simplesolv

v0.4

Table of contents:

`simplesolv` is a tool to compute the solution of the equations of the "Simplified approach" to the repulsive Bose gas introduced in [CJL20, CJL21, CHe21]. This approach provides an approximation to various observables of the ground state of the Hamiltonian

$$H_N = -\frac{1}{2}\sum_{i=1}^{N}\Delta_i + \sum_{1\leqslant i\leqslant j\leqslant N} v(|x_i - x_j|) \tag{0.0.1}$$

in three dimensions, with periodic boundary conditions, in the thermodynamic limit $N \to \infty$ at fixed density ρ.

`simplesolv` is written in `julia`. The source code is located in the `src` directory of this bundle. Throughout the documentation, we will refer to the directory containing the `src` directory as the "installation directory", and will denote it by the bash variable `$SIMPLESOLV` (so that the main julia file, for instance, is located at `$SIMPLESOLV/src/main.jl`).

1. Basic usage

Denoting the location of the installation directory by `$SIMPLESOLV`, `simplesolv` is run by calling

```
julia $SIMPLESOLV/src/main.jl [args] <command>
```

where the optional arguments `[args]` take the form `[-p params] [-U potential] [-M method] [-s savefile]`.

A few commands support multithreaded execution. To enable `julia` to run on several processors, it should be run with the `-p` option. For example, to run on 8 CPUs, run

```
julia -p 8 $SIMPLESOLV/src/main.jl [args] <command>
```

`command` specifies which computation is to be carried out, such as `energy` to compute the ground state energy, or `condensate_fraction` for the uncondensed fraction. The list of available commands depends on the `method` argument, which specifies one of the available methods to solve the equation at hand. The available methods are (see section 2 for further details)

- `easyeq` (**default**) for the Simple or Medium equation, or any interpolation between them, with a soft potential using the Newton algorithm,
- `anyeq` for any equation in the "Simplified approach" using the Newton algorithm.
- `simpleq-Kv` for the Simple equation using explicit expressions involving $\mathfrak{K}v$ (see (2.3.3)),
- `simpleq-hardcore` for the Simple equation with a hard core potential using the Newton algorithm,
- `simpleq-iteration` for the Simple equation with a soft potential using the iteration defined in [CJL20].

Each method is described in detail below, along with the list of commands (`command`) and parameters (`params`) compatible with them. In addition, the scattering length can be displayed using the `scattering_length` command:

```
julia $SIMPLESOLV/src/main.jl scattering_length
```

`params` should be a '`;`' separated list of entries, each of which is of the form `key=value`. For example `-p "rho=1e-6;v_a=2"`. (Note that you should not end the list of parameters by a '`;`', otherwise `simplesolv` will interpret that as there being an empty parameter entry, which it cannot split into a key and value, and will fail.)

`potential` specifies which potential v should be used, from the following list (see section 3 for further details).

- `exp` (**default**) for $v(|x|) = ae^{-|x|}$,
- `tent` for $v(|x|) = \mathbb{1}_{|x|<b} a\frac{2\pi}{3}(1-\frac{|x|}{b})^2(\frac{|x|}{b}+2)$,
- `expcry` for $v(|x|) = e^{-|x|} - ae^{-b|x|}$,
- `npt` for $v(|x|) = x^2 e^{-|x|}$,
- `alg` for $v(|x|) = \frac{1}{1+\frac{1}{4}|x|^4}$,
- `algwell` for $v(|x|) = \frac{1+a|x|^4}{(1+|x|^2)^4}$.
- `exact` for $v(|x|) = \frac{12c(|x|^6 b^6(2e-b^2)+b^4|x|^4(9e-7b^2)+4b^2|x|^2(3e-2b^2)+(5e+16b^2))}{(1+b^2|x|^2)^2(4+b^2|x|^2)^2((1+b^2|x|^2)^2-c)}$

The parameters in the potential can be set using the `params` argument: to set a set `v_a`, to set b set `v_b`, to set c set `v_c`, and to set e set `v_e`.

`savefile` can be used to accelerate the computation of observables in the `compleq` equation. Indeed, as is discussed in section 2.2, the computation of `compleq` is based on the computation of a large matrix, which can be pre-computed, saved in a file using the `save_Abar` command, and reused by specifying that file in the `savefile` argument.

2. Methods

In this section, we describe the different computation methods.

2.1. `easyeq`

This method is used to solve a family of equations, called `easyeq`, that interpolate between the Simple equation and the Medium equation:

$$-\Delta u = v(1-u) - 2\rho K + \rho^2 L \tag{2.1.1}$$

with

$$K := \beta_K u * S + (1-\beta_K)\frac{2e}{\rho}u, \quad L := \beta_L u * u * S + (1-\beta_L)\frac{2e}{\rho}u * u \tag{2.1.2}$$

$$S := (1-u)v, \quad e := \frac{\rho}{2}\int dx\ (1-u(|x|))v(|x|). \tag{2.1.3}$$

for a soft potential v at density $\rho > 0$.

The special choice $\beta_K = \beta_L = 0$ is called the Simple equation (`simpleq`), and the choice $\beta_K = \beta_L = 1$ is called the Medium equation (`medeq`)

2.1.1. Usage

Unless otherwise noted, this method takes the following parameters (specified via the `[-p params]` flag, as a '`;`' separated list of entries).

- `rho` (`Float64`, default: 10^{-6}): density ρ.
- `tolerance` (`Float64`, default: 10^{-11}): maximal size of final step in Newton iteration.
- `maxiter` (`Int64`, default: 21): maximal number of iterations of the Newton algorithm before giving up.
- `order` (`Int64`, default: 100): order used for all Gauss quadratures (denoted by N below).
- `minlrho_init` (`Float64`, default: -6): to initialize the Newton algorithm, we first compute the solution for a smaller ρ, `minlrho` is the minimal value for $\log_{10}\rho$ to start this initialization process.
- `nlrho_init` (`Int64`, default: 0): number of steps in the initialization process described above. Set to 0 to disable the incremental initialization process.
- `bK`, `bL` (`Float64`, default: 1, 1): the values of β_K and β_L.
- `eq` (`String`, default: "`simpleq`", acceptable values: "`simpleq`", "`medeq`"): A shortcut to select either the Simple equation ($\beta_K = \beta_L = 0$) or the Medium equation ($\beta_L = \beta_K = 1$). When this option is set, neither `bK` nor `bL` should be set.

The available `commands` are the following.

- `energy`: compute the energy e at a given ρ.
  Output: $[e]$ [Newton error ϵ].
- `energy_rho`: compute the energy e as a function of ρ. The Newton algorithm is initialized with the hardcore scattering solution (2.1.26) for the lowest ρ, and with the previously computed ρ for the larger densities.
  Disabled parameters: `rho`.
  Extra parameters:
  - `minlrho` (`Float64`, default: 10^{-6}): minimal value for $\log_{10}\rho$.
  - `maxlrho` (`Float64`, default: 10^{2}): maximal value for $\log_{10}\rho$.
  - `nlrho` (`Int64`, default: 100): number of values for ρ (spaced logarithmically).
  - `minrho` (`Float64`, default: 10^{-6}): minimal value for ρ.
  - `maxrho` (`Float64`, default: 10^{2}): maximal value for ρ.
  - `nrho` (`Int64`, default: 0): number of values for ρ (spaced linearly). If `nrho` is $\neq 0$, then the linear spacing will be used, and `minlrho`, `maxlrho`, `nlrho` will be ignored. Otherwise, the logarithmic spacing will be used and `minrho`, `maxrho` will be ignored.
  - `rhos` (`Array{Float64}`): list of values for ρ, specified as a '`,`' separated list. This parameter takes precedence over `minlrho`, `maxlrho`, `nlrho`, `minrho`, `maxrho`, `nrho`.

  Output (one line for each value of ρ): $[\rho]$ $[e]$ [Newton error ϵ].
- `condensate_fraction`: compute the uncondensed fraction η at a given ρ.
  Output: $[\eta]$ [Newton error ϵ].
- `condensate_fraction_rho`: compute the uncondensed fraction η as a function of ρ. The Newton algorithm is initialized with the hardcore scattering solution (2.1.26) for the lowest ρ, and with the previously computed ρ for the larger densities.
  Disabled parameters: same as `energy_rho`.
  Extra parameters: same as `energy_rho`.
  Output (one line for each value of ρ): $[\rho]$ $[\eta]$ [Newton error ϵ].

- `uk`: compute the Fourier transform $\hat{u}(|k|)$. The values $|k|$ at which $\hat{u}$ is computed are those coming from the Gauss quadratures, and cannot be set.
  Output (one line for each value of $|k|$): $[|k|]$ $[\hat{u}(|k|)]$
- `ux`: compute u as a function of $|x|$.
  Extra parameters:
  - `xmin` (`Float64`, default: 0): minimum of the range of $|x|$ to be printed.
  - `xmax` (`Float64`, default: 100): maximum of the range of $|x|$ to be printed.
  - `nx` (`Int64`, default: 100): number of points to print (linearly spaced).

  Output (one line for each value of x): $[|x|]$ $[u(|x|)]$
- `uux`: compute $2u - \rho u * u$ as a function of $|x|$.
  Extra parameters: Same as `ux`.
  Output (one line for each value of x): $[|x|]$ $[2u(|x|) - \rho u * u(|x|)]$
- `2pt`: compute the spherically averaged two-point correlation function $C_2(|x|)$ at a given ρ.
  Extra parameters: same as `ux`, plus
  - `window_L` (`Float64`, default: 10^3): size of the Hann window used to numerically invert the Fourier transform in the computation of the two-point correlation function, see (2.2.139).

  Output (one line for each value of $|x|$): $[|x|]$ $[C_2(|x|)]$
- `2pt_max`: compute the maximum of the spherically averaged two-point correlation function $C_2(|x|)$ at a given ρ.
  Extra parameters: `window_L` plus
  - `dx` (`Float64`, default: 10^{-7}): step used to numerically approximate derivatives.
  - `x0` (`Float64`, default: 1): initial guess for the maximum is $\rho^{-1/3}$`x0`.
  - `maxstep` (`Float64`, default: ∞): maximal size of single step in maximization algorithm.
  - `tolerance_max` (`Float64`, default: `tolerance`): same as `tolerance`, used for the Newton algorithm underlying the maximization algorithm.

  Output: $[|x_{\max}|]$ $[C_2(|x_{\max}|)]$
- `2pt_max_rho`: compute the maximum of the spherically averaged two-point correlation function $C_2(|x|)$ for a range of ρ.
  Extra parameters: same as `easyeq_2pt_max` plus those of `energy_rho`.
  Output (one line for each value of ρ): $[\rho]$ $[|x_{\max}|]$ $[C_2(|x_{\max}|)]$
  Multithread support: yes, different values of ρ split up among workers.
- `2pt_fourier`: compute the spherically averaged Fourier transform of the two-point correlation function $\hat{C}_2(|k|)$ at a given ρ.
  Extra parameters:
  - `kmin` (`Float64`, default: 0): minimum of the range of $|k|$ to be printed.
  - `kmax` (`Float64`, default: 10): maximum of the range of $|k|$ to be printed.
  - `nk` (`Int64`, default: 100): number of $|k|$'s to be printed.
  - `window_L` (`Float64`, default: 1000): what is actually computed is $\hat{C}_2$ convolved with a Gaussian of variance $1/\sqrt{L}$ with $L =$ `window_L`, see (2.1.67).

  Output (one line for each value of $|k|$): $[|k|]$ $[\hat{C}_2(|k|)]$.
  Multithread support: yes, different values of k are split up among workers.

- `2pt_fourier_max`: compute the maximum of the spherically averaged Fourier transformed two-point correlation function $\hat{C}_2(|k|)$.
  Extra parameters: `window_L`, `maxstep` plus
  - `dk` (`Float64`, default: 10^{-7}): step used to numerically approximate derivatives.
  - `k0` (`Float64`, default: 1): initial guess for the maximum is $\rho^{1/3}$`k0`.

  Output: $[|k_{\max}|]$ $[\hat{C}_2(|k_{\max}|)]$
- `2pt_fourier_max_rho`: compute the maximum of the spherically averaged Fourier transformed two-point correlation function $\hat{C}_2(|k|)$ for a range of ρ.
  Extra parameters: same as those of `2pt_fourier_max` plus those of `energy_rho`.
  Output (one line for each value of ρ): $[\rho]$ $[|k_{\max}|]$ $[\hat{C}_2(|k_{\max}|)]$
  Multithread support: yes, different values of ρ split up among workers.
- `momentum_distribution`: compute the momentum distribution $\mathcal{M}(|k|)$ at a given ρ. The momentum distribution is computed for $k = k_{l,j}$ (see (2.2.40)).
  Extra parameters:
  - `kmin` (`Float64`, default: 0): minimum of the range of $|k|$ to be printed.
  - `kmax` (`Float64`, default: 10): maximum of the range of $|k|$ to be printed.
  - `window_L` (`Float64`, default: 1000): what is actually computed is $\mathfrak{M}_k$ convolved with a Gaussian of variance $1/\sqrt{L}$ where $L = \sqrt{\texttt{window\_L}}/k^2$, see (2.1.79).

  Output (one line for each value of $|k|$): $[|k|]$ $[\mathcal{M}(|k|)]$

2.1.2. Description

2.1.2.1. Fourier space formulation

The computation is carried out in Fourier space. We take the convention that the Fourier transform of a function $f(|x|)$ is

$$\hat{f}(|k|) = \int_{\mathbb{R}^3} dx\ e^{ikx} f(|x|) = \frac{4\pi}{|k|}\int_0^\infty dr\ r\sin(|k|r) f(r). \tag{2.1.4}$$

In Fourier space, (2.1.1) becomes

$$k^2\hat{u} = \hat{S} - 2\rho\hat{A}_K\hat{u} + \rho^2\hat{A}_L\hat{u}^2 \tag{2.1.5}$$

$$\hat{A}_K := \beta_K\hat{S} + (1-\beta_K)\frac{2e}{\rho}, \quad \hat{A}_L := \beta_L\hat{S} + (1-\beta_L)\frac{2e}{\rho} \tag{2.1.6}$$

with

$$\hat{S}(|k|) = \int_{\mathbb{R}^3} dx\ e^{ikx}(1-u(|x|))v(|x|) = \hat{v}(k) - \hat{u}\hat{*}\hat{v}(k), \quad \hat{f}\hat{*}\hat{g}(|k|) := \int_{\mathbb{R}^3}\frac{dp}{(2\pi)^3}\ \hat{f}(|k-p|)\hat{g}(|p|). \tag{2.1.7}$$

We write this as a quadratic equation for $\hat{u}$, and solve it, keeping the solution that decays as $|k| \to \infty$:

$$\rho\hat{u}(|k|) = \frac{A_K(|k|)}{A_L(|k|)}\left(\xi(|k|) + 1 - \sqrt{(\xi(|k|)+1)^2 - \frac{A_L(|k|)}{A_K^2(|k|)}\hat{S}(|k|)}\right) \tag{2.1.8}$$

that is,

$$\rho\hat{u}(|k|) = \frac{\hat{S}(|k|)}{2A_K(|k|)(\xi(|k|)+1)}\Phi\left(\tfrac{A_L(|k|)}{A_K^2(|k|)}\tfrac{\hat{S}(|k|)}{(\xi(|k|)+1)^2}\right) \tag{2.1.9}$$

with

$$\xi(|k|) := \frac{k^2}{2\rho\hat{A}_K}, \quad \Phi(x) := \frac{2(1-\sqrt{1-x})}{x}. \tag{2.1.10}$$

We can write this as a root finding problem:

$$\Omega(u) := \rho\hat{u}(|k|) - \frac{\hat{S}(|k|)}{2A_K(|k|)(\xi(|k|)+1)}\Phi\left(\frac{A_L(|k|)}{A_K^2(|k|)}\frac{\hat{S}(|k|)}{(\xi(|k|)+1)^2}\right) = 0. \tag{2.1.11}$$

Furthermore, using bipolar coordinates (see lemma A3.1), we write $\hat{S}$ as

$$\hat{S}(|k|) = \hat{v}(|k|) - \frac{1}{8\pi^3}\int_0^\infty dt\ t\hat{u}(t)H(|k|,t), \quad H(y,t) := \frac{2\pi}{y}\int_{|y-t|}^{y+t} ds\ s\hat{v}(s). \tag{2.1.12}$$

By a simple change of variables,

$$H(y,t) = 4\pi\left(\mathbb{1}_{y>t}\frac{t}{y} + \mathbb{1}_{y\leqslant t}\right)\int_0^1 ds\ ((y+t)s + |y-t|(1-s))v((y+t)s+|y-t|(1-s)). \tag{2.1.13}$$

2.1.2.2. Evaluating integrals

To compute these integrals numerically, we will use Gauss-Legendre quadratures:

$$\int_0^1 dt\ f(t) \approx \frac{1}{2}\sum_{i=1}^N w_i f\left(\tfrac{r_i+1}{2}\right) \tag{2.1.14}$$

where w_i and r_i are the Gauss-Legendre *weights* and *abcissa*. The order N corresponds to the parameter `order`. The error made by the quadrature is estimated in appendix A2. We compactify the integrals to the interval $(0,1)$: $y := \frac{1}{t+1}$:

$$\hat{S}(|k|) = \hat{v}(|k|) - \frac{1}{8\pi^3}\int_0^1 dy\ \frac{(1-y)\hat{u}(\frac{1-y}{y})H(|k|,\frac{1-y}{y})}{y^3} \tag{2.1.15}$$

so, using the Gauss-Legendre quadrature, we approximate

$$\hat{S}(|k|) \approx \hat{v}(|k|) - \frac{1}{16\pi^3}\sum_{j=1}^N w_j\frac{(1-y_j)\hat{u}(\frac{1-y_j}{y_j})H(|k|,\frac{1-y_j}{y_j})}{y_j^3}, \quad y_j := \frac{r_j+1}{2}. \tag{2.1.16}$$

This suggests a natural discretization of Fourier space: let

$$k_i := \frac{1-r_i}{1+r_i} \equiv \frac{1-y_i}{y_i}. \tag{2.1.17}$$

Thus, defining

$$\mathbb{U}_i := \rho\hat{u}(k_i), \quad \hat{v}_i := \hat{v}(k_i) \tag{2.1.18}$$

and approximate (2.1.9):

$$\mathbb{U}_i = \frac{\mathbb{T}_i}{2(\mathbb{X}_i+1)}\Phi\left(\mathbb{B}_i\frac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right) \tag{2.1.19}$$

with

$$\mathbb{A}_{K,i} := \beta_K\mathbb{S}_i + (1-\beta_K)\mathbb{E}, \quad \mathbb{B}_i := \frac{\beta_L\mathbb{S}_i + (1-\beta_L)\mathbb{E}}{\mathbb{A}_{K,i}}, \quad \mathbb{T}_i := \frac{\mathbb{S}_i}{\mathbb{A}_{K,i}} \tag{2.1.20}$$

$$\mathbb{S}_i := \hat{v}_i - \frac{1}{16\pi^3\rho}\sum_{j=1}^N w_j\frac{(1-y_j)\mathbb{U}_jH(k_i,k_j)}{y_j^3}, \quad \mathbb{E} := \hat{v}(0) - \frac{1}{16\pi^3\rho}\sum_{j=1}^N w_j\frac{(1-y_j)\mathbb{U}_jH(0,k_j)}{y_j^3} \tag{2.1.21}$$

$$\mathbb{X}_i := \frac{k_i^2}{2\rho\mathbb{A}_{K,i}}. \tag{2.1.22}$$

This is a discrete equation for the vector $(\mathbb{U}_i)_{i=1}^N$.

