA solution, using a similar iterative algorithm to the HNC closure. This routine is the basic MSA solver and contains appropriate calls to `oz_solve`, `msa_picard`, `msa_ng` and `conv_test`. A warning is issued if the convergence criterion is not met. If the flag `auto_fns` is set the three `make_*` functions are also called, resulting in a complete solution to the problem. Like the others, the function `msa_solve` should be called only after `initialise` and the potential has been initialised. It may be called multiple times with different densities for a given potential, or the potential can be reset between calls. In these subsequent calls, the existing $c'_{\mu\nu}$ within the hard core regions is used as a starting point unless the flag `cold_start` is reset. In all cases it is ensured that outside the hard core regions $c'_{\mu\nu} = -\beta v'_{\mu\nu}$. Note that the MSA is the same as the RPA if there are no hard cores (therefore, use `rpa_solve` in that case!).

- `oz_solve` – Solve the OZ relation in the form of Eq. (20) to find $e'_{\mu\nu}$ from $c'_{\mu\nu}$ (as a byproduct, this generates $\tilde{e}'_{\mu\nu}$ and $\tilde{c}'_{\mu\nu}$).

- `oz_solve2` – Solve the OZ relation in the form of Eq. (24) to find $e'_{\mu\nu}$ and $c'_{\mu\nu}$ from $h_{\mu\nu}$ (and as a byproduct generate $\tilde{e}'_{\mu\nu}$ and $\tilde{c}'_{\mu\nu}$).

- `hnc_picard` – Solve the HNC closure to find $c'_{\mu\nu}$ from $e'_{\mu\nu}$, and take a Picard step in the sense of mixing a fraction `alpha` of the new $c'_{\mu\nu}$ with the old $c'_{\mu\nu}$.

- `hnc_ng` – Solve the HNC closure to find a new $c'_{\mu\nu}$ given $e'_{\mu\nu}$, but using the Ng acceleration scheme.

- `msa_picard` – Solve the MSA closure to find $c'_{\mu\nu}$ from $e'_{\mu\nu}$, and take a Picard step in the sense of mixing a fraction `alpha` of the new $c'_{\mu\nu}$ with the old $c'_{\mu\nu}$.

- `msa_ng` – Solve the MSA closure to find a new $c'_{\mu\nu}$ given $e'_{\mu\nu}$, but using the Ng acceleration scheme.

- `conv_test` – Check convergence in iterative schemes by calculating the difference (saved in `error`) between the current and immediately preceeding $c'_{\mu\nu}$.
- `make_pair_functions` – from Eq. (21). The choice to present $h_{\mu\nu}$ rather $g_{\mu\nu}$ is made because there may be interest in the asymptotic behaviour of $h_{\mu\nu} \to 0$ as $r \to \infty$, and it is assumed the user can add '1' to get the pair distribution functions.
- `make_structure_factors` – from Eq. (22).
- `make_thermodynamics` – everything in section 2.6.
- `write_params` – Write out basic information.
- `write_thermodynamics` – Write out all thermodynamic quantities, assuming `make_thermodynamics` has been called either explicitly or implicitly.

Note that the three `make_*` routines are independent and can be called in any order.

5.2 User parameters

- `ncomp` – The number of component species, arbitrary ≥ 1 (default 1).
- `ng` – The real space grid size, n, usually a power of 2 (default 4096).
- `nps` – The length of the Ng trajectory (usually safe to leave at the default 6).
- `npic` – The number of Picard steps to pump-prime the Ng acceleration scheme (usually safe to leave at the default 6, but should be $\geq$ `nps`).
- `maxsteps` – maximum number of steps to take before giving up on convergence (usually safe to leave at default 100).
- `verbose` – Flag (0 or 1) to print diagnostics (default 0).
- `cold_start` – Flag (0 or 1) to indicate that that the solver should execute a cold start by initialising $c'_{\mu\nu}$; this flag is by default only set for the first call to the solvers since it is often the case that a later call can re-use the current solution as the initial guess. The flag can be re-set by the user for subsequent calls.