2.1.2.3. Main algorithm to compute $\mathbb{U}$

1 - We rewrite (2.1.19) as a root finding problem:

$$\Xi_i(\mathbb{U}) := \mathbb{U}_i - \frac{\mathbb{T}_i}{2(\mathbb{X}_i+1)}\Phi\left(\mathbb{B}_i \tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right) = 0 \tag{2.1.23}$$

which we solve using the Newton algorithm, that is, we define a sequence of $\mathbb{U}$'s:

$$\mathbb{U}^{(n+1)} = \mathbb{U}^{(n)} - (D\Xi(\mathbb{U}^{(n)}))^{-1}\Xi(\mathbb{U}^{(n)}) \tag{2.1.24}$$

where $D\Xi$ is the Jacobian of Ξ:

$$(D\Xi)_{i,j} := \frac{\partial \Xi_i}{\partial \mathbb{U}_j}. \tag{2.1.25}$$

2 - For small values of ρ, we initialize the algorithm with the hardcore scattering solution

$$\hat{u}_0(k) = \frac{4\pi a_0}{k^2} \tag{2.1.26}$$

where a_0 is the scattering length of the potential v (or an approximation thereof, which need not be very good). Thus,

$$\mathbb{U}_i^{(0)} = \frac{4\pi a_0}{k_i^2}. \tag{2.1.27}$$

This is a good approximation for small ρ. For larger ρ, we choose $\mathbb{U}^{(0)}$ as the solution of `easyeq` for a slightly smaller ρ, and proceed inductively (using the parameters `minlrho_init` and `nlrho_init`).

3 - We are left with computing the Jacobian of Ξ:

$$\begin{aligned}\frac{\partial \Xi_i}{\partial \mathbb{U}_j} = \delta_{j,i} - \frac{1}{2(\mathbb{X}_i+1)}\left(\partial_j \mathbb{T}_i - \frac{\mathbb{T}_i \partial_j \mathbb{X}_i}{\mathbb{X}_i+1}\right)\Phi\left(\mathbb{B}_i \tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right) \\ - \frac{\mathbb{T}_i}{2(\mathbb{X}_i+1)^3}\left(\mathbb{B}_i\partial_j\mathbb{T}_i + \mathbb{T}_i\partial_j\mathbb{B}_i - 2\frac{\mathbb{B}_i\mathbb{T}_i\partial_j\mathbb{X}_i}{\mathbb{X}_i+1}\right)\partial\Phi\left(\mathbb{B}_i \tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right)\end{aligned} \tag{2.1.28}$$

with

$$\partial_j \mathbb{B}_i = (\beta_L(1-\beta_K) - \beta_K(1-\beta_L))\frac{\mathbb{E}\partial_j\mathbb{S}_i - \mathbb{S}_i\partial_j\mathbb{E}}{(\beta_K\mathbb{S}_i + (1-\beta_K)\mathbb{E})^2} \tag{2.1.29}$$

$$\partial_j \mathbb{T}_i = (1-\beta_K)\frac{\mathbb{E}\partial_j\mathbb{S}_i - \mathbb{S}_i\partial_j\mathbb{E}}{(\beta_K\mathbb{S}_i + (1-\beta_K)\mathbb{E})^2}, \quad \partial_j \mathbb{A}_{K,i} = \beta_K\partial_j\mathbb{S}_i + (1-\beta_K)\partial_j\mathbb{E} \tag{2.1.30}$$

$$\partial_j \mathbb{S}_i := -\frac{1}{16\pi^3\rho}w_j\frac{(1-y_j)H(k_i,k_j)}{y_j^3}, \quad \partial_j\mathbb{E} := -\frac{1}{16\pi^3\rho}\sum_{j=1}^{N} w_j\frac{(1-y_j)H(0,k_j)}{y_j^3} \tag{2.1.31}$$

$$\partial_j \mathbb{X}_i := -\frac{k_i^2}{2\rho\mathbb{A}_{K,i}^2}\partial_j\mathbb{A}_{K,i}. \tag{2.1.32}$$

4 - We iterate the Newton algorithm until the Newton relative error ϵ becomes smaller than the `tolerance` parameter. The Newton error is defined as

$$\epsilon = \frac{\|\mathbb{U}^{(n+1)} - \mathbb{U}^{(n)}\|_2}{\|\mathbb{U}^{(n)}\|_2} \tag{2.1.33}$$

where $\|\cdot\|_2$ is the l_2 norm. The energy thus obtained is

$$e = \frac{\rho}{2}\mathbb{E} \tag{2.1.34}$$

the Fourier transform $\hat{u}$ of the solution is

$$\hat{u}(k_i) \approx \frac{\mathbb{U}_i}{\rho} \tag{2.1.35}$$

and the solution u in real space is obtained by inverting the Fourier transform

$$\begin{aligned} u(|x|) = \int \frac{dk}{8\pi^3}\, e^{-ikx}\hat{u}(|k|) &= \frac{1}{2\pi^2|x|}\int_0^1 dy\, \frac{(1-y)\sin(|x|\frac{1-y}{y})\hat{u}(\frac{1-y}{y})}{y^3} \\ &\approx \frac{1}{4\pi^2|x|}\sum_{j=1}^N w_j(1+k_j)^2 k_j \sin(|x|k_j)\hat{u}(k_j). \end{aligned} \tag{2.1.36}$$

To compute $2u - \rho u * u$, we replace $\hat{u}$ with $2\hat{u} - \rho\hat{u}^2$ in the previous equation.

2.1.2.4. Condensate fraction

Finally, to compute the uncondensed fraction, we solve the modified `easyeq` (see [CJL21])

$$(-\Delta + 2\mu)u_\mu = v(1-u_\mu) - 2\rho K + \rho^2 L \tag{2.1.37}$$

where K and L are defined as in (2.1.2)-(2.1.3) in which u is replaced with u_μ. The uncondensed fraction is then

$$\eta = \partial_\mu e|_{\mu=0} = -\frac{\rho}{2}\int dx\; v(|x|)\partial_\mu u_\mu(|x|)|_{\mu=0}. \tag{2.1.38}$$

To compute the energy in the presence of the parameter μ, we proceed in the same way as for $\mu = 0$, the only difference being that k^2 should formally be replaced by $k^2 + 2\mu$. In other words, we consider $\mathbb{U}_i = u_\mu(|k_i|)$ and define $\Xi(\mathbb{U}, \mu)$ in the same way as in (2.1.23), except that $\mathbb{X}_i$ should be replaced by

$$\frac{k_i^2 + 2\mu}{2\rho\mathbb{A}_{K,i}}. \tag{2.1.39}$$

We then solve

$$\Xi(\mathbb{U}, \mu) = 0. \tag{2.1.40}$$

By differentiating this identity with respect to μ, we find $\partial_\mu u_\mu$:

$$D\Xi\partial_\mu\mathbb{U} = -\partial_\mu\Xi \tag{2.1.41}$$

and the uncondensed fraction is

$$\eta = \frac{1}{2}\int dx\; v(x)(D\Xi)^{-1}\partial_\mu\Xi|_{\mu=0} \tag{2.1.42}$$

thus

$$\eta = \frac{1}{16\pi^3}\int_0^1 dy\, \frac{(1-y)H(0,\frac{1-y}{y})(D\Xi)^{-1}\partial_\mu\Xi|_{\mu=0}(\frac{1-y}{y})}{y^3} \tag{2.1.43}$$

which we approximate using a Gauss-Legendre quadrature:

$$\eta \approx \frac{1}{32\pi^3}\sum_{j=1}^N w_j(1-k_j)^2 k_j H(0,k_j)\sum_{i=1}^N (D\Xi)^{-1}_{j,i}\partial_\mu\Xi_i|_{\mu=0}. \tag{2.1.44}$$

We then compute, using (2.1.28),

$$\partial_\mu\Xi_i|_{\mu=0} = \frac{\mathbb{T}_i}{2\rho\mathbb{A}_{K,i}(\mathbb{X}_i+1)^2}\Phi\left(\mathbb{B}_i\tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right) + \frac{\mathbb{B}_i\mathbb{T}_i^2}{\rho\mathbb{A}_{K,i}(\mathbb{X}_i+1)^4}\partial\Phi\left(\mathbb{B}_i\tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right). \tag{2.1.45}$$

2.1.2.5. Correlation function (spherical average)

The two-point correlation function is

$$c_2(x) := 2\rho \frac{\delta e}{\delta v(x)} \tag{2.1.46}$$

and its spherical average is

$$C_2(|x|) := \frac{1}{4\pi|x|^2} \int dy\ \delta(|x| - |y|) c_2(y). \tag{2.1.47}$$

In Fourier space,

$$c_2(x) = 2\rho \int dk\ e^{ikx} \frac{\delta e}{\delta \hat{v}(k)} \tag{2.1.48}$$

so

$$C_2(|x|) = 2\rho \int dk\ \left(\frac{1}{4\pi|x|^2} \int dy\ \delta(|x| - |y|) e^{iky} \right) \frac{\delta e}{\delta \hat{v}(k)} = 2\rho \int dk\ \frac{\sin(|k||x|)}{|k||x|} \frac{\delta e}{\delta \hat{v}(k)}. \tag{2.1.49}$$

1 - We can compute this quantity by considering a modified `easyeq` in Fourier space, by formally replacing $\hat{v}$ with

$$\hat{v} + \lambda g(|k|), \quad g(|k|) := \frac{\sin(|k||x|)}{|k||x|}. \tag{2.1.50}$$

Indeed, if e_λ denotes the energy of this modified equation,

$$\partial_\lambda e_\lambda|_{\lambda=0} = \int dk\ \frac{\delta e}{\delta \hat{v}(k)} \partial_\lambda (\hat{v}(k) + \lambda g(|k|)) = \int dk\ g(|k|) \frac{\delta e}{\delta \hat{v}(k)}. \tag{2.1.51}$$

So, denoting the solution of the modified equation by u_λ,

$$C_2(x) = 2\rho \partial_\lambda e_\lambda|_{\lambda=0} = \rho^2 g(0) - \rho^2 \int \frac{dk}{(2\pi)^3}\ (g(k)\hat{u}(k) + \hat{v}(k)\partial_\lambda \hat{u}_\lambda(k)|_{\lambda=0}). \tag{2.1.52}$$

We compute $\partial_\lambda u_\lambda|_{\lambda=0}$ in the same way as the uncondensed fraction: we define $\Xi(\mathbb{U}, \lambda)$ by formally adding $\lambda g(|k|)$ to $\hat{v}$, solve $\Xi(\mathbb{U}, \lambda) = 0$, and differentiate:

$$\partial_\lambda \mathbb{U}|_{\lambda=0} = -(D\Xi)^{-1} \partial_\lambda \Xi|_{\lambda=0}. \tag{2.1.53}$$

2 - We compute $\partial_\lambda \Xi|_{\lambda=0}$:

$$\begin{aligned} \partial_\lambda \Xi_i = &-\frac{1}{2(\mathbb{X}_i + 1)} \left(\partial_\lambda \mathbb{T}_i - \frac{\mathbb{T}_i \partial_\lambda \mathbb{X}_i}{\mathbb{X}_i + 1} \right) \Phi \left(\mathbb{B}_i \tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2} \right) \\ &- \frac{\mathbb{T}_i}{2(\mathbb{X}_i + 1)^3} \left(\mathbb{B}_i \partial_\lambda \mathbb{T}_i + \mathbb{T}_i \partial_\lambda \mathbb{B}_i - 2 \frac{\mathbb{B}_i \mathbb{T}_i \partial_\lambda \mathbb{X}_i}{\mathbb{X}_i + 1} \right) \partial\Phi \left(\mathbb{B}_i \tfrac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2} \right) \end{aligned} \tag{2.1.54}$$

with

$$\partial_\lambda \mathbb{B}_i = (\beta_L(1 - \beta_K) - \beta_K(1 - \beta_L)) \frac{\mathbb{E}\partial_\lambda \mathbb{S}_i - \mathbb{S}_i \partial_\lambda \mathbb{E}}{(\beta_K \mathbb{S}_i + (1 - \beta_K)\mathbb{E})^2} \tag{2.1.55}$$

$$\partial_\lambda \mathbb{T}_i = (1 - \beta_K) \frac{\mathbb{E}\partial_\lambda \mathbb{S}_i - \mathbb{S}_i \partial_\lambda \mathbb{E}}{(\beta_K \mathbb{S}_i + (1 - \beta_K)\mathbb{E})^2}, \quad \partial_\lambda \mathbb{A}_{K,i} = \beta_K \partial_\lambda \mathbb{S}_i + (1 - \beta_K)\partial_\lambda \mathbb{E} \tag{2.1.56}$$

$$\partial_\lambda \mathbb{S}_i := g(k_i) - \frac{1}{16\pi^3 \rho} \sum_{j=1}^{N} w_j \frac{(1 - y_j)\mathbb{U}_j \partial_\lambda H(k_i, k_j)}{y_j^3} \tag{2.1.57}$$

$$\partial_\lambda \mathbb{E} := g(0) - \frac{1}{16\pi^3\rho}\sum_{j=1}^{N} w_j \frac{(1-y_j)\mathbb{U}_j \partial_\lambda H(0,k_j)}{y_j^3} \tag{2.1.58}$$

$$\partial_\lambda \mathbb{X}_i := -\frac{k_i^2}{2\rho \mathbb{A}_{K,i}^2}\partial_\lambda \mathbb{A}_{K,i} \tag{2.1.59}$$

where $\partial\lambda H$ is computed similarly to H (2.1.13) but with $\hat{v}$ replaced by g:

$$\partial_\lambda H(y,t) = 4\pi\left(\mathbb{1}_{y>t}\frac{t}{y} + \mathbb{1}_{y\leqslant t}\right)\int_0^1 ds\ ((y+t)s + |y-t|(1-s))g((y+t)s+|y-t|(1-s)). \tag{2.1.60}$$

3 - In order to invert the Fourier transform in (2.1.49) numerically, we will use a Hann window (see appendix A4)

$$H_L(k) := \mathbb{1}_{|k|<\frac{L}{2}}\cos^2(\tfrac{\pi|k|}{L}). \tag{2.1.61}$$

The parameter L is set using `window_L`. The computation is changed only in that g is changed to $H_L(k)\frac{\sin(|k||x|)}{|k||x|}$.

4 - To compute the maximum of C_2, we use a modified Newton algorithm. The initial guess for the maximum is $|x_0| = \rho^{-\frac{1}{3}}$`x0`. The modified Newton algorithm is an iteration:

$$x_{n+1} = x_n + \frac{\partial C_2(|x_n|)}{|\partial^2 C_2(|x_n|)|} \tag{2.1.62}$$

in which the derivatives are approximated using finite differences:

$$\partial C_2(x) \approx \frac{C_2(|x|+dx) - C_2(|x|)}{dx}, \quad \partial^2 C_2(x) \approx \frac{C_2(|x|+dx) + C_2(|x|-dx) - 2C_2(|x|)}{dx^2}. \tag{2.1.63}$$

This is a modification of the usual Newton iteration $x_n + \partial C_2/\partial^2 C_2$ which is designed to follow the direction of the gradient, and thus to move toward a local maximum. In addition, if $|\partial C_2|/|\partial^2 C_2|$ is larger than `maxstep`, then the step is replaced with $\pm$`maxstep`. This prevents the algorithm from stepping over a maximum and land on another, further away. This is useful if one has a good idea of where the global maximum is, and does not want to get trapped in a smaller local maximum.

The algorithm is run for a maximum of `maxiter` iterations, or until $|x_{n+1} - x_n|$ is smaller than `tolerance`. If the maximal number of iterations is reached, or if the solution found is not a local maximum, then the algorithm fails, and returns $+\infty$. The point thus computed is therefore a local maximum, but it is not guaranteed to be the global maximum.

2.1.2.6. Fourier transform of two-point correlation (spherical average)

The Fourier transform of the two-point correlation function is

$$\hat{c}_2(q) := 2\rho\frac{\delta e}{\delta v(q)} \tag{2.1.64}$$

and its spherical average is

$$\hat{C}_2(|q|) := \frac{1}{4\pi|q|^2}\int dk\ \delta(|q|-|k|)c_2(k) = \frac{\rho}{2\pi|q|^2}\int dk\ \delta(|q|-|k|)\frac{\delta e}{\delta \hat{v}(k)}. \tag{2.1.65}$$

1 - To compute $\frac{\delta e}{\delta \hat{v}(q)}$, one idea would be to proceed in the same way as for the two-point correlation function, by replacing $\hat{v}$ with

$$\hat{v} + \lambda g(|k|), \quad g(|k|) := \frac{1}{4\pi|q|^2}\delta(|q| - |k|) \tag{2.1.66}$$

where δ is the Dirac-delta function distribution (compare this with (2.1.50)). However, the δ function causes all sorts of problems with the quadratures.

2 - Instead, we approximate $\hat{C}_2$ by convolving it with a normalized Gaussian: let

$$\Gamma_L(|k|) := \left(\frac{L}{2\pi}\right)^{\frac{3}{2}} e^{-\frac{L}{2}k^2} \tag{2.1.67}$$

$$\hat{\mathfrak{C}}_2(|q|) := \int dp\ \hat{C}_2(|q-p|)\Gamma_L(|p|) = \int dk \int dp\ \frac{\rho}{2\pi|q-p|^2}\delta(|q-p|-|k|)\frac{\delta e}{\delta \hat{v}(k)}\Gamma_L(|p|) \tag{2.1.68}$$

which by lemma A3.1 is

$$\hat{\mathfrak{C}}_2(|q|) = \int dk\ \frac{\rho}{|q|}\frac{\delta e}{\delta \hat{v}(k)}\int_0^\infty dt \int_{||q|-t|}^{|q|+t} ds\ s\frac{\delta(t-|k|)}{t}\Gamma_L(s) \tag{2.1.69}$$

that is

$$\hat{\mathfrak{C}}_2(|q|) = \int dk\ \frac{\delta e}{\delta \hat{v}(k)}\frac{\rho}{|q||k|}\int_{||q|-|k||}^{|q|+|k|} ds\ s\Gamma_L(s) \tag{2.1.70}$$

which is the directional derivative of e with respect to $\hat{v}$ in the direction of $2\rho g$ with

$$g(|k|) := \frac{1}{2|q||k|}\int_{||q|-|k||}^{|q|+|k|} ds\ s\Gamma_L(s) = \frac{1}{2|k|rL}(\Gamma_L(|k|-r) - \Gamma_L(|k|+r)). \tag{2.1.71}$$

Note that

$$g(0) := \Gamma_L(|q|). \tag{2.1.72}$$

To compute this derivative, we replace $\hat{v}$ with

$$\hat{v} + \lambda g(|k|) \tag{2.1.73}$$

so, denoting the solution of the modified equation by u_λ, for $q \neq 0$,

$$\hat{\mathfrak{C}}_2(|q|) = 2\rho\partial_\lambda e_\lambda|_{\lambda=0} = \rho^2\left(-\int \frac{dk}{(2\pi)^3}\ g(|k|)\hat{u}(|k|) - \int \frac{dk}{(2\pi)^3}\ \hat{v}(|k|)\partial_\lambda \hat{u}_\lambda(|k|)|_{\lambda=0}\right). \tag{2.1.74}$$

To compute $\partial_\lambda \hat{u}_\lambda|_{\lambda=0}$, we differentiate $\Xi(\mathbb{U}, \lambda) = 0$:

$$\partial_\lambda \mathbb{U}|_{\lambda=0} = -(D\Xi)^{-1}\partial_\lambda \Xi|_{\lambda=0}. \tag{2.1.75}$$

The computation of $\partial_\lambda \Xi|_{\lambda=0}$ is identical to (2.1.54), but with the g defined in (2.1.71).

3 - To compute the maximum of $\hat{C}_2$, we proceed as for C_2, see (2.1.62)-(2.1.63). The only difference is that the algorithm is initialized with $|k_0| = \rho^{\frac{1}{3}}$`k0`.

2.1.2.7. Momentum distribution

To compute the momentum distribution (see [CHe21]), we add a parameter λ to `easyeq`:

$$-\Delta u_\lambda(|x|) = (1 - u_\lambda(|x|))v(|x|) - 2\rho K(|x|) + \rho^2 L(|x|) - 2\lambda \hat{u}_0(q)\cos(q\cdot x) \tag{2.1.76}$$

($\hat{u}_0 \equiv \hat{u}_\lambda|_{\lambda=0}$). The momentum distribution is then

$$\mathcal{M}(q) = \partial_\lambda e|_{\lambda=0} = -\frac{\rho}{2}\int \frac{dk}{(2\pi)^3}\ \hat{v}(k)\partial_\lambda \hat{u}_\lambda(k)|_{\lambda=0}. \tag{2.1.77}$$

1 - Note that the Fourier transform of $2\lambda\hat{u}_0(q)\cos(q\cdot x)$ is

$$-(2\pi)^3\lambda\hat{u}_0(q)(\delta(q+k)+\delta(q-k)). \tag{2.1.78}$$

The presence of delta functions does not play well with the quadratures. To get around this, we instead compute a regularization of $\mathcal{M}(q)$ by convolving it with a peaked spherically symmetric function. Let Γ_L denote the Gaussian with variance $1/\sqrt{L}$

$$\Gamma_L(|k|) := \left(\frac{L}{2\pi}\right)^{\frac{3}{2}} e^{-\frac{L}{2}k^2}. \tag{2.1.79}$$

In fact, we will scale L with k, and set L to

$$L = \sqrt{\texttt{window\_L}}/k^2. \tag{2.1.80}$$

To compute

$$\mathfrak{M}(q) := \mathcal{M} * \Gamma_L(q) \tag{2.1.81}$$

we solve the equation

$$-\Delta u_\lambda(|x|) = (1 - u_\lambda(|x|))v(|x|) - 2\rho K(|x|) + \rho^2 L(|x|) - 2\lambda\int dk\ \hat{u}_0(k)\cos(k\cdot x)\Gamma_L(q-k). \tag{2.1.82}$$

Note that the Fourier transform of

$$-2\lambda\int dk\ \hat{u}_0(k)\cos(k\cdot x)\Gamma_L(q-k) \tag{2.1.83}$$

is

$$-(2\pi)^3\lambda\hat{u}_0(q)(\Gamma_L(k+q)+\Gamma_L(k-q)). \tag{2.1.84}$$

Since the ground state is unique, $\mathcal{M}$ is spherically symmetric. The term $\Gamma_L(k\pm q)$ is not, so we take its spherical average (which will not change the final result): by lemma A3.1,

$$-\frac{1}{4\pi r^2}\int dq\ \delta(|q|-r)(2\pi)^3\lambda\hat{u}_0(q)(\Gamma_L(k+q)+\Gamma_L(k-q)) = -\frac{(2\pi)^3}{|k|r}\lambda\hat{u}_0(r)\int_{||k|-r|}^{|k|+r} ds\ s\Gamma_L(s). \tag{2.1.85}$$

In this setup, the approximation of the delta function is thus

$$\tilde{\delta}(|k|,r) := \frac{1}{2|k|r}\int_{||k|-r|}^{|k|+r} ds\ s\Gamma_L(s) = \frac{1}{2|k|rL}(\Gamma_L(|k|-r)-\Gamma_L(|k|+r)). \tag{2.1.86}$$

2 - To compute the momentum distribution at q, we define $\Xi(\mathbb{U},\lambda)$ by replacing $\mathbb{T}$ with

$$\mathbb{T}_i = \frac{1}{\mathbb{A}_{K,i}}\left(\mathbb{S}_i - 2(2\pi)^3\lambda\hat{u}(|q|)\tilde{\delta}(k_i,|q|)\right). \tag{2.1.87}$$

Then we solve $\Xi(\mathbb{U},\lambda)=0$, and differentiate:

$$\partial_\lambda \mathbb{U}|_{\lambda=0} = -(D\Xi)^{-1}\partial_\lambda \Xi|_{\lambda=0}. \tag{2.1.88}$$

Finally,

$$\partial_\lambda \Xi_i|_{\lambda=0} = -\partial_\lambda \mathbb{T}_i|_{\lambda=0}\left(\frac{1}{2(\mathbb{X}_i+1)}\Phi\left(\mathbb{B}_i\frac{\mathbb{T}_i}{(\mathbb{X}_i+1)^2}\right)+\frac{\mathbb{B}_i\mathbb{T}_i}{2(\mathbb{X}_i+1)^3}\partial\Phi\left(\mathbb{B}_i\frac{\mathbb{T}_i}{(\mathbb{X}_{l,j}+1)^2}\right)\right) \tag{2.1.89}$$

with

$$\partial_\lambda \mathbb{T}_i|_{\lambda=0} = -\frac{2(2\pi)^3}{\mathbb{A}_{K,i}}\hat{u}(|q|)\tilde{\delta}(k_i,|q|). \tag{2.1.90}$$

2.2. `anyeq`

This method is used to solve any of the equations in the Simplified approach. Specifically, it solves the equation

$$-\Delta u = v(1-u) - 2\rho K + \rho^2 L \tag{2.2.1}$$

with

$$K = \gamma_K(1-\alpha_K u)\left(\beta_K u * S + (1-\beta_K)\frac{2e}{\rho}u\right) \tag{2.2.2}$$

and

$$L := L_1 + L_2 + L_3 \tag{2.2.3}$$

$$L_1 := (1-\alpha_{L,1}u)\left(\beta_{L,1}u*u*S + (1-\beta_{L,1})\frac{2e}{\rho}u*u\right) \tag{2.2.4}$$

$$L_2 := -\gamma_{L,2}(1-\alpha_{L,2}u)\left(\beta_{L,2}2u*(u(u*S)) + (1-\beta_{L,2})\frac{4e}{\rho}u*u^2\right) \tag{2.2.5}$$

$$L_3 := \gamma_{L,3}(1-\alpha_{L,3}u)\left(\beta_{L,3}\frac{1}{2}\int dydz\ u(y)u(z-x)u(z)u(y-x)S(z-y) + (1-\beta_{L,3})\frac{e}{\rho}u^2*u^2\right) \tag{2.2.6}$$

$$e = \frac{\rho}{2}\int dx\ (1-u(|x|))v(|x|). \tag{2.2.7}$$

The parameters $\alpha_.$, $\beta_.$ and $\gamma_.$ can be set to turn (2.2.1) into any of the approximations of the Simplified approach. For ease of use, there are several predefined equations, given in the following table.

	α_K	β_K	γ_K	$\alpha_{L,1}$	$\beta_{L,1}$	$\alpha_{L,2}$	$\beta_{L,2}$	$\gamma_{L,2}$	$\alpha_{L,3}$	$\beta_{L,3}$	$\gamma_{L,3}$
`compleq`	1	1	1	1	1	1	1	1	1	1	1
`bigeq`	1	1	1	1	1	1	1	1	-	-	0
`fulleq`	1	1	1	1	1	1	1	1	1	0	1
`medeq`	0	1	1	0	1	-	-	0	-	-	0
`simpleq`	0	0	1	0	0	-	-	0	-	-	0

Note that there is no $\gamma_{L,1}$, whose computation would be rather different. Note, in addition, that `simpleq` and `medeq` coincide with their definitions in (2.1.1). The method used to solve this equation is very different from `easyeq`, and is significantly longer to run.

2.2.1. Usage

Unless otherwise noted, this method takes the following parameters (specified via the `[-p params]` flag, as a '`;`' separated list of entries).

- `rho` (`Float64`, default: 10^{-6}): density ρ.
- `tolerance` (`Float64`, default: 10^{-11}): maximal size of final step in Newton iteration.
- `maxiter` (`Int64`, default: 21): maximal number of iterations of the Newton algorithm before giving up.
- `P` (`Int64`, default: 11): order of all Chebyshev polynomial expansions (denoted by P below).
- `N` (`Int64`, default: 12): order of all Gauss quadratures (denoted by N below).
- `J` (`Int64`, default: 10): number of splines (denoted by J below).
- `nlrho_init` (`Int64`, default: 0): we initialize the Newton algorithm using the solution of `medeq`, computed using the methods in `easyeq`. If `nlrho_init` is $\neq 0$, then the solution of `medeq` is first computed for `nlrho` smaller values of ρ starting from $10^{\texttt{minlrho\_init}}$. This is useful when ρ is too large for the solution of `medeq` to be computed directly.
- `minlrho_init` (`Float64`, default: -6): see `nlrho_init`.
- `aK`, `bK`, `gK`, `aL1`, `bL1`, `aL2`, `bL2`, `gL2`, `aL3`, `bL3`, `gL3` (`Float64`, default: 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0): the values of α_K, β_K, γ_K, $\alpha_{L,1}$, $\beta_{L,1}$, $\alpha_{L,2}$, $\beta_{L,2}$, $\gamma_{L,2}$, $\alpha_{L,3}$, $\beta_{L,3}$, $\gamma_{L,3}$.
- `eq` (`String`, default: "`bigeq`", acceptable values: "`compleq`", "`bigeq`", "`fulleq`", "`medeq`", "`simpleq`"): A shortcut to select any of the equations defined in the table above. When this option is set, none of `aK`, `bK`, `gK`, `aL1`, `bL1`, `aL2`, `bL2`, `gL2`, `aL3`, `bL3`, `gL3` should be set.

The available `commands` are the following.

- `energy`: compute the energy e at a given ρ.
  Output: $[e]$ [Newton error ϵ].
- `energy_rho`: compute the energy e as a function of ρ. The Newton algorithm is initialized with the solution of `medeq`.
  Disabled parameters: `rho`.
  Extra parameters:
  - `minlrho` (`Float64`, default: 10^{-6}): minimal value for $\log_{10}\rho$.
  - `maxlrho` (`Float64`, default: 10^2): maximal value for $\log_{10}\rho$.
  - `nlrho` (`Int64`, default: 100): number of values for ρ (spaced logarithmically).
  - `minrho` (`Float64`, default: 10^{-6}): minimal value for ρ.
  - `maxrho` (`Float64`, default: 10^2): maximal value for ρ.
  - `nrho` (`Int64`, default: 0): number of values for ρ (spaced linearly). If `nrho` is $\neq 0$, then the linear spacing will be used, and `minlrho`, `maxlrho`, `nlrho` will be ignored. Otherwise, the logarithmic spacing will be used and `minrho`, `maxrho` will be ignored.
  - `rhos` (`Array{Float64}`): list of values for ρ, specified as a '`,`' separated list. This parameter takes precedence over `minlrho`, `maxlrho`, `nlrho`, `minrho`, `maxrho`, `nrho`.

  Output (one line for each value of ρ): $[\rho]$ $[e]$ [Newton error ϵ].
  Multithread support: yes, different values of ρ split up among workers.

- `energy_rho_init_prevrho`: compute the energy e as a function of ρ. The Newton algorithm is initialized with the solution of `medeq` for the lowest ρ, and with the previously computed ρ for the larger densities.
  Disabled parameters: same as `energy_rho`.
  Extra parameters: same as `energy_rho`.
  Output (one line for each value of ρ): [ρ] [e] [Newton error ϵ].
- `energy_rho_init_nextrho`: same as `energy_rho_init_prevrho` except that the energy is computed for decreasing densities instead of increasing ones. The Newton algorithm is initialized with the solution of `medeq` for the largest ρ, and with the previously computed ρ for the smaller densities.
  Disabled parameters: same as `energy_rho`.
  Extra parameters: same as `energy_rho`.
  Output (one line for each value of ρ): [ρ] [e] [Newton error ϵ].
- `condensate_fraction`: compute the uncondensed fraction η at a given ρ.
  Output: [η] [Newton error ϵ].
- `condensate_fraction_rho`: compute the uncondensed fraction η as a function of ρ. The Newton algorithm is initialized with the solution of `medeq`.
  Disabled parameters: same as `energy_rho`.
  Extra parameters: same as `energy_rho`.
  Output (one line for each value of ρ): [ρ] [η] [Newton error ϵ].
  Multithread support: yes, different values of ρ split up among workers.
- `uk`: compute the Fourier transform $\hat{u}(|k|)$. The values $|k|$ at which $\hat{u}$ is computed are those coming from the Gauss quadratures, and cannot be set.
  Output (one line for each value of $|k|$): [$|k|$] [$\hat{u}(|k|)$]
- `ux`: compute u as a function of $|x|$.
  Extra parameters:
  - `xmin` (`Float64`, default: 0): minimum of the range of $|x|$ to be printed.
  - `xmax` (`Float64`, default: 100): maximum of the range of $|x|$ to be printed.
  - `nx` (`Int64`, default: 100): number of points to print (linearly spaced).