- `start_type` – In a cold start initialise $c'_{\mu\nu} = 0$ (start type 1), $c'_{\mu\nu} = -\beta v'_{\mu\nu}$ (start type 2), or $c'_{\mu\nu} = \exp(-\beta v'_{\mu\nu}) - 1$ (start type 3, default). Usually safe to leave at default.

- `auto_fns` – Flag (0 or 1) if set (default) the integral equation solver calls all the `make_*` routines on exit. Since these routines are relatively inexpensive compared to the calculating the solution, one would usually leave this flag set.

- `deltar` – The real space grid spacing δ_r (default 0.01).

- `alpha` – Fraction of new solution in Picard method (usually safe to leave at default 0.2).

- `tol` – Error tolerance for claiming convergence (usually safe to leave at default 10^{-12}).

- `rc` – DPD repulsion range r_c (default 1.0).

- `lb` – Coulomb coupling strength l_B (default 0.0)

- `sigma` – smearing length σ (default 1.0) in the case of Gaussian or Bessel charges in the DPD potential (`charge_type = 1-2`), and URPM potentials; hard core diameter σ in the case of the RPM.

- `kappa` – parameter κ (default -1.0) in the softened RPM potential. A negative value is interpreted as $\kappa \to \infty$ and generates the pure RPM.

- `sigmap` – charge size σ' (default 1.0), used for the softened URPM potential.

- `rgroot` – linear (Groot) charge smearing range R (default 1.0) in the DPD potential (`charge_type = 3`).

- `lbda` – Slater charge smearing length λ (default 1.0) in the DPD potential with exact interaction (`charge_type = 4`). The name is truncated because `lambda` is a reserved word in python.

- `beta` – Slater charge smearing parameter β (default 1.0) in the DPD potential with approximate interaction (`charge_type = 5`). One would usually calculate this from $\beta = 1/\lambda$, or (better) $\beta = 5/(8\lambda)$.

- `rho(:)` – An array of size `ncomp` where `rho(mu)` is the density ρ_μ. All entries default to zero after a call to `initialise`.

- `z(:)` – An array of size `ncomp` where `z(mu)` is z_μ. All entries default to zero after a call to `initialise`.

- `arep(:, :)` – An array of size `ncomp`×`ncomp` where `arep(mu, nu)` is $A_{\mu\nu}$. Note that only elements above the diagonal are used, *e. g.* `arep(1,2)`, `arep(1,3)`, and `arep(2,3)`. **Elements below the diagonal are ignored**, *c. f.* the matrix in Eq. (73). All entries default to zero after a call to `initialise`.

5.3 Outputs

- `deltak` – The reciprocal space grid spacing $\delta_k = \pi/(n\delta_k)$, fixed by the needs of the fast Fourier transform algorithm. Available after a call to `initialise`.

- `hr(:, :, :)` – An array containing the total correlation functions $h_{\mu\nu}(r)$ as in Eq. (21). Specifically `hr(i, mu, nu)` contains $h_{\mu\nu}(r_i)$ where $r_i = i \times \delta_r$ for $i = 1 \ldots n-1$. The values at $r = 0$ can be obtained by linear extrapolation [4]. These arrays are available after a call to `make_pair_functions` (which is done automatically by the solver routines unless `auto_fns` is turned off).

- `sk(:, :, :)` – An array containing structure factors $S_{\mu\nu}(k)$ as in Eq. (22). Specifically `sk(j, mu, nu)` contains $S_{\mu\nu}(k_j)$ where $k_j = j \times \delta_k$ for $j = 1 \ldots n-1$. These arrays are available after a call to `make_structure_functions` (which is done automatically by the solver routines unless `auto_fns` is turned off).

- `r(:)` – The array `r(i)` contains $r_i = i \times \delta_r$ for $i = 1 \ldots n-1$. Available after a call to `initialise`.