  Output (one line for each value of x): [$|x|$] [$u(|x|)$]
- `2pt`: compute the spherically averaged two-point correlation function $C_2(|x|)$ at a given ρ.
  Extra parameters: same as `ux`, plus
  - `window_L` (`Float64`, default: 10^3): size of the Hann window used to numerically invert the Fourier transform in the computation of the tow-point correlation function, see (2.1.61).

  Output (one line for each value of $|x|$): [$|x|$] [$C_2(|x|)$]
- `2pt_max`: compute the maximum of the spherically averaged two-point correlation function $C_2(|x|)$ at a given ρ.
  Extra parameters: `window_L` plus
  - `dx` (`Float64`, default: 10^{-7}): step used to numerically approximate derivatives.
  - `x0` (`Float64`, default: 1): initial guess for the maximum is $\rho^{-1/3}$`x0`.
  - `maxstep` (`Float64`, default: ∞): maximal size of single step in maximization algorithm.
  - `tolerance_max` (`Float64`, default: `tolerance`): same as `tolerance`, used for the Newton algorithm underlying the maximization algorithm.

Output: $[|x_{\max}|]$ $[C_2(|x_{\max}|)]$

- `2pt_max_rho`: compute the maximum of the spherically averaged two-point correlation function $C_2(|x|)$ for a range of ρ.
  Extra parameters: same as `anyeq_2pt_max` plus those of `energy_rho`.
  Output (one line for each value of ρ): $[\rho]$ $[|x_{\max}|]$ $[C_2(|x_{\max}|)]$
  Multithread support: yes, different values of ρ split up among workers.

- `2pt_fourier`: compute the spherically averaged Fourier transform of the two-point correlation function $\hat{C}_2(|k|)$ at a given ρ.
  Extra parameters:

  - `kmin` (`Float64`, default: 0): minimum of the range of $|k|$ to be printed.
  - `kmax` (`Float64`, default: 10): maximum of the range of $|k|$ to be printed.
  - `nk` (`Int64`, default: 100): number of $|k|$'s to be printed.
  - `window_L` (`Float64`, default: 1000): what is actually computed is $\hat{C}_2$ convolved with a Gaussian of variance $1/\sqrt{L}$ where $L = \sqrt{\texttt{window\_L}}$, see (2.2.120).

  Output (one line for each value of $|k|$): $[|k|]$ $[\hat{C}_2(|k|)]$.
  Multithread support: yes, different values of k are split up among workers.

- `2pt_fourier_max`: compute the maximum of the spherically averaged Fourier transformed two-point correlation function $\hat{C}_2(|k|)$.
  Extra parameters: `window_L`, `maxstep` plus

  - `dk` (`Float64`, default: 10^{-7}): step used to numerically approximate derivatives.
  - `k0` (`Float64`, default: 1): initial guess for the maximum is $\rho^{1/3}$`k0`.

  Output: $[|k_{\max}|]$ $[\hat{C}_2(|k_{\max}|)]$

- `2pt_fourier_max_rho`: compute the maximum of the spherically averaged Fourier transformed two-point correlation function $\hat{C}_2(|k|)$ for a range of ρ.
  Extra parameters: same as those of `2pt_fourier_max` plus those of `energy_rho`.
  Output (one line for each value of ρ): $[\rho]$ $[|k_{\max}|]$ $[\hat{C}_2(|k_{\max}|)]$
  Multithread support: yes, different values of ρ split up among workers.

- `momentum_distribution`: compute the momentum distribution $\mathcal{M}(|k|)$ at a given ρ. The momentum distribution is computed for $k = k_{l,j}$ (see (2.2.40)).
  Extra parameters:

  - `kmin` (`Float64`, default: 0): minimum of the range of $|k|$ to be printed.
  - `kmax` (`Float64`, default: 10): maximum of the range of $|k|$ to be printed.
  - `window_L` (`Float64`, default: 1000): what is actually computed is $\mathfrak{M}_k$ convolved with a Gaussian of variance $1/\sqrt{L}$ where $L = \sqrt{\texttt{window\_L}}/k^2$, see (2.2.143).

  Output (one line for each value of $|k|$): $[|k|]$ $[\mathcal{M}(|k|)]$

- `compressibility_rho`: compute the compressibility χ as a function of ρ. The Newton algorithm is initialized with the solution of `medeq`. Disabled parameters: same as `energy_rho`.
  Extra parameters: same as `energy_rho`.
  Output (one line for each value of ρ): $[\rho]$ $[\chi]$.

- `save_Abar`: compute the matrix $\bar{A}$. This matrix is used to compute observables for `compleq`. This command is useful to output the value of $\bar{A}$ to a file once and for all, and use this file to run commands without recomputing $\bar{A}$.

Disabled parameters: `rho`, `tolerance`, `maxiter`, `minlrho_init`, `nlrho_init`.
Output: $[\bar{A}]$ (the output is not designed to be human-readable; it is obtained through nested `for` loops; for details, see the code).
Multithread support: yes, the first indices are split up among workers, which produces NJ jobs.

2.2.2. Description

2.2.2.1. Fourier space formulation

The computation is carried out in Fourier space. We take the convention that the Fourier transform of a function $f(|x|)$ is

$$\hat{f}(|k|) = \int_{\mathbb{R}^3} dx\ e^{ikx} f(|x|) = \frac{4\pi}{|k|} \int_0^\infty dr\ \frac{\sin(|k|r)}{r} f(r). \tag{2.2.8}$$

We define a Fourier-space convolution:

$$\hat{f} \hat{*} \hat{g}(|k|) := \int_{\mathbb{R}^3} \frac{dp}{(2\pi)^3}\ \hat{f}(|k-p|)\hat{g}(|p|). \tag{2.2.9}$$

In Fourier space, (2.2.1) becomes

$$k^2 \hat{u}(|k|) = \hat{S}(|k|) - 2\rho\hat{K}(|k|) + \rho^2 \hat{L}(|k|) \tag{2.2.10}$$

with

$$\hat{S}(|k|) = \int_{\mathbb{R}^3} dx\ e^{ikx}(1-u(|x|))v(|x|) = \hat{v}(k) - \hat{u}\hat{*}\hat{v}(k) \tag{2.2.11}$$

$$\rho\hat{K} = \gamma_K(\beta_K\rho\hat{S} + (1-\beta_K)2e)\hat{u} - \gamma_K\alpha_K\hat{u}\hat{*}((\beta_K\rho\hat{S} + (1-\beta_K)2e)\hat{u}) \tag{2.2.12}$$

$$\rho\hat{L}_1 = (\beta_{L,1}\rho\hat{S} + (1-\beta_{L,1})2e)\hat{u}^2 - \alpha_{L,1}\hat{u}\hat{*}((\beta_{L,1}\rho\hat{S} + (1-\beta_{L,1})2e)\hat{u}^2) \tag{2.2.13}$$

$$\rho\hat{L}_2 = -\gamma_{L,2}(\beta_{L,2}2\rho(\hat{u}\hat{*}(\hat{u}\hat{S}))+(1-\beta_{L,2})4e\hat{u}\hat{*}\hat{u})\hat{u}+\gamma_{L,2}\alpha_{L,2}\hat{u}\hat{*}((\beta_{L,2}2\rho(\hat{u}\hat{*}(\hat{u}\hat{S}))+(1-\beta_{L,2})4e\hat{u}\hat{*}\hat{u})\hat{u}) \tag{2.2.14}$$

$$\rho\hat{L}_3 = \gamma_{L,3}(1-\beta_{L,3})e(\hat{u}\hat{*}\hat{u})^2 - \gamma_{L,3}\alpha_{L,3}(1-\beta_{L,3})\hat{u}\hat{*}(e(\hat{u}\hat{*}\hat{u})^2) + \gamma_{L,3}\beta_{L,3}\hat{l}_3 - \frac{1}{2\rho}\gamma_{L,3}\alpha_{L,3}\beta_{L,3}\hat{u}\hat{*}\hat{l}_3 \tag{2.2.15}$$

with

$$\hat{l}_3(|k|) := \frac{1}{\rho^2}\int \frac{dq}{(2\pi)^3}\frac{dq'}{(2\pi)^3}\ \rho\hat{u}(|k-q'|)\rho\hat{u}(|q|)\rho\hat{u}(|q'|)\rho\hat{u}(|k-q|)\hat{S}(|q'-q|) \tag{2.2.16}$$

Therefore,

$$\Omega(u) = 0 \tag{2.2.17}$$

with

$$\Omega(u) := \rho^2\hat{u}(|k|)^2\sigma_{L,1}(|k|) - \left(\frac{k^2}{\rho} + 2\sigma_K(|k|) + 2f_1(|k|)\right)\rho\hat{u}(|k|) + \left(\hat{S}(|k|) + \frac{1}{2}f_2(|k|) + G(|k|)\right) \tag{2.2.18}$$

with

$$\sigma_K := \frac{1}{\rho}\gamma_K(\beta_K\rho\hat{S} + (1-\beta_K)2e), \quad \sigma_{L,1} := \frac{1}{\rho}(\beta_{L,1}\rho\hat{S} + (1-\beta_{L,1})2e) \tag{2.2.19}$$

$$f_1 := \frac{1}{\rho^2}\gamma_{L,2}(\beta_{L,2}\rho(\rho\hat{u}\hat{*}(\rho\hat{u}\hat{S})) + (1-\beta_{L,2})2e\rho\hat{u}\hat{*}\rho\hat{u}) \tag{2.2.20}$$

$$f_2 := \gamma_{L,3}(1-\beta_{L,3})\frac{2e}{\rho^3}(\rho\hat{u}\hat{*}\rho\hat{u})^2 + \gamma_{L,3}\beta_{L,3}\hat{l}_3 \tag{2.2.21}$$

$$G := \frac{2}{\rho^2}\gamma_K\alpha_K\rho\hat{u}\hat{*}((\beta_K\rho\hat{S} + (1-\beta_K)2e)\rho\hat{u}) - \frac{1}{\rho^2}\alpha_{L,1}\rho\hat{u}\hat{*}((\beta_{L,1}\rho\hat{S} + (1-\beta_{L,1})2e)\rho\hat{u}^2) + \frac{2}{\rho}\alpha_{L,2}\rho\hat{u}\hat{*}(f_1\rho\hat{u}) - \frac{1}{2\rho}\alpha_{L,3}\rho\hat{u}\hat{*}f_2. \tag{2.2.22}$$

Therefore,

$$\rho\hat{u} = 1 + \xi - \sqrt{(1+\xi)^2 - (1+\zeta)} \tag{2.2.23}$$

with

$$\xi := \frac{1}{\sigma_{L,1}}\left(\sigma_K - \sigma_{L,1} + \frac{k^2}{2\rho} + f_1\right), \quad \zeta := \frac{1}{\sigma_{L,1}}\left(\hat{S} - \sigma_{L,1} + \frac{1}{2}f_2 + G\right) \tag{2.2.24}$$

We rewrite (2.2.23) as

$$U = \frac{1+\zeta}{2(\xi+1)}\ \Phi\left(\tfrac{1+\zeta}{(\xi+1)^2}\right) \tag{2.2.25}$$

with

$$U := \rho\hat{u} \tag{2.2.26}$$

$$\Phi(x) := \frac{2(1-\sqrt{1-x})}{x}. \tag{2.2.27}$$

2.2.2.2. Evaluating integrals

To evaluate integrals numerically, we will split integration intervals over splines and use Gauss-Legendre quadratures. More specifically, to compute integrals of the form

$$\int \frac{dk}{(2\pi)^3}\ f(U(k),k) = \frac{1}{2\pi^2}\int_0^\infty dt\ t^2 f(U(t),t) \tag{2.2.28}$$

we first compactify space to $(-1,1]$ by changing variables to $\tau = \frac{1-t}{1+t}$:

$$\int \frac{dk}{(2\pi)^3}\ f(U(k),k) = \frac{1}{\pi^2}\int_{-1}^1 d\tau\ \frac{(1-\tau)^2}{(1+\tau)^4} f(U(\tfrac{1-\tau}{1+\tau}), \tfrac{1-\tau}{1+\tau}). \tag{2.2.29}$$

We then split $(-1,1]$ into J sub-intervals (given by the parameter `J`), called *splines*:

$$(-1,1] = \bigcup_{j=0}^{J-1}(\tau_j, \tau_{j+1}]. \tag{2.2.30}$$

The τ are taken to be equally spaced, but the code is designed in such a way that this could be changed easily in the future:

$$\tau_j = -1 + \frac{2j}{J}. \tag{2.2.31}$$

In these terms,

$$\int \frac{dk}{(2\pi)^3}\ f(U(k),k) = \sum_{l=0}^{J-1}\int_{\tau_l}^{\tau_{l+1}} d\tau\ \frac{(1-\tau)^2}{(1+\tau)^4} g(U(\tfrac{1-\tau}{1+\tau}, \tfrac{1-\tau}{1+\tau}). \tag{2.2.32}$$

We then change variables to r:

$$\tau = -\frac{\tau_{l+1}-\tau_l}{2}\sin(\tfrac{\pi r}{2}) + \frac{\tau_{l+1}+\tau_l}{2} =: \frac{1-\vartheta_l(r)}{1+\vartheta_l(r)} \tag{2.2.33}$$

(the reason for this specific change of variables will become clear at the end of this paragraph and in the next one) and find

$$\int \frac{dk}{(2\pi)^3}\ f(U(k),k) = \sum_{l=0}^{J-1} \frac{\tau_{l+1}-\tau_l}{16\pi} \int_{-1}^{1} dr\ \cos(\tfrac{\pi r}{2})(1+\vartheta_l(r))^2\vartheta_l^2(r) f(U(\vartheta_l(r)),\vartheta_l(r)). \tag{2.2.34}$$

We then approximate the integral using a Gauss-Legendre quadrature of order N (see appendix-A2):

$$\int \frac{dk}{(2\pi)^3}\ f(k) \approx \sum_{l=0}^{J-1} \frac{\tau_{l+1}-\tau_l}{16\pi} \sum_{j=1}^{N} w_j \cos(\tfrac{\pi x_j}{2})(1+\vartheta_l(x_j))^2\vartheta_l^2(x_j) f(\mathbb{U}_{l,j},\vartheta_l(x_j)) \tag{2.2.35}$$

with

$$\mathbb{U}_{l,j} := U(k_{l,j}), \quad k_{l,j} := \frac{2+(\tau_{l+1}-\tau_l)\sin(\frac{\pi x_j}{2}) - (\tau_{l+1}+\tau_\alpha)}{2-(\tau_{l+1}-\tau_l)\sin(\frac{\pi x_j}{2}) + (\tau_{l+1}+\tau_l)}. \tag{2.2.36}$$

The choice of the change of variables (2.2.33) is made so that U is evaluated at $k_{l,j}$, which are the momenta that appear naturally in the Chebyshev polynomial expansion below (2.2.40). In this way, we can compute arbitrary integrals of functions of U by just computing the values of $\mathbb{U}_{l,j}$.

2.2.2.3. Chebyshev polynomial expansion

In the case of `easyeq`, we saw that using Gauss quadratures reduced the computation to evaluating U at a fixed set of discrete momenta. This is not the case here, due to the presence of more complicated convolutions of U. Instead, we will approximate U using polynomial approximations. We will now discuss this approximation for an arbitrary function $a(|k|)$, as we will need it for other functions than simply U.

1 - As in the previous paragraph, we compactify $[0,\infty)$ to $(-1,1]$ via the change of variables $r \mapsto \frac{1-r}{1+r}$, and split the interval into splines. Inside each spline, we use a Chebyshev polynomial expansion. However, we need to be careful in the first spline, which encapsulates the behavior at $|k| \to \infty$: U decays to 0 as $|k| \to \infty$, and this behavior cannot be reproduced by a polynomial expansion. To avoid this, if $a \sim |k|^{-\nu_a}$, we expand $|k|^{\nu_a}a$ instead of a. Actually, to simplify expressions in the presence of the compactification, we will expand $2^{-\nu_a}(1+|k|)^{\nu_a}a$, which makes no conceptual difference. In the case of $a = U$, we will use $\nu = 2$, which we know holds for the Simple equation [CJL20]. Putting all this together, we write, for $\tau \in (-1,1]$, if $|k| \equiv \frac{1-\tau}{1+\tau}$,

$$2^{-\nu_a}(1+|k|)^{\nu_a}a(|k|) \equiv \frac{1}{(1+\tau)^{\nu_a}} a(\tfrac{1-\tau}{1+\tau}) = \sum_{l=0}^{J-1} \mathbb{1}_{\tau_l<\tau\leqslant\tau_{l+1}} \sum_{n=0}^{\infty} \mathbf{F}_{l,n}^{(\nu_a)}(a) T_n(\tfrac{2\tau-(\tau_l+\tau_{l+1})}{\tau_{l+1}-\tau_l}) \tag{2.2.37}$$

in which $\mathbb{1}_{\tau_l<\tau\leqslant\tau_{l+1}} \in \{0,1\}$ is equal to 1 if and only if $\tau_l < \tau \leqslant \tau_{l+1}$, and

$$\mathbf{F}_{l,n}^{(\nu_a)}(a) := \frac{2-\delta_{n,0}}{\pi} \int_0^\pi d\theta\ \frac{\cos(n\theta)}{(1+\frac{\tau_{l+1}-\tau_l}{2}\cos(\theta)+\frac{\tau_{l+1}+\tau_l}{2})^{\nu_a}} a(\tfrac{2-(\tau_{l+1}-\tau_l)\cos(\theta)-(\tau_{l+1}+\tau_\alpha)}{2+(\tau_{l+1}-\tau_l)\cos(\theta)+(\tau_{l+1}+\tau_l)}) \tag{2.2.38}$$

(see appendix A1 for a discussion of the Chebyshev polynomial expansion and the error of its truncations).

2 - In order to compute an approximation for a using (2.2.37), we will truncate the sum over n to a finite value P (given by the parameter `P`). In addition, to compute the integral in (2.2.38), we will use a Gauss-Legendre quadrature of order N (given by the parameter `N`), see appendix A2:

$$\mathbf{F}_{l,n}^{(\nu_a)}(a) \approx \mathfrak{F}_{l,n}^{(\nu_a)}(\mathfrak{a}) := \frac{2-\delta_{n,0}}{2} \sum_{j=1}^{N} w_j \cos(\tfrac{n\pi(1+x_j)}{2}) \frac{\mathfrak{a}_{l,j}}{(1-\frac{\tau_{l+1}-\tau_l}{2}\sin(\frac{\pi x_j}{2})+\frac{\tau_{l+1}+\tau_l}{2})^{\nu_a}} \tag{2.2.39}$$

with

$$\mathfrak{a}_{l,j} := a(k_{l,j}), \quad k_{l,j} := \frac{2 + (\tau_{l+1} - \tau_l)\sin(\frac{\pi x_j}{2}) - (\tau_{l+1} + \tau_\alpha)}{2 - (\tau_{l+1} - \tau_l)\sin(\frac{\pi x_j}{2}) + (\tau_{l+1} + \tau_l)} \tag{2.2.40}$$

and (x_j, w_j) are the abcissa and weights for Gauss-Legendre quadratures.

3 - All in all, we approximate

$$a(\tfrac{1-\tau}{1+\tau}) \approx (1+\tau)^{\nu_a} \sum_{l=0}^{J-1} \mathbb{1}_{\tau_l < \tau \leqslant \tau_{l+1}} \sum_{n=0}^{P} \mathfrak{F}_{l,n}^{(\nu_a)}(\mathfrak{a}) T_n(\tfrac{2\tau - (\tau_l + \tau_{l+1})}{\tau_{l+1} - \tau_l}) \tag{2.2.41}$$

with $\mathfrak{F}$ defined in (2.2.39). Furthermore, using the Chebyshev polynomial expansion and Gauss-Legendre quadratures, we can compute all the observables we are interested in by computing $\mathbb{U}_{l,j} \equiv U(k_{l,j})$. With this in mind, we will represent the function U as a vector of dimension NJ whose components are $\mathbb{U}_{l,j}$.

2.2.2.4. Convolutions

Using the Chebyshev polynomial approximation, we can compute convolutions as follows.

1 - First, we rewrite the convolution as a two-dimensional integral, using bipolar coordinates (see lemma A3.1):

$$(a \hat{*} b)(|k|) = \frac{1}{4\pi^2 |k|} \int_0^\infty dt\ t a(t) \int_{||k|-t|}^{|k|+t} ds\ s b(s) \tag{2.2.42}$$

We change variables to compactify the integrals $\tau = \frac{1-t}{1+t}$, $\sigma = \frac{1-s}{1+s}$:

$$(a \hat{*} b)(|k|) = \frac{1}{4\pi^2 |k|} \int_{-1}^{1} d\tau\ \frac{2(1-\tau)}{(1+\tau)^3} a(\tfrac{1-\tau}{1+\tau}) \int_{\alpha_-(|k|,\tau)}^{\alpha_+(|k|,\tau)} d\sigma\ \frac{2(1-\sigma)}{(1+\sigma)^3} b(\tfrac{1-\sigma}{1+\sigma}) \tag{2.2.43}$$

with

$$\alpha_-(|k|,\tau) := \frac{1 - |k| - \frac{1-\tau}{1+\tau}}{1 + |k| + \frac{1-\tau}{1+\tau}}, \quad \alpha_+(|k|,\tau) := \frac{1 - ||k| - \frac{1-\tau}{1+\tau}|}{1 + ||k| - \frac{1-\tau}{1+\tau}|}. \tag{2.2.44}$$

Therefore, using the approximation (2.2.41), if $\mathfrak{a}_{l,j} := a(k_{l,j})$ and $\mathfrak{b}_{l,j} := b(k_{l,j})$ (2.2.40),

$$(a \hat{*} b)(|k_{l,j}|) \approx (\mathfrak{a} \odot \mathfrak{b})_{l,j} := \sum_{n,m=0}^{P} \sum_{l',l''=0}^{J-1} \mathfrak{F}_{l',n}^{(\nu_a)}(\mathfrak{a}) \mathfrak{F}_{l'',m}^{(\nu_b)}(\mathfrak{b}) A_{l',n;l'',m}^{(\nu_a,\nu_b)}(|k_{l,j}|) \tag{2.2.45}$$

with

$$\begin{aligned} A_{l',n;l'',m}^{(\nu_a,\nu_b)}(|k|) := \frac{1}{4\pi^2 |k_{l,j}|} \int_{\tau_{l'}}^{\tau_{l'+1}} d\tau\ \frac{2(1-\tau)}{(1+\tau)^{3-\nu_a}} T_n(\tfrac{2\tau - (\tau_{l'} + \tau_{l'+1})}{\tau_{l'+1} - \tau_{l'}}) \cdot \\ \cdot \mathbb{1}_{\alpha_-(|k|,\tau) < \tau_{l''+1}} \mathbb{1}_{\alpha_+(|k|,\tau) > \tau_{l''}} \int_{\max(\tau_{l''}, \alpha_-(|k|,\tau))}^{\min(\tau_{l''+1}, \alpha_+(|k|,\tau))} d\sigma\ \frac{2(1-\sigma)}{(1+\sigma)^{3-\nu_b}} T_m(\tfrac{2\sigma - (\tau_{l''} + \tau_{l''+1})}{\tau_{l''+1} - \tau_{l''}}) \end{aligned} \tag{2.2.46}$$

(we need the indicator functions to ensure that the bounds of the integral are correct). Note that A is independent of a and b, and can be computed once and for all at the beginning of execution for all values of $k_{l,j}$ (2.2.40). We then compute the integrals using Gauss-Legendre quadratures (as is proved in the next paragraph, the integrand is non singular provided $\nu_a, \nu_b \geqslant 2$).

2 - Note that these integrals are not singular as long as $\nu_a, \nu_b \geqslant 2$: indeed (since the only possible problems occur at -1, it suffices to consider the case with only one spline), $\alpha_-, \alpha_+ > -1$ for $\tau > -1$, and

$$\alpha_\pm(k,\tau) = -1 + (1+\tau) \pm \frac{k}{2}(1+\tau)^2 + O(1+\tau)^3 \tag{2.2.47}$$

and

$$\int_{\alpha_-(|k|,\tau)}^{\alpha_+(|k|,\tau)} d\sigma \ \left|\frac{2(1-\sigma)}{(1+\sigma)^{3-\nu_b}}T_m(\sigma)\right| \leqslant 4\int_{\alpha_-(|k|,\tau)}^{\alpha_+(|k|,\tau)} d\sigma \ \frac{1}{(1+\sigma)^{3-\nu_b}} \tag{2.2.48}$$

which, if $\nu_b = 2$, yields

$$4\log\left(\frac{1+\alpha_+(|k|,\tau)}{1+\alpha_-(|k|,\tau)}\right) = 4|k|(1+\tau) + O(1+\tau)^2 \tag{2.2.49}$$

and if $\nu_b > 2$, yields

$$\frac{4}{\nu_b-2}\left((1+\alpha_+(|k|,\tau))^{\nu_b-2} - (1+\alpha_-(|k|,\tau))^{\nu_b-2}\right) = \frac{4}{\nu_b-2}|k|(1+\tau)^{\nu_b-1}(1+O(1+\tau)). \tag{2.2.50}$$

Therefore,

$$\left|\frac{2(1-\tau)}{(1+\tau)^{3-\nu_a}}T_n(\tau)\int_{\alpha_-(|k|,\tau)}^{\alpha_+(|k|,\tau)} d\sigma \ \frac{2(1-\sigma)}{(1+\sigma)^{3-\nu_b}}T_m(\sigma)\right| \leqslant \frac{8}{c_{\nu_b}}|k|(1-\tau)(1+\tau)^{\nu_a+\nu_b-4}(1+O(1+\tau)) \tag{2.2.51}$$

where $c_2 := 1$ and $c_{\nu_b} := \nu_b - 2$ if $\nu_b > 2$. The integrand is, therefore, not singular as long as $\nu_a, \nu_b \geqslant 2$.

3 - Evaluating convolutions at $k = 0$ is not immediate, as the formula for $A(0)$ involves a bit of a computation. To compute $A(0)$, we expand

$$\alpha_-(|k|,\tau) = \tau - \frac{|k|(1+\tau)^2}{2} + O(k^2), \quad \alpha_+(|k|,\tau) = \tau + \frac{|k|(1+\tau)^2}{2} + O(k^2) \tag{2.2.52}$$

so

$$A^{(\nu_a,\nu_b)}_{\zeta,n;\zeta',m}(0) = \mathbb{1}_{\zeta=\zeta'}\frac{1}{\pi^2}\int_{\tau_\zeta}^{\tau_{\zeta+1}} d\tau \ \frac{(1-\tau)^2}{(1+\tau)^{4-\nu_a-\nu_b}}T_n(\tfrac{2\tau-(\tau_\zeta+\tau_{\zeta+1})}{\tau_{\zeta+1}-\tau_\zeta})T_m(\tfrac{2\tau-(\tau_\zeta+\tau_{\zeta+1})}{\tau_{\zeta+1}-\tau_\zeta}) \tag{2.2.53}$$

which we then compute using a Gauss-Legendre quadrature. We can then evaluate convolutions at $k = 0$:

$$(a\hat{*}b)(0) \approx \langle \mathfrak{a}\mathfrak{b}\rangle := \frac{1}{4\pi^2|k_{l,j}|}\sum_{n,m=0}^{P}\sum_{l,l'=0}^{J-1}\mathfrak{F}^{(\nu_a)}_{l,n}(\mathfrak{a})\mathfrak{F}^{(\nu_b)}_{l',m}(\mathfrak{b})A^{(\nu_a,\nu_b)}_{l,n;l',m}(0) \tag{2.2.54}$$

4 - Let us now compute some choices of $\mathfrak{a}, \mathfrak{b}$ more explicitly.

4-1 - Let us start with $\mathbb{1}_{l,n}\hat{*}a$ where $\mathbb{1}_{l,n}$ is the vector which has 0's everywhere except at position (l,n). We have

$$\mathfrak{F}^{(\nu_1)}_{l',m}(\mathbb{1}_{l,n}) = \delta_{l',l}\frac{2-\delta_{m,0}}{2}w_n\frac{\cos(\frac{m\pi(1+x_n)}{2})}{(1-\frac{\tau_{l+1}-\tau_l}{2}\sin(\frac{\pi x_j}{2})+\frac{\tau_{l+1}+\tau_l}{2})^{\nu_1}} \tag{2.2.55}$$

so

$$(\mathbb{1}_{n,l''}\odot\mathfrak{a})_{m,l} = \sum_{l,p}\sum_{l'}\frac{2-\delta_{l,0}}{2}w_n\frac{\cos(\frac{l\pi(1+x_n)}{2})}{(1-\frac{\tau_{l''+1}-\tau_{l''}}{2}\sin(\frac{\pi x_n}{2})+\frac{\tau_{l''+1}+\tau_{l''}}{2})^{\nu_1}}A^{(\nu_1,\nu_a)}_{l'',l;l',p}(k_{l,m})\mathfrak{F}^{(\nu_a)}_{l',p}(\mathfrak{a}). \tag{2.2.56}$$

4-2 - We now turn to $a\hat{*}\hat{v}$. Let

$$\Upsilon(t,|k|) := \int_{||k|-t|}^{|k|+t} ds \ \frac{s\hat{v}(s)}{|k|} \tag{2.2.57}$$

We have

$$a\hat{*}\hat{v} = \frac{1}{4\pi^2}\int_0^\infty dt \ ta(t)\Upsilon(t,|k|). \tag{2.2.58}$$

Following the integration scheme (2.2.35),

$$(\mathfrak{a}\odot\mathfrak{v})_{l,i}=\frac{1}{32\pi}\sum_{l'=0}^{J-1}(\tau_{l'+1}-\tau_{l'})\sum_{j=1}^{N}w_j\cos(\tfrac{\pi x_j}{2})(1+\vartheta_{l'}(x_j))^2\vartheta_{l'}(x_j)\mathfrak{a}_{l',j}\Upsilon(\vartheta(x_j),k_{l,i}). \tag{2.2.59}$$

At $k=0$,

$$\langle\mathfrak{a}\mathfrak{v}\rangle=\frac{1}{32\pi}\sum_{l'=0}^{J-1}(\tau_{l'+1}-\tau_{l'})\sum_{j=1}^{N}w_j\cos(\tfrac{\pi x_j}{2})(1+\vartheta_{l'}(x_j))^2\vartheta_{l'}(x_j)\mathfrak{a}_{l',j}\Upsilon(\vartheta(x_j),0) \tag{2.2.60}$$

with

$$\Upsilon(t,0)=2tv(t). \tag{2.2.61}$$

The quantities $\Upsilon(\vartheta(x_n),k_{l,i})$ and $\Upsilon(\vartheta(x_n),0)$ are independent of a and can be computed once and for all at execution. The integral in (2.2.57) is computed using Gauss-Legendre quadratures, but without splitting into splines. To maintain a high precision, we set the order of the integration to `J`×`N`.

4-3 - Finally,

$$(\mathbb{1}_{l',n}\odot\mathfrak{v})_{l,i}=\frac{\tau_{l'+1}-\tau_{l'}}{32\pi}w_n\cos(\tfrac{\pi x_n}{2})(1+\vartheta_{l'}(x_n))^2\vartheta_{l'}(x_n)\Upsilon(\vartheta(x_n),k_{l,i}) \tag{2.2.62}$$

and

$$\langle\mathbb{1}_{l',n}\mathfrak{v}\rangle=\frac{\tau_{l'+1}-\tau_{l'}}{32\pi}w_n\cos(\tfrac{\pi x_n}{2})(1+\vartheta_{l'}(x_n))^2\vartheta_{l'}(x_n)\Upsilon(\vartheta(x_n),0). \tag{2.2.63}$$

2.2.2.5. Evaluating $\hat{l}_3$

The only term in (2.2.1) that does not involve just convolutions (whose computation was described above) is $\hat{l}_3$ (2.2.16). To evaluate it, we first change variables, using a generalization of bipolar coordinates (see lemma A3.2):

$$\hat{l}_3(|k|)=\frac{1}{(2\pi)^5\rho^2|k|^2}\int_0^\infty dt\int_{||k|-t|}^{|k|+t}ds\int_0^\infty dt'\int_{||k|-t'|}^{|k|+t'}ds'\int_0^{2\pi}d\theta\ sts't'U(s)U(t)U(s')U(t')\cdot \\ \cdot S(\mathfrak{R}(s,t,s',t',\theta,|k|)) \tag{2.2.64}$$

with

$$\mathfrak{R}(s,t,s',t',\theta,|k|):=\frac{1}{\sqrt{2}|k|}\Big(|k|^2(s^2+t^2+(s')^2+(t')^2)-|k|^4-(s^2-t^2)((s')^2-(t')^2) \\ -\sqrt{(4|k|^2s^2-(|k|^2+s^2-t^2)^2)(4|k|^2(s')^2-(|k|^2+(s')^2-(t')^2)^2)}\cos\theta\Big)^{\frac{1}{2}}. \tag{2.2.65}$$

We change variables as in (2.2.43):

$$\hat{l}_3(|k|)=\frac{1}{(2\pi)^5\rho^2|k|^2}\int_{-1}^1 d\tau\ \frac{2(1-\tau)}{(1+\tau)^3}U(\tfrac{1-\tau}{1+\tau})\int_{\alpha_-(|k|,\tau)}^{\alpha_+(|k|,\tau)}d\sigma\ \frac{2(1-\sigma)}{(1+\sigma)^3}U(\tfrac{1-\sigma}{1+\sigma})\int_{-1}^1 d\tau'\ \frac{2(1-\tau')}{(1+\tau')^3}\cdot \\ \cdot U(\tfrac{1-\tau'}{1+\tau'})\int_{\alpha_-(|k|,\tau')}^{\alpha_+(|k|,\tau')}d\sigma'\ \frac{2(1-\sigma')}{(1+\sigma')^3}U(\tfrac{1-\sigma'}{1+\sigma'})\int_0^{2\pi}d\theta\ S(\mathfrak{R}(\tfrac{1-\sigma}{1+\sigma},\tfrac{1-\tau}{1+\tau},\tfrac{1-\sigma'}{1+\sigma'},\tfrac{1-\tau'}{1+\tau'},\theta,|k|)). \tag{2.2.66}$$