- `k(:)` – The array `k(j)` contains $k_j = j \times \delta_k$ for $j = 1 \ldots n-1$. Available after a call to `initialise`.

- `error` – The final error estimate for the iterative solvers, should be less than `tol` if converged.

- `potential_type` – The last used potential type. This is zero if no potential has been initialised, is equal to `charge_type` in the case where `dpd_potential` was called, is equal to `10 + use_short` when `soft_urpm_potential` is called, and is equal to `20 + use_short` when `soft_rpm_potential` is called.

- The fixed parameters `pi`, `twopi`, and `fourpi` (π, 2π, and 4π) are available and can be utilised if convenient.

The following quantities are available after a call to `make_thermodynamics` (which is done automatically by the solver routines unless `auto_fns` is turned off). They are also reported by `write_thermodynamics`.

- `press` – The virial route pressure βp.
- `cf_mf` – The mean-field contribution to the virial route compressibility factor $\beta p/\rho$.
- `cf_gc` – The contact contribution to the virial route compressibility factor $\beta p/\rho$.
- `cf_xc` – The correlation contribution to the virial route compressibility factor $\beta p/\rho$.
- `comp` – The compressibility $\partial(\beta p)/\partial\rho$ (not to be confused with the compressibility factor $\beta p/\rho$).
- `comp_xc` – The correlation contribution to the compressibility.
- `uv` – Internal energy density $\beta U/V$ (excludes kinetic contribution).
- `un` – Internal energy per particle $\beta U/N$ (excludes kinetic contribution).
- `un_mf` – The mean field contribution to `un`.
- `un_corrn` – The correlation contribution to `un`.
- `muex(:)` – Chemical potentials, `muex(mu)` contains $\beta\mu_\mu^{\mathrm{ex}}$ (HNC only).
- `fvex` – Excess free energy density $\beta F^{\mathrm{ex}}/V$ (HNC only).
- `fnex` – Excess free energy per particle $\beta F^{\mathrm{ex}}/N$ (HNC only).

6 Examples

6.1 Virial route pressure

As a basic example the following minimalist `python` code solves the HNC closure problem for standard one-component DPD at $\rho = 3$ and $A = 25$, and prints the virial route pressure,

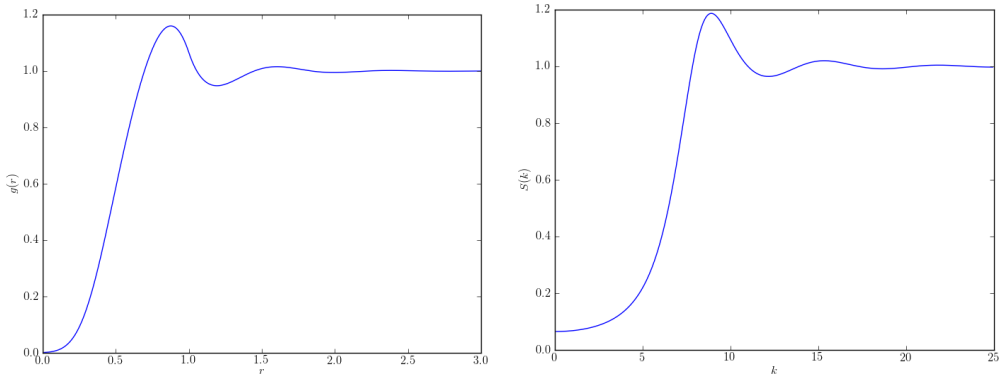

Figure 1: HNC pair distribution function and structure factor for standard DPD at $\rho = 3$ and $A = 25$.

```
from oz import wizard as w
w.initialise()
w.arep[0,0] = A = 25.0
w.dpd_potential(1)
w.rho[0] = rho = 3.0
w.hnc_solve()
print 'rho =', rho, ' A =', A
print 'pressure =', w.press
print 'energy density =', w.uv
```

with the result

```
rho = 3.0  A = 25.0
pressure = 23.5641475668
energy density = 13.7619524487
```

The first line imports the `wizard` of `oz` and renames it `w`. The `oz` package and the `wizard` module are automatically generated by `f2py`. The actual results from accurate Monte-Carlo simulations are $p = 23.71 \pm 0.01$ and $e = 13.83 \pm 0.01$, so the HNC virial route result is in error by $\sim 0.5\%$.