We expand U and S into Chebyshev polynomials as in (2.2.37), and split the integrals into splines:

$$\hat{l}_3(|k_{l,j}|) \approx (\mathbb{U}^{\otimes 4} \otimes \mathbb{S})_{l,j} := \sum_{\lambda_1,\cdots,\lambda_5=0}^{J-1} \sum_{n_1,\cdots,n_5=1}^{\infty} \bar{A}^{(\nu)}_{\lambda_1,n_1;\cdots;\lambda_5,n_5}(|k_{l,j}|) \prod_{i=1}^{4} \left(\mathfrak{F}^{(\nu)}_{\lambda_i,n_i}(\mathbb{U})\right) \mathfrak{F}^{(\nu)}_{\lambda_5,n_5}(\mathbb{S}) \tag{2.2.67}$$

with $\mathbb{U}_{l,j} := U(k_{l,j})$, $\mathbb{S}_{l,j} := S(k_{l,j})$ and

$$\begin{aligned}
\bar{A}^{(\nu)}_{\lambda_1,n_1;\cdots;\lambda_5,n_5}(|k|) = {} & \frac{1}{(2\pi)^5\rho^2|k|^2} \int_{\tau_{\lambda_1}}^{\tau_{\lambda_1+1}} d\tau \; \frac{2(1-\tau)}{(1+\tau)^{3-\nu}} T_{n_1}\left(\tfrac{2\tau-(\tau_{\lambda_1}+\tau_{\lambda_1+1})}{\tau_{\lambda_1+1}-\tau_{\lambda_1}}\right)\cdot \\
& \cdot \mathbb{1}_{\alpha_-(|k|,\tau)<\tau_{\lambda_2+1}} \mathbb{1}_{\alpha_+(|k|,\tau)>\tau_{\lambda_2}} \int_{\max(\tau_{\lambda_2},\alpha_-(|k|,\tau))}^{\min(\tau_{\lambda_2+1},\alpha_+(|k|,\tau))} d\sigma \; \frac{2(1-\sigma)}{(1+\sigma)^{3-\nu}} T_{n_2}\left(\tfrac{2\sigma-(\tau_{\lambda_2}+\tau_{\lambda_2+1})}{\tau_{\lambda_2+1}-\tau_{\lambda_2}}\right)\cdot \\
& \cdot \int_{\tau_{\lambda_3}}^{\tau_{\lambda_3+1}} d\tau' \; \frac{2(1-\tau')}{(1+\tau')^{3-\nu}} T_{n_3}\left(\tfrac{2\tau'-(\tau_{\lambda_3}+\tau_{\lambda_3+1})}{\tau_{\lambda_3+1}-\tau_{\lambda_3}}\right)\cdot \\
& \cdot \mathbb{1}_{\alpha_-(|k|,\tau')<\tau_{\lambda_4+1}} \mathbb{1}_{\alpha_+(|k|,\tau')>\tau_{\lambda_4}} \int_{\max(\tau_{\lambda_4},\alpha_-(|k|,\tau'))}^{\min(\tau_{\lambda_4+1},\alpha_+(|k|,\tau'))} d\sigma' \; \frac{2(1-\sigma')}{(1+\sigma')^{3-\nu}} T_{n_4}\left(\tfrac{2\sigma'-(\tau_{\lambda_4}+\tau_{\lambda_4+1})}{\tau_{\lambda_4+1}-\tau_{\lambda_4}}\right)\cdot \\
& \cdot \int_0^{2\pi} d\theta \; \frac{2^\nu}{(2+\Re)^\nu} T_{n_5}\left(\tfrac{2\frac{1-\Re}{1+\Re}-(\tau_{\lambda_5}+\tau_{\lambda_5+1})}{\tau_{\lambda_5+1}-\tau_{\lambda_5}}\right) \mathbb{1}_{\tau_{\lambda_5}<\frac{1-\Re}{1+\Re}<\tau_{\lambda_5+1}}
\end{aligned} \tag{2.2.68}$$

in which $\Re$ is a shorthand for $\Re(\frac{1-\sigma}{1+\sigma}, \frac{1-\tau}{1+\tau}, \frac{1-\sigma'}{1+\sigma'}, \frac{1-\tau'}{1+\tau'}, \theta, |k|)$. Note that $\bar{A}$ is independent of U, and can be computed once and for all at the beginning of execution. Since the tensor $\bar{A}$ is quite large (it contains $(NJ)^5$ entries), and its computation can be rather long it can be inconvenient to compute $\bar{A}$ at every execution. Instead, one can use the `save_Abar` method to compute $\bar{A}$ and save it to a file, which can then be recalled via the `-s` option on the command line.

2.2.2.6. Main algorithm to compute U

We are now ready to detail how $U \equiv \rho\hat{u}$ is computed. All of the observables we are interested in are approximated from the values $\mathbb{U}_{l,j} := U(k_{l,j})$ (2.2.40).

1 - The equation for $(\mathbb{U}_{l,j})_{l\in\{0,\cdots,J-1\},j\in\{1,\cdots,N\}}$ is obtained by approximating (2.2.25) according to the prescriptions detailed above:

$$\mathbb{U}_{l,j} := \frac{1+\mathbb{Y}_{l,j}}{2(\mathbb{X}_{l,j}+1)} \; \Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right) \tag{2.2.69}$$

$$\mathbb{X}_{l,j} := \frac{1}{\mathbb{L}_{l,j}}\left(\mathbb{K}_{l,j} - \mathbb{L}_{l,j} + \frac{k_{l,j}^2}{2\rho} + \mathbb{I}_{l,j}\right), \quad \mathbb{Y}_{l,j} := \frac{1}{\mathbb{L}_{l,j}}\left(\mathbb{S}_{l,j} - \mathbb{L}_{l,j} + \frac{1}{2}\mathbb{J}_{l,j} + \mathbb{G}_{l,j}\right) \tag{2.2.70}$$

$$\mathbb{S}_{l,j} := \mathfrak{v}_{l,j} - \frac{1}{\rho}(\mathbb{U}\odot\mathfrak{v})_{l,j}, \quad \mathfrak{v}_{l,j} := \hat{v}(k_{l,j}), \quad \mathbb{E} := \hat{v}(0) - \frac{1}{\rho}\langle\mathbb{U}\mathfrak{v}\rangle \tag{2.2.71}$$

$$\mathbb{I}_{l,j} := \frac{1}{\rho}\gamma_{L,2}(\beta_{L,2}(\mathbb{U}\odot(\mathbb{U}\mathbb{S}))_{l,j} + (1-\beta_{L,2})\mathbb{E}(\mathbb{U}\odot\mathbb{U})_{l,j}) \tag{2.2.72}$$

$$\mathbb{K}_{l,j} := \gamma_K(\beta_K\mathbb{S}_{l,j} + (1-\beta_K)\mathbb{E}), \quad \mathbb{L}_{l,j} := \gamma_{L,1}(\beta_{L,1}\mathbb{S}_{l,j} + (1-\beta_{L,1})\mathbb{E}) \tag{2.2.73}$$

$$\mathbb{J}_{l,j} := \gamma_{L,3}(1-\beta_{L,3})\frac{\mathbb{E}}{\rho^2}(\mathbb{U}\odot\mathbb{U})^2_{l,j} + \gamma_{L,3}\beta_{L,3}(\mathbb{U}^{\otimes 4}\otimes\mathbb{S})_{l,j} \tag{2.2.74}$$

$$\mathbb{G}_{l,j} := \frac{2}{\rho}\gamma_K\alpha_K(\mathbb{U}\odot((\beta_K\mathbb{S}+(1-\beta_K)\mathbb{E})\mathbb{U}))_{l,j} \\ -\frac{1}{\rho}\alpha_{L,1}(\mathbb{U}\odot((\beta_{L,1}\mathbb{S}+(1-\beta_{L,1})\mathbb{E})\mathbb{U}^2))_{l,j}+\frac{2}{\rho}\alpha_{L,2}(\mathbb{U}\odot(\mathbb{I}\mathbb{U}))_{l,j}-\frac{1}{2\rho}\alpha_{L,3}(\mathbb{U}\odot\mathbb{J})_{l,j} \tag{2.2.75}$$

(see (2.2.45), (2.2.67) and (2.2.54) for the definitions of $\odot$, $\otimes$ and $\langle\cdot\rangle$).

2 - We rewrite (2.2.69) as a root finding problem:

$$\Xi_{l,j}(\mathbb{U}) := \mathbb{U}_{l,j} - \frac{1+\mathbb{Y}_{l,j}}{2(\mathbb{X}_{l,j}+1)}\,\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right) = 0 \tag{2.2.76}$$

which we solve using the Newton algorithm, that is, we define a sequence of $\mathbb{U}$'s:

$$\mathbb{U}^{(n+1)} = \mathbb{U}^{(n)} - (D\Xi(\mathbb{U}^{(n)}))^{-1}\Xi(\mathbb{U}^{(n)}) \tag{2.2.77}$$

where $D\Xi$ is the Jacobian of Ξ:

$$(D\Xi)_{l,j;l',i} := \frac{\partial\Xi_{l,j}}{\partial\mathbb{U}_{l',i}}. \tag{2.2.78}$$

3 - We initialize the algorithm with the solution of the Medium equation, which is computed using the `easyeq` method. However, `easyeq` only computes $\hat{u}$ at the momenta given by the Gauss-Legendre quadratures (2.1.17). To obtain a value for $\hat{u}$ at $k_{l,j}$, we use a linear interpolation (code is provided for a polynomial interpolation, which has not performed as well). The parameters `tolerance`, `maxiter` are passed on to `easyeq` as is, and the order of the Gauss-Legendre quadratures is set to `J`×`N`.

4 - We are left with computing the Jacobian of Ξ:

$$\partial_{l',i}\Xi_{l,j} \equiv \frac{\partial\Xi_{l,j}}{\partial\mathbb{U}_{l',i}} = \delta_{l,l'}\delta_{i,j} + \frac{1}{2(\mathbb{X}_{l,j}+1)}\left(\frac{(1+\mathbb{Y}_{l,j})\partial_{l',i}\mathbb{X}_{l,j}}{\mathbb{X}_{l,j}+1} - \partial_{l',i}\mathbb{Y}_{l,j}\right)\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right) \\ +\frac{(1+\mathbb{Y}_{l,j})}{2(\mathbb{X}_{l,j}+1)^3}\left(\frac{2(1+\mathbb{Y}_{l,j})}{\mathbb{X}_{l,j}+1}\partial_{l',i}\mathbb{X}_{l,j} - \partial_{l',i}\mathbb{Y}_{l,j}\right)\partial\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right) \tag{2.2.79}$$

with

$$\partial_{l',i}\mathbb{X}_{l,j} = \frac{1}{\mathbb{L}_{l,j}}\left(\partial_{l',i}\mathbb{K}_{l,j} - \partial_{l',i}\mathbb{L}_{l,j} + \partial_{l',i}\mathbb{I}_{l,j}\right) - \frac{\partial_{l',i}\mathbb{L}_{l,j}}{\mathbb{L}_{l,j}}\mathbb{X}_{l,j} \tag{2.2.80}$$

$$\partial_{l',i}\mathbb{Y}_{l,j} = \frac{1}{\mathbb{L}_{l,j}}\left(\partial_{l',i}\mathbb{S}_{l,j} - \partial_{l',i}\mathbb{L}_{l,j} + \frac{1}{2}\partial_{l',i}\mathbb{J}_{l,j} + \partial_{l',i}\mathbb{G}_{l,j}\right) - \frac{\partial_{l',i}\mathbb{L}_{l,j}}{\mathbb{L}_{l,j}}\mathbb{Y}_{l,j} \tag{2.2.81}$$

$$\partial_{l',i}\mathbb{S}_{l,j} = -\frac{1}{\rho}(\mathbb{1}_{l',i}\odot\mathfrak{v})_{l,j}, \quad \partial_{l',i}\mathbb{E} = -\frac{1}{\rho}\left\langle\mathbb{1}_{l',i}\mathfrak{v}\right\rangle \tag{2.2.82}$$

$$\partial_{l',i}\mathbb{K}_{l,j} := \gamma_K(\beta_K\partial_{l',i}\mathbb{S}_{l,j} + (1-\beta_K)\partial_{l',i}\mathbb{E}), \quad \partial_{l',i}\mathbb{L}_{l,j} := \gamma_{L,1}(\beta_{L,1}\partial_{l',i}\mathbb{S}_{l,j} + (1-\beta_{L,1})\partial_{l',i}\mathbb{E}) \tag{2.2.83}$$

$$\partial_{l',i}\mathbb{I}_{l,j} = \frac{1}{\rho}\gamma_{L,2}\left(\beta_{L,2}(\mathbb{1}_{l',i}\odot(\mathbb{U}\mathbb{S}) + \mathbb{U}\odot(\mathbb{1}_{l',i}\mathbb{S} + \mathbb{U}\partial_{l',i}\mathbb{S}))_{l,j}\right. \\ \left.+(1-\beta_{L,2})(\partial_{l',i}\mathbb{E}(\mathbb{U}\odot\mathbb{U})_{l,j} + 2\mathbb{E}(\mathbb{U}\odot\mathbb{1}_{l',i})_{l,j})\right) \tag{2.2.84}$$

$$\partial_{l',i}\mathbb{J}_{l,j} = \frac{1}{\rho^2}\gamma_{L,3}(1-\beta_{L,3})\left(\partial_{l',i}\mathbb{E}(\mathbb{U}\odot\mathbb{U})^2_{l,j} + 4\mathbb{E}(\mathbb{U}\odot\mathbb{U})_{l,j}(\mathbb{U}\odot\mathbb{1}_{l',i})_{l,j}\right)$$
$$+\gamma_{L,3}\beta_{L,3}(4\mathbb{1}_{l',i}\otimes\mathbb{U}^{\otimes 3}\otimes\mathbb{S} + \mathbb{U}^{\otimes 4}\otimes\partial_{l',i}\mathbb{S})_{l,j} \tag{2.2.85}$$

$$\partial_{l',i}\mathbb{G}_{l,j} = \frac{2}{\rho}\gamma_K\alpha_K\beta_K\left(\mathbb{1}_{l',i}\odot(\mathbb{S}\mathbb{U}) + \mathbb{U}\odot(\mathbb{S}\mathbb{1}_{l',i} + \partial_{l',i}\mathbb{S}\mathbb{U})\right)_{l,j}$$
$$+\frac{2}{\rho}\gamma_K\alpha_K(1-\beta_K)\left(\partial_{l',i}\mathbb{E}(\mathbb{U}\odot\mathbb{U})_{l,j} + 2\mathbb{E}(\mathbb{1}_{l',i}\odot\mathbb{U})_{l,j}\right)$$
$$-\frac{1}{\rho}\alpha_{L,1}\beta_{L,1}\left(\mathbb{1}_{l',i}\odot(\mathbb{S}\mathbb{U}^2) + \mathbb{U}\odot(\partial_{l',i}\mathbb{S}\mathbb{U}^2 + 2\mathbb{S}\mathbb{U}\mathbb{1}_{l',i})\right)_{l,j} \tag{2.2.86}$$
$$-\frac{1}{\rho}\alpha_{L,1}(1-\beta_{L,1})\left(\mathbb{E}\mathbb{1}_{l',i}\odot\mathbb{U}^2 + \mathbb{U}\odot(\partial_{l',i}\mathbb{E}\mathbb{U}^2 + 2\mathbb{E}\mathbb{U}\mathbb{1}_{l',i})\right)_{l,j}$$
$$+\frac{2}{\rho}\alpha_{L,2}((\mathbb{1}_{l',i}\odot(\mathbb{I}\mathbb{U}))_{l,j} + \mathbb{U}\odot(\partial_{l',i}\mathbb{I}\mathbb{U} + \mathbb{I}\mathbb{1}_{l',i}))_{l,j} - \frac{1}{2\rho}\alpha_{L,3}(\mathbb{1}_{l',i}\odot\mathbb{J} + \mathbb{U}\odot\partial_{l',i}\mathbb{J})_{l,j}.$$