A virtue of `python` is the large package library, for instance for graphical output. The following code uses the `matplotlib` package to plot the pair distribution function and structure factor (with a standard normalisation),

```
import matplotlib.pyplot as plt
from oz import wizard as w
w.initialise()
w.arep[0,0] = A = 25.0
```

```
w.dpd_potential(1)
w.rho[0] = rho = 3.0
w.hnc_solve()
plt.figure(1)  # This will be g(r)
imax = int(3.0 / w.deltar)
plt.plot(w.r[0:imax], 1.0 + w.hr[0:imax,0,0])
plt.xlabel('$r$')
plt.ylabel('$g(r)$')
plt.figure(2)  # This will be S(k)
jmax = int(25.0 / w.deltak)
plt.plot(w.k[0:jmax], w.sk[0:jmax,0,0]/rho)
plt.xlabel('$k$')
plt.ylabel('$S(k)$')
plt.show()
```

The result is shown in Fig. 1 (imagine trying to do this in pure FORTRAN!).

6.2 Compressibility route pressure

The following `python` routine integrates the compressibility along an isotherm to calculate the compressibility route pressure,

```
def cr_press(drho):
    p_xc = prev = 0.0
    n = int(w.rho[0]/drho + 0.5)
    w.cold_start = 1
    for i in range(n):
        w.rho[0] = drho * (i + 1.0)
        w.hnc_solve()
        p_xc = p_xc + 0.5 * drho * (prev + w.comp_xc)
        prev = w.comp_xc
    return w.rho[0] + p_xc
```

Points to note are that a trapezium rule is used for the numerical integration, and the routine actually integrates the excess compressibility since the ideal contribution to the pressure is trivially ρ. When the `cr_press` routine finishes, the system is in the same state as it started, with the thermodynamics completely solved. So for example

```
w.initialise()
w.arep[0,0] = 25.0
w.dpd_potential(1)
```

```
w.rho[0] = 3.0
print 'CR pressure =', cr_press(0.05)
print 'VR pressure =', w.press
```

results in

```
CR pressure = 22.6662332439
VR pressure = 23.5641475668
```

Pressures calculated by the two routes differ by about 4% which appears to be typical for HNC for soft potentials. Also, further investigation reveals the dependence on $\delta\rho$ is linear and extrapolates to $p = 22.31$ at $\delta\rho \to 0$, for $\rho = 3$ and $A = 25$. So the error from discretising the isotherm is here $< 2\%$.

6.3 Free energy

One can continue in the same vein, to calculate the pressure *via* the energy route, but this really involves the calculation of the free energy. This can be done by coupling constant integration. The basic result can be derived by differentiating the fundamental expression $e^{-\beta F} = \int d\Omega\, e^{-\beta V}$ with respect to a parameter in the potential V. For instance for standard single-component DPD, $\partial F/\partial A = \langle V \rangle / A$ where $\langle V \rangle \equiv U$ is the internal energy. Thus

$$F = \int_0^A \frac{dA'}{A'} \langle V \rangle_{A'} \tag{75}$$

which should be evaluated along an isochore (constant density line). The following `python` routine uses this to compute the excess free energy per particle, $f_N^{\rm ex}$,

```
def fnex1(dA):
    fnex_xc = prev = 0.0
    n = int(w.arep[0,0]/dA + 0.5)
    w.cold_start = 1
    for i in range(n):
        w.arep[0,0] = dA * (i + 1.0)
        w.dpd_potential(1)
        w.hnc_solve()
        curr = w.un_xc / w.arep[0,0]
        fnex_xc = fnex_xc + 0.5*dA*(prev + curr)
        prev = curr
    return w.un_mf + fnex_xc
```

Points to note are again that this implements a trapezium rule, and that the actual integration is carried out for the correlation contribution for which U/A vanishes as $A \to 0$. The mean field contribution to the internal energy is strictly proportional to A, so it passes unscathed through Eq. (75) to contribute to the free energy.