5 - We iterate the Newton algorithm until the Newton relative error ϵ becomes smaller than the `tolerance` parameter. The Newton error is defined as

$$\epsilon = \frac{\|\mathbb{U}^{(n+1)} - \mathbb{U}^{(n)}\|_2}{\|\mathbb{U}^{(n)}\|_2} \tag{2.2.87}$$

where $\|\cdot\|_2$ is the l_2 norm. The energy thus obtained is

$$e = \frac{\rho}{2}\mathbb{E} \tag{2.2.88}$$

the Fourier transform $\hat{u}$ of the solution is

$$\hat{u}(k_{l,j}) \approx \frac{\mathbb{U}_{l,j}}{\rho} \tag{2.2.89}$$

where $k_{l,j}$ was defined in (2.2.40), and the solution u in real space is obtained by inverting the Fourier transform, following the prescription of (2.2.35):

$$u(|x|) = \int \frac{dk}{(2\pi)^3}\, e^{-ikx}\hat{u}(|k|) \approx \sum_{l=0}^{J-1}\frac{\tau_{l+1}-\tau_l}{16\pi}\sum_{j=1}^{N} w_j\cos(\tfrac{\pi x_j}{2})(1+\vartheta_l(x_j))^2\vartheta_l(x_j)\mathbb{U}_{l,j}\sin(\vartheta_l(x_j)|x|). \tag{2.2.90}$$

2.2.2.7. Condensate fraction

To compute the condensate fraction, we solve the modified `anyeq` (see [CJL21]):

$$(-\Delta + 2\mu)u_\mu = (1-u_\mu)v - 2\rho K + \rho^2 L. \tag{2.2.91}$$

where K and L are defined as in (2.2.2)-(2.2.7) in which u is replaced with u_μ. The uncondensed fraction is then

$$\eta = \partial_\mu e|_{\mu=0} = -\frac{\rho}{2}\int dx\; v(x)\partial_\mu u_\mu(x)|_{\mu=0}. \tag{2.2.92}$$

To compute the energy in the presence of the parameter μ, we proceed in the same way as for $\mu = 0$, the only difference being that k^2 should formally be replaced by $k^2 + 2\mu$. In other words,

we consider $\mathbb{U}_{j,l} = u_\mu(|k_{j,l}|)$ and define $\Xi(\mathbb{U}, \mu)$ in the same way as in (2.2.76), except that $\mathbb{X}_i$ should be replaced by

$$\mathbb{X}_{l,j} = \frac{1}{\mathbb{L}_{l,j}} \left(\mathbb{K}_{l,j} - \mathbb{L}_{l,j} + \frac{k_{l,j}^2 + 2\mu}{2\rho} + \mathbb{I}_{l,j} \right). \tag{2.2.93}$$

We then solve

$$\Xi(\mathbb{U}_\mu, \mu) = 0 \tag{2.2.94}$$

By differentiating this identity with respect to μ, we find $\partial_\mu u_\mu$:

$$\partial_\mu \mathbb{U}|_{\mu=0} = -(D\Xi)^{-1} \partial_\mu \Xi|_{\mu=0} \tag{2.2.95}$$

and

$$\partial_\mu \Xi|_{\mu=0} = \frac{(1+\mathbb{Y}_{l,j})\partial_\mu \mathbb{X}_{l,j}}{2(\mathbb{X}_{l,j}+1)^2} \Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right) + \frac{(1+\mathbb{Y}_{l,j})^2}{(\mathbb{X}_{l,j}+1)^4} \partial_\mu \mathbb{X}_{l,j} \partial\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right), \quad \partial_\mu \mathbb{X}_{l,j} = \frac{1}{\rho \mathbb{L}_{l,j}}. \tag{2.2.96}$$

We then approximate

$$\eta \approx -\frac{1}{2} \langle \mathfrak{v} \partial_\mu \mathbb{U} \rangle \tag{2.2.97}$$

(see (2.2.54)).

2.2.2.8. Correlation function (spherical average)

The two-point correlation function is

$$c_2(x) := 2\rho \frac{\delta e}{\delta v(x)} \tag{2.2.98}$$

and its spherical average is

$$C_2(|x|) := \frac{1}{4\pi |x|^2} \int dy \; \delta(|x| - |y|) c_2(y). \tag{2.2.99}$$

In Fourier space,

$$c_2(x) = 2\rho \int dk \; e^{ikx} \frac{\delta e}{\delta \hat{v}(k)} \tag{2.2.100}$$

so

$$C_2(|x|) = 2\rho \int dk \; \left(\frac{1}{4\pi |x|^2} \int dy \; \delta(|x| - |y|) e^{iky} \right) \frac{\delta e}{\delta \hat{v}(k)} = 2\rho \int dk \; \frac{\sin(|k||x|)}{|k||x|} \frac{\delta e}{\delta \hat{v}(k)}. \tag{2.2.101}$$

1 - We can compute this quantity by considering a modified `anyeq` in Fourier space, by formally replacing $\hat{v}$ with

$$\hat{v} + \lambda g(|k|), \quad g(|k|) := \frac{\sin(|k||x|)}{|k||x|}. \tag{2.2.102}$$

Indeed, if e_λ denotes the energy of this modified equation,

$$\partial_\lambda e_\lambda|_{\lambda=0} = \int dk \; \frac{\delta e}{\delta \hat{v}(k)} \partial_\lambda (\hat{v}(k) + \lambda g(|k|)) = \int dk \; g(|k|) \frac{\delta e}{\delta \hat{v}(k)}. \tag{2.2.103}$$

So, denoting the solution of the modified equation by u_λ,

$$C_2(x) = 2\rho \partial_\lambda e_\lambda|_{\lambda=0} = \rho^2 g(0) - \rho^2 \int \frac{dk}{(2\pi)^3} \; (g(k)\hat{u}(k) + \hat{v}(k)\partial_\lambda \hat{u}_\lambda(k)|_{\lambda=0}). \tag{2.2.104}$$

We compute $\partial_\lambda u_\lambda|_{\lambda=0}$ in the same way as the uncondensed fraction: we define $\Xi(\mathbb{U},\lambda)$ by formally adding $\lambda g(|k|)$ to $\hat{v}$, solve $\Xi(\mathbb{U},\lambda)=0$, and differentiate:

$$\partial_\lambda \mathbb{U}|_{\lambda=0} = -(D\Xi)^{-1}\partial_\lambda \Xi|_{\lambda=0}. \tag{2.2.105}$$

2 - We compute $\partial_\lambda \Xi|_{\lambda=0}$:

$$\begin{aligned}\partial_\lambda \Xi_{l,j} = \delta_{l,l'}\delta_{i,j} + \frac{1}{2(\mathbb{X}_{l,j}+1)}\left(\frac{(1+\mathbb{Y}_{l,j})\partial_\lambda \mathbb{X}_{l,j}}{\mathbb{X}_{l,j}+1} - \partial_\lambda \mathbb{Y}_{l,j}\right)\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right)\\ + \frac{(1+\mathbb{Y}_{l,j})}{2(\mathbb{X}_{l,j}+1)^3}\left(\frac{2(1+\mathbb{Y}_{l,j})}{\mathbb{X}_{l,j}+1}\partial_\lambda \mathbb{X}_{l,j} - \partial_\lambda \mathbb{Y}_{l,j}\right)\partial\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right)\end{aligned} \tag{2.2.106}$$

with

$$\partial_\lambda \mathbb{S}_{l,j} = \mathfrak{g}_{l,j} - \frac{1}{\rho}(\mathbb{U}\odot\mathfrak{g})_{l,j}, \quad \partial_\lambda \mathbb{E} = g(0) - \frac{1}{\rho}\langle \mathbb{U}\mathfrak{g}\rangle \tag{2.2.107}$$

$$\partial_\lambda \mathbb{X}_{l,j} = \frac{1}{\mathbb{L}_{l,j}}\left(\partial_\lambda \mathbb{K}_{l,j} - \partial_\lambda \mathbb{L}_{l,j} + \partial_\lambda \mathbb{I}_{l,j}\right) - \frac{\partial_\lambda \mathbb{L}_{l,j}}{\mathbb{L}_{l,j}}\mathbb{X}_{l,j} \tag{2.2.108}$$

$$\partial_\lambda \mathbb{Y}_{l,j} = \frac{1}{\mathbb{L}_{l,j}}\left(\partial_\lambda \mathbb{S}_{l,j} - \partial_\lambda \mathbb{L}_{l,j} + \frac{1}{2}\partial_\lambda \mathbb{J}_{l,j} + \partial_\lambda \mathbb{G}_{l,j}\right) - \frac{\partial_\lambda \mathbb{L}_{l,j}}{\mathbb{L}_{l,j}}\mathbb{Y}_{l,j} \tag{2.2.109}$$

$$\partial_\lambda \mathbb{K}_{l,j} := \gamma_K(\beta_K \partial_\lambda \mathbb{S}_{l,j} + (1-\beta_K)\partial_\lambda \mathbb{E}), \quad \partial_\lambda \mathbb{L}_{l,j} := \gamma_{L,1}(\beta_{L,1}\partial_\lambda \mathbb{S}_{l,j} + (1-\beta_{L,1})\partial_\lambda \mathbb{E}) \tag{2.2.110}$$

$$\partial_\lambda \mathbb{I}_{l,j} = \frac{1}{\rho}\gamma_{L,2}\left(\beta_{L,2}(\mathbb{U}\odot(\mathbb{U}\partial_\lambda \mathbb{S}))_{l,j} + (1-\beta_{L,2})\partial_\lambda \mathbb{E}(\mathbb{U}\odot\mathbb{U})_{l,j}\right) \tag{2.2.111}$$

$$\partial_\lambda \mathbb{J}_{l,j} = \frac{1}{\rho^2}\gamma_{L,3}(1-\beta_{L,3})\partial_\lambda \mathbb{E}(\mathbb{U}\odot\mathbb{U})^2_{l,j} + \gamma_{L,3}\beta_{L,3}(\mathbb{U}^{\otimes 4}\otimes\partial_\lambda \mathbb{S})_{l,j} \tag{2.2.112}$$

$$\begin{aligned}\partial_\lambda \mathbb{G}_{l,j} = \frac{2}{\rho}\gamma_K\alpha_K\beta_K\left(\mathbb{U}\odot(\partial_\lambda \mathbb{S}\mathbb{U})\right)_{l,j} + \frac{2}{\rho}\gamma_K\alpha_K(1-\beta_K)\partial_\lambda \mathbb{E}(\mathbb{U}\odot\mathbb{U})_{l,j} - \frac{1}{\rho}\alpha_{L,1}\beta_{L,1}\left(\mathbb{U}\odot\partial_\lambda \mathbb{S}\mathbb{U}^2\right)_{l,j}\\ -\frac{1}{\rho}\alpha_{L,1}(1-\beta_{L,1})\partial_\lambda \mathbb{E}\left(\mathbb{U}\odot(\mathbb{U}^2)\right)_{l,j} + \frac{2}{\rho}\alpha_{L,2}\mathbb{U}\odot(\partial_\lambda \mathbb{I}\mathbb{U})_{l,j} - \frac{1}{2\rho}\alpha_{L,3}(\mathbb{U}\odot\partial_\lambda \mathbb{J})_{l,j}.\end{aligned} \tag{2.2.113}$$

To evaluate $(\mathbb{U}\odot\mathfrak{g})$ and $\langle\mathbb{U}\mathfrak{g}\rangle$, we proceed as in (2.2.59) and (2.2.60). To do so, we replace $\hat{v}$ with g in the computation of Υ.

3 - In order to invert the Fourier transform in (2.2.101) numerically, we will use a Hann window (see appendix A4)

$$H_L(k) := \mathbb{1}_{|k|<\frac{L}{2}}\cos^2(\tfrac{\pi|k|}{L}). \tag{2.2.114}$$

The parameter L is set using `window_L`. The computation is changed only in that g is changed to $H_L(k)\frac{\sin(|k||x|)}{|k||x|}$.

4 - To compute the maximum of C_2, we use a modified Newton algorithm. The initial guess for the maximum is $|x_0| = \rho^{-\frac{1}{3}}$`x0`. The modified Newton algorithm is an iteration:

$$x_{n+1} = x_n + \frac{\partial C_2(|x_n|)}{|\partial^2 C_2(|x_n|)|} \tag{2.2.115}$$

in which the derivatives are approximated using finite differences:

$$\partial C_2(x) \approx \frac{C_2(|x|+dx) - C_2(|x|)}{dx}, \quad \partial^2 C_2(x) \approx \frac{C_2(|x|+dx) + C_2(|x|-dx) - 2C_2(|x|)}{dx^2}. \tag{2.2.116}$$

This is a modification of the usual Newton iteration $x_n + \partial C_2/\partial^2 C_2$ which is designed to follow the direction of the gradient, and thus to move toward a local maximum. In addition, if $|\partial C_2|/|\partial^2 C_2|$ is larger than `maxstep`, then the step is replaced with $\pm$`maxstep`. This prevents the algorithm from stepping over a maximum and land on another, further away. This is useful if one has a good idea of where the global maximum is, and does not want to get trapped in a smaller local maximum.

The algorithm is run for a maximum of `maxiter` iterations, or until $|x_{n+1} - x_n|$ is smaller than `tolerance`. If the maximal number of iterations is reached, or if the solution found is not a local maximum, then the algorithm fails, and returns $+\infty$. The point thus computed is therefore a local maximum, but it is not guaranteed to be the global maximum.

2.2.2.9. Fourier transform of two-point correlation (spherical average)

The Fourier transform of the two-point correlation function is

$$\hat{c}_2(q) := 2\rho \frac{\delta e}{\delta v(q)} \tag{2.2.117}$$

and its spherical average is

$$\hat{C}_2(|q|) := \frac{1}{4\pi|q|^2} \int dk\ \delta(|q| - |k|) c_2(k) = \frac{\rho}{2\pi|q|^2} \int dk\ \delta(|q| - |k|) \frac{\delta e}{\delta \hat{v}(k)}. \tag{2.2.118}$$

1 - To compute $\frac{\delta e}{\delta \hat{v}(q)}$, one idea would be to proceed in the same way as for the two-point correlation function, by replacing $\hat{v}$ with

$$\hat{v} + \lambda g(|k|), \quad g(|k|) := \frac{1}{4\pi|q|^2} \delta(|q| - |k|) \tag{2.2.119}$$

where δ is the Dirac-delta function distribution (compare this with (2.2.102)). However, the δ function causes all sorts of problems with the Chebyshev polynomial exansion and the quadratures.

2 - Instead, we approximate $\hat{C}_2$ by convolving it with a normalized Gaussian: let

$$\Gamma_L(|k|) := \left(\frac{L}{2\pi}\right)^{\frac{3}{2}} e^{-\frac{L}{2}k^2} \tag{2.2.120}$$

$$\hat{\mathfrak{C}}_2(|q|) := \int dp\ \hat{C}_2(|q-p|)\Gamma_L(|p|) = \int dk \int dp\ \frac{\rho}{2\pi|q-p|^2} \delta(|q-p| - |k|) \frac{\delta e}{\delta \hat{v}(k)} \Gamma_L(|p|) \tag{2.2.121}$$

which by lemma A3.1 is

$$\hat{\mathfrak{C}}_2(|q|) = \int dk\ \frac{\rho}{|q|} \frac{\delta e}{\delta \hat{v}(k)} \int_0^\infty dt \int_{||q|-t|}^{|q|+t} ds\ s \frac{\delta(t - |k|)}{t} \Gamma_L(s) \tag{2.2.122}$$

that is

$$\hat{\mathfrak{C}}_2(|q|) = \int dk\ \frac{\delta e}{\delta \hat{v}(k)} \frac{\rho}{|q||k|} \int_{||q|-|k||}^{|q|+|k|} ds\ s\Gamma_L(s) \tag{2.2.123}$$

which is the directional derivative of e with respect to $\hat{v}$ in the direction of $2\rho g$ with

$$g(|k|) := \frac{1}{2|q||k|} \int_{||q|-|k||}^{|q|+|k|} ds\ s\Gamma_L(s) = \frac{1}{2|k|rL} (\Gamma_L(|k| - r) - \Gamma_L(|k| + r)). \tag{2.2.124}$$

Note that

$$g(0) := \Gamma_L(|q|). \tag{2.2.125}$$

To compute this derivative, we replace $\hat{v}$ with

$$\hat{v} + \lambda g(|k|) \tag{2.2.126}$$

so, denoting the solution of the modified equation by u_λ, for $q \neq 0$,

$$\hat{\mathfrak{C}}_2(|q|) = 2\rho\partial_\lambda e_\lambda|_{\lambda=0} = \rho^2 \left(- \int \frac{dk}{(2\pi)^3}\ g(|k|)\hat{u}(|k|) - \int \frac{dk}{(2\pi)^3}\ \hat{v}(|k|)\partial_\lambda \hat{u}_\lambda(|k|)|_{\lambda=0} \right). \tag{2.2.127}$$

To compute $\partial_\lambda \hat{u}_\lambda|_{\lambda=0}$, we differentiate $\Xi(\mathbb{U}, \lambda) = 0$:

$$\partial_\lambda \mathbb{U}|_{\lambda=0} = -(D\Xi)^{-1}\partial_\lambda \Xi|_{\lambda=0}. \tag{2.2.128}$$

The computation of $\partial_\lambda \Xi|_{\lambda=0}$ is identical to (2.2.106), but taking

$$\mathfrak{g}_{l,j} = g(|k_{l,j}|). \tag{2.2.129}$$

with g defined in (2.2.124).

3 - To compute the maximum of $\hat{C}_2$, we proceed as for C_2, see (2.2.115)-(2.2.116). The only difference is that the algorithm is initialized with $|k_0| = \rho^{\frac{1}{3}}\texttt{k0}$.