For the HNC specifically, we could also calculate $f_N^{\rm ex}$ by integrating $\mu^{\rm ex}$ along an isotherm, giving a second routine

```
def fnex2(drho):
    fvec = prev = 0.0
    n = int(w.rho[0]/drho + 0.5)
    w.cold_start = 1
    for i in range(n):
        w.rho[0] = drho * (i + 1.0)
        w.hnc_solve()
        fvex = fvex + 0.5*drho*(prev + w.muex[0])
        prev = w.muex[0]
    return fvex / w.rho[0]
```

Note that integrating $\mu^{\rm ex}$ with respect to ρ generates the excess free energy density, so the routine divides this ρ to get $f_N^{\rm ex}$. Both routines leave the system in the same state as it started. As an example, to test these,

```
w.initialise()
w.rho[0] = 3.0
w.arep[0,0] = 25.0
w.dpd_potential(1)
print 'energy fnex =', fnex1(0.1)
print 'mu fnex =    ', fnex2(0.05)
print 'HNC fnex =   ', w.fnex
```

(the last of these uses the built-in free energy evaluation for HNC), giving

```
energy fnex = 5.31588405588
mu fnex =     5.31606130848
HNC fnex =    5.31593361272
```

We see that all three routes agree to the fourth decimal place, which is a good test of convergence.

Sections 6.1–6.3 have been coded up as `examples.py`.

6.4 Further examples

- `gw_p_compare.py` – Uses `matplotlib` to plot $(p-\rho)/A\rho^2$ versus ρ (using virial route pressure), and compare with data from Fig. 4 of Ref. [10]. The agreement is very good.

- `wsg_fig5_compare.py` – The free energy difference ΔF between a system of $N-1$ A particles and a B particle, and a pure system of N A particles (as a function of ΔA, where $A_{00} = A_{11} = 25$, $A_{01} = 25 + \Delta A$), was introduced in Ref. [11] and is representative of an effective χ value. Here, the free energy change is calculated as $\Delta F = \mu_1^{\text{ex}} - \mu_0^{\text{ex}}$ in a two component problem with the mole fraction of the second species set to zero ($\rho_0 = 3$ and $\rho_1 = 0$). The results are compared with data from Fig. 5 of Ref. [11], and with separate accurate Monte-Carlo simulations using trial particle identity swap moves.

- `x_dmu_compare.py` – In an unpublished extension to Ref. [11], $\Delta F = \mu_1^{\text{ex}} - \mu_0^{\text{ex}}$ is plotted as a function of the mole fraction x in a mixture where $\rho_0 = (1-x)\rho$ and $\rho_1 = x\rho$, for $\Delta A = 5$ (one cannot have ΔA too large otherwise an underlying demixing transition causes HNC to fail). The results are compared to accurate Monte-Carlo (MC) simulations. The agreement between HNC and MC is excellent, including the small deviation from linearity. The near-exact linear dependence can be represented by $\Delta F = \chi(1-2x)$ where χ is the Flory χ-parameter. This is the basic reason why DPD is so good at representing fluid mixtures which fit Flory-Huggins (regular solution) theory.

- `hm_fig10-2_compare.py` solves the MSA and the HNC for the RPM and compares with Monte-Carlo data, as in Fig 10.2 of Ref. [3]. See comments in file for more details.

- `urpm_oneoff.py` and `urpm_scan.py` are a pair of driver routines for solving the ultrasoft restricted primitive model (URPM) [1], including the possibility of a neutral solvent species. The first routine can be used for one-off calculations of structure and thermodynamics at a chosen state point. The second routine is intended to scan a range of densities to zero in on the Kirkwood transition between pure exponential and damped oscillatory behaviour in the tails of the total correlation functions. Both routines are fully provided with command line options and sensible defaults, via the `argparse` module. To see the available options use `--help`. The option `--ncomp` selects between the pure URPM case (`--ncomp=2`, default) and the solvated URPM case (`--ncomp=3`).

- `rpm_oneoff.py` is a similar driver routine for exploring the RPM.
- `urpm_press.py` and `urpm_targp.py` are auxiliary scripts targetted at the high density URPM phase boundary where HNC still has a solution even at $l_B \gtrsim 100$.
- `surpm_oneoff.py` and `surpm_gr.py` report various aspects of solutions for the softened URPM model.
- `wertheim_thermo.py`, `wertheim_solver.py` are python scripts, and `wertheim_coex.sh` is a shell script, to solve the URPM coexistence problem using HNC for the high density phase boundary and Wertheim theory with a softened URPM reference fluid (solved by HNC) for the low density phase boundary. For more details and sample results see comments in the scripts.
- Sundry items: `driver2-4.f90` and `urpm.f90` are sample FORTRAN 90 driver routines, retained for regression testing, as is `driver3.py`; and `fftw_test.f90` is test code to comprehend how FFTW works.

References

[1] P. B. Warren, A. Vlasov, L. Anton and A. J. Masters "Screening properties of Gaussian electrolyte models, with application to dissipative particle dynamics", J. Chem. Phys. **138**, 204907 (2013).

[2] K.-C. Ng, "Hypernetted chain solutions for the classical one-component plasma up to $\Gamma = 7000$", J. Chem. Phys. **61**, 2680–2689 (1974).

[3] J.-P. Hansen and I. R. McDonald, "Theory of simple liquids 3rd edition" (Academic Press, Amsterdam, 2006).

[4] C. T. Kelley and B. Montgomery Pettitt, "A fast solver for the Ornstein-Zernike equations", J. Comp. Phys. **197**, 491–501 (2004). Beware typos in this paper! Note, i and j in the main text are $i-1$ and $j-1$ in this paper, and $n = N-1$. This paper also suggests that c and e at $r = 0$ are evaluated by simple linear extrapolation, $c_0 = 2c_1 - c_2$ and $e_0 = 2e_1 - e_2$; and c and e at $r = n\delta_r$ are zero, $c_n = e_n = 0$.

[5] L. Vrbka, M. Lund, I. Kalcher, J. Dzubiella, R. R. Netz, and W. Kunz, "Ion-specific thermodynamics of multicomponent electrolytes: A hybrid HNC/MD approach", J. Chem. Phys. **131**, 154109 (2009).

[6] P. B. Warren and A. Vlasov "Screening properties of four mesoscale smoothed charge models, with application to dissipative particle dynamics", J. Chem. Phys. **140**, 084904 (2014).

[7] R. D. Groot, "Electrostatic interactions in dissipative particle dynamics—simulation of polyelectrolytes and anionic surfactants", J. Chem. Phys. **118**, 11265 (2003).

[8] M. González-Melchor, E. Mayoral, M. E. Velázquez and J. Alejandre, "Electrostatic interactions in dissipative particle dynamics using the Ewald sums", J. Chem. Phys. **125**, 224107 (2006).

[9] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, "Numerical recipes in FORTRAN 77: 2nd edition" (CUP, Cambridge, 1992).

[10] R. D. Groot and P. B. Warren, "Dissipative particle dynamics: bridging the gap between atomistic and mesoscopic simulation", J. Chem. Phys. **107**, 4423 (1997).

[11] C. M. Wijmans, B. Smit and R. D. Groot, "Phase behavior of monomeric mixtures and polymer solutions with soft interaction potentials", J. Chem. Phys. **114**, 7644 (2001).