2.2.2.10. Correlation function of uncondensed particles (spherical average)

To compute the correlation function among uncondensed particles, denoted by $\gamma_2(|\xi|)$, we solve the modified `anyeq` (see [Ja23]):

$$-\Delta u_\mu = (1 - u_\mu)v - 2\rho K + \rho^2 L - \frac{\rho\mu}{2\pi|\xi|^2} u(|\xi|)\delta(|\xi| - |x|) \tag{2.2.130}$$

where K and L are defined as in (2.2.2)-(2.2.7) in which u is replaced with u_μ. In Fourier space,

$$k^2 \hat{u}_\mu = \hat{S} - 2\rho\hat{K} + \rho^2 \hat{L} - 2\rho\mu u(|\xi|)\frac{\sin(|k||\xi|)}{|k||\xi|}. \tag{2.2.131}$$

The uncondensed correlation function is then

$$\gamma_2(|\xi|) = \partial_\mu e|_{\mu=0} = -\frac{\rho}{2} \int dx\ v(x)\partial_\mu u_\mu(x)|_{\mu=0}. \tag{2.2.132}$$

To compute the energy in the presence of the parameter μ, we proceed in the same way as for $\mu = 0$, the only difference being that the term

$$\mu g(|k|) := -\mu 2\rho u(|\xi|)\frac{\sin(|k||\xi|)}{|k||\xi|} \tag{2.2.133}$$

should formally be added to the right side of (2.2.18). In other words, we consider $\mathbb{U}_{j,l} = u_\mu(|k_{j,l}|)$ and define $\Xi(\mathbb{U}, \mu)$ in the same way as in (2.2.76), except that $\mathbb{Y}_{l,j}$ should be replaced by

$$\mathbb{Y}_{l,j} = \frac{1}{\mathbb{L}_{l,j}} \left(\mathbb{S}_{l,j} - \mathbb{L}_{l,j} + \frac{1}{2}\mathbb{J}_{l,j} + \mathbb{G}_{l,j} + \mu g(k_{l,j}) \right). \tag{2.2.134}$$

We then solve

$$\Xi(\mathbb{U}_\mu, \mu) = 0 \tag{2.2.135}$$

By differentiating this identity with respect to μ, we find $\partial_\mu u_\mu$:

$$\partial_\mu \mathbb{U}|_{\mu=0} = -(D\Xi)^{-1}\partial_\mu \Xi|_{\mu=0} \tag{2.2.136}$$

and

$$\partial_\mu \Xi|_{\mu=0} = -\frac{\partial_\mu \mathbb{Y}_{l,j}}{2(\mathbb{X}_{l,j}+1)} \Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right) - \frac{(1+\mathbb{Y}_{l,j})\partial_\mu \mathbb{Y}_{l,j}}{2(\mathbb{X}_{l,j}+1)^3} \partial\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right), \quad \partial_\mu \mathbb{Y}_{l,j} = \frac{g(k_{l,j})}{\mathbb{L}_{l,j}}. \tag{2.2.137}$$

We then approximate

$$\gamma_2(\xi) \approx -\frac{1}{2} \langle \mathfrak{v} \partial_\mu \mathbb{U} \rangle \tag{2.2.138}$$

(see (2.2.54)).

In order to avoid numerical oscillations due to the sin function, we will use a Hann window (see appendix A4)

$$H_L(k) := \mathbb{1}_{|k|<\frac{L}{2}} \cos^2(\tfrac{\pi|k|}{L}). \tag{2.2.139}$$

The parameter L is set using `window_L`. The computation is changed only in that g is changed to $H_L(k)g(k)$.

2.2.2.11. Momentum distribution

To compute the momentum distribution (see [CHe21]), we add a parameter λ to `anyeq`:

$$-\Delta u_\lambda(|x|) = (1-u_\lambda(|x|))v(|x|) - 2\rho K(|x|) + \rho^2 L(|x|) - 2\lambda \hat{u}_0(q)\cos(q\cdot x) \tag{2.2.140}$$

($\hat{u}_0 \equiv \hat{u}_\lambda|_{\lambda=0}$). The momentum distribution is then

$$\mathcal{M}(q) = \partial_\lambda e|_{\lambda=0} = -\frac{\rho}{2}\int \frac{dk}{(2\pi)^3}\ \hat{v}(k)\partial_\lambda \hat{u}_\lambda(k)|_{\lambda=0}. \tag{2.2.141}$$

Note that the Fourier transform of $2\lambda\hat{u}_0(q)\cos(q\cdot x)$ is

$$-(2\pi)^3 \lambda \hat{u}_0(q)(\delta(q+k)+\delta(q-k)). \tag{2.2.142}$$

The presence of delta functions does not play well with the Chebyshev polynomial expansion and the quadratures.

1 - We will consider two different ways of getting around this.

1-1 - One idea is to compute a regularization of $\mathcal{M}(q)$ by convolving it with a peaked spherically symmetric function. Let Γ_L denote the Gaussian with variance $1/\sqrt{L}$:

$$\Gamma_L(|k|) := \left(\frac{L}{2\pi}\right)^{\frac{3}{2}} e^{-\frac{L}{2}k^2}. \tag{2.2.143}$$

In fact, we will scale L with k, and set L to

$$L = \sqrt{\texttt{window\_L}}/k^2. \tag{2.2.144}$$

To compute

$$\mathfrak{M}(q) := \mathcal{M} * \Gamma_L(q) \tag{2.2.145}$$

we solve the equation

$$-\Delta u_\lambda(|x|) = (1-u_\lambda(|x|))v(|x|) - 2\rho K(|x|) + \rho^2 L(|x|) - 2\lambda \int dk\ \hat{u}_0(k)\cos(k\cdot x)\Gamma_L(q-k). \tag{2.2.146}$$

Note that the Fourier transform of

$$-2\lambda \int dk\ \hat{u}_0(k)\cos(k\cdot x)\Gamma_L(q-k) \tag{2.2.147}$$

is

$$-(2\pi)^3\lambda\hat{u}_0(q)(\Gamma_L(k+q)+\Gamma_L(k-q)). \tag{2.2.148}$$

Since the ground state is unique, $\mathcal{M}$ is spherically symmetric. The term $\Gamma_L(k\pm q)$ is not, so we take its spherical average (which will not change the final result): by lemma A3.1,

$$-\frac{1}{4\pi r^2}\int dq\ \delta(|q|-r)(2\pi)^3\lambda\hat{u}_0(q)(\Gamma_L(k+q)+\Gamma_L(k-q)) = -\frac{(2\pi)^3}{|k|r}\lambda\hat{u}_0(r)\int_{||k|-r|}^{|k|+r} ds\ s\Gamma_L(s). \tag{2.2.149}$$

In this setup, the approximation of the delta function is thus

$$\tilde{\delta}(|k|,r) := \frac{1}{2|k|r}\int_{||k|-r|}^{|k|+r} ds\ s\Gamma_L(s) = \frac{1}{2|k|rL}(\Gamma_L(|k|-r)-\Gamma_L(|k|+r)). \tag{2.2.150}$$

To choose this method, set `window_L` to a finite value.

1-2 - Another approach is to contruct a discrete analog of the delta-functions in (2.2.142). The starting point we take is, for $q\neq 0$,

$$\int dk\ f(|k|)\delta(k-q) \equiv \frac{1}{4\pi|q|^2}\int dk\ f(|k|)\delta(|k|-|q|) = f(|q|) \tag{2.2.151}$$

so, when approximating the integral according to (2.2.35), we find

$$\frac{\pi}{8|q|^2}\sum_{l=0}^{J-1}(\tau_{l+1}-\tau_l)\sum_{j=1}^{N} w_j\cos(\tfrac{\pi x_j}{2})(1+\vartheta_l(x_j))^2\vartheta_l^2(x_j)f(\vartheta_l(x_j))\tilde{\delta}(\vartheta_l(x_j),|q|) = f(|q|) \tag{2.2.152}$$

where $\tilde{\delta}$ is the approximation of the delta-function. Since

$$\vartheta_l(x_j) \equiv k_{l,j} \tag{2.2.153}$$

(see (2.2.33)), we define the approximation of the delta function as

$$\tilde{\delta}(k_{l,j},k_{l',i}) := \delta_{l,l'}\delta_{j,i}\frac{8}{\pi}\left((\tau_{l+1}-\tau_l)w_j\cos(\tfrac{\pi x_j}{2})(1+k_{l,j})^2\right)^{-1}. \tag{2.2.154}$$

To choose this method, set `window_L` to `Inf`. This method seems to yield some fairly poor results!

2 - To compute the momentum distribution at q, we define $\Xi(\mathbb{U},\lambda)$ by formally adding

$$-2(2\pi)^3\lambda\hat{u}_0(|q|)\tilde{\delta}(k_{l,j},|q|) \tag{2.2.155}$$

to $\mathbb{G}_{l,j}$, which corresponds to replacing $\mathbb{Y}$ with

$$\mathbb{Y}_{l,j} = \frac{1}{\mathbb{L}_{l,j}}\left(\mathbb{S}_{l,j}-\mathbb{L}_{l,j}+\frac{1}{2}\mathbb{J}_{l,j}+\mathbb{G}_{l,j}-2(2\pi)^3\lambda\hat{u}(|q|)\tilde{\delta}(k_{l,j},|q|)\right). \tag{2.2.156}$$

Then we solve $\Xi(\mathbb{U},\lambda)=0$, and differentiate:

$$\partial_\lambda\mathbb{U}|_{\lambda=0} = -(D\Xi)^{-1}\partial_\lambda\Xi|_{\lambda=0}. \tag{2.2.157}$$

Finally,

$$\partial_\lambda\Xi_{l,j}|_{\lambda=0} = -\partial_\lambda\mathbb{Y}_{l,j}|_{\lambda=0}\left(\frac{1}{2(\mathbb{X}_{l,j}+1)}\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right)+\frac{(1+\mathbb{Y}_{l,j})}{2(\mathbb{X}_{l,j}+1)^3}\partial\Phi\left(\tfrac{1+\mathbb{Y}_{l,j}}{(\mathbb{X}_{l,j}+1)^2}\right)\right) \tag{2.2.158}$$

with

$$\partial_\lambda\mathbb{Y}_{l,j}|_{\lambda=0} = -\frac{2(2\pi)^3}{\mathbb{L}_{l,j}}\hat{u}(|q|)\tilde{\delta}(k_{l,j},|q|). \tag{2.2.159}$$

2.2.2.12. Compressibility

The compressibility is defined as

$$\chi := \frac{1}{\rho^2 \partial_\rho^2(\rho e)} = \frac{1}{\partial_{\log\rho}^2(\rho e) - \partial_{\log\rho}(\rho e)}. \tag{2.2.160}$$

We approximate these derivatives by finite differences:

$$\partial_{\log\rho}^2 f(\rho) \approx \frac{f(\rho_{j+1}) + f(\rho_{j-1}) - 2f(\rho_j)}{(\log\rho_{j+2} - \log\rho_{j+1})(\log\rho_{j+1} - \log\rho_j)} \tag{2.2.161}$$

and

$$\partial_{\log\rho} f(\rho) \approx \frac{1}{2}\left(\frac{f(\rho_{j+1}) - f(\rho_j)}{\log\rho_{j+1} - \log\rho_j} + \frac{f(\rho_j) - f(\rho_{j-1})}{\log\rho_j - \log\rho_{j-1}}\right). \tag{2.2.162}$$

2.3. `simpleq-Kv`

The method is used to compute observables for the simple equation

$$-\Delta u = v(1-u) - 4eu + 2e\rho u * u, \quad e = \frac{\rho}{2}\int dx\ (1 - u(|x|))v(|x|). \tag{2.3.1}$$

One can show [CJL21, Theorem 1.6] that the condensate fraction is

$$\eta = \frac{\rho \int v(x)\mathfrak{K}u(x)\ dx}{1 - \rho\int v(x)\mathfrak{K}(2u(x) - \rho u * u(x))\ dx} \tag{2.3.2}$$

with

$$\mathfrak{K} = (-\Delta + v + 4e(1 - \rho u*))^{-1}. \tag{2.3.3}$$

Similarly, the two-point correlation function is [CHe21, (45)]

$$C_2 = \rho^2(1-u) + \rho^2 \frac{\mathfrak{K}v(1-u) - 2\rho u * \mathfrak{K}v + \rho^2 u * u * \mathfrak{K}v}{1 - \rho\int dx\ v(x)\mathfrak{K}(2u(x) - \rho u * u(x))}. \tag{2.3.4}$$

Thus, using the fact that $\mathfrak{K}$ is self-adjoint, we can compute these observables of the simple equation directly from the knowledge of $\mathfrak{K}v$.

2.3.1. Usage

The computation uses the same approximation scheme as `anyeq`, as well as using the solution of `anyeq`. As such, it takes a similar list of parameters: `rho`, `tolerance`, `maxiter`, `P`, `N`, `J`, `minlrho_init`, `nlrho_init`.

The available `commands` are the following.

- `Kv`: compute $\mathfrak{K}v$ as a function of $|x|$.
  Extra parameters:
  - `xmin` (`Float64`, default: 0): minimum of the range of $|x|$ to be printed.
  - `xmax` (`Float64`, default: 100): maximum of the range of $|x|$ to be printed.
  - `nx` (`Int64`, default: 100): number of points to print (linearly spaced).

  Output (one line for each value of $|x|$): $[|x|]$ $[\mathfrak{K}v]$
- `condensate-fraction`: compute the uncondensed fraction as a function of ρ

Extra parameters:

- `minlrho` (`Float64`, default: 10^{-6}): minimal value for $\log_{10}\rho$.
- `maxlrho` (`Float64`, default: 10^{2}): maximal value for $\log_{10}\rho$.
- `nlrho` (`Int64`, default: 100): number of values for ρ (spaced logarithmically).
- `minrho` (`Float64`, default: 10^{-6}): minimal value for ρ.
- `maxrho` (`Float64`, default: 10^{2}): maximal value for ρ.
- `nrho` (`Int64`, default: 0): number of values for ρ (spaced linearly). If `nrho` is $\neq 0$, then the linear spacing will be used, and `minlrho`, `maxlrho`, `nlrho` will be ignored. Otherwise, the logarithmic spacing will be used and `minrho`, `maxrho` will be ignored.
- `rhos` (`Array{Float64}`): list of values for ρ, specified as a '`,`' separated list. This parameter takes precedence over `minlrho`, `maxlrho`, `nlrho`, `minrho`, `maxrho`, `nrho`.

Output (one line for each value of ρ): $[\rho]$ $[\eta]$

- `2pt`: compute the two-point correlation as a function of $|x|$.
  Extra parameters: same as `Kv`.
  Output (one line for each value of $|x|$): $[|x|]$ $[C_2]$

2.3.2. Description

In Fourier space (2.2.8),

$$\hat{\mathfrak{K}}^{-1} f := \int dx\; e^{ikx}(-\Delta + 4e(1-\rho u*) + v) f = (k^2 + 4e(1-\rho\hat{u}(|k|)) + \hat{v}\hat{*})\hat{f} \tag{2.3.5}$$

where $\hat{*}$ is defined in (2.2.9). We follow the same approximation scheme as `anyeq`:

$$\hat{\mathfrak{K}}^{-1} f(k_{l,j}) \approx (k_{l,j}^2 + 4e(1-\mathbb{U}_{l,j}))\mathfrak{f}_{l,j} + (\mathfrak{v}\odot\mathfrak{f})_{l,j} \tag{2.3.6}$$

with $\mathfrak{f}_{l,j} := f(k_{l,j})$, $\mathbb{U}_{l,j} := \rho u(|k_{l,j}|)$, $\odot$ is defined in (2.2.45), and $k_{l,j}$ is defined in (2.2.40). Therefore, we approximate the operator $\hat{\mathfrak{K}}^{-1}$ by a matrix:

$$\hat{\mathfrak{K}}^{-1} f(k_{l,j}) \approx \sum_{l'=0}^{J-1}\sum_{i=1}^{N} M_{l,j;l',i}\mathfrak{f}_{l',i} \tag{2.3.7}$$

with, by (2.2.45) and (2.2.39),

$$\begin{aligned} M_{l,j;l',i} := {} & \delta_{l',l}\delta_{j,i}(k_{l,j}^2 + 4e(1-\mathbb{U}_{l,j})) \\ & + \frac{1}{4\pi^2|k_{l,j}|}\sum_{n,m=0}^{P}\sum_{l''=0}^{J-1}\mathfrak{F}^{(\nu_v)}_{l'',n}(\mathfrak{v})A^{(\nu_v,\nu_f)}_{l'',n;l',m}(|k_{l,j}|)\frac{\frac{2-\delta_{m,0}}{2}w_i\cos(\frac{m\pi(1+x_i)}{2})}{(1-\frac{\tau_{l'+1}-\tau_{l'}}{2}\sin(\frac{\pi x_i}{2})+\frac{\tau_{l'+1}+\tau_{l'}}{2})^{\nu_f}}. \end{aligned} \tag{2.3.8}$$

Defining $\mathbb{1}_{l',i}$ as the vector whose only non-vanishing component is that indexed by l',i which is equal to 1, we can rewrite

$$M_{l,j;l',i} := \delta_{l',l}\delta_{j,i}(k_{l,j}^2 + 4e(1-\mathbb{U}_{l,j})) + (\mathfrak{v}\odot\mathbb{1}_{l',i})_{l,j}. \tag{2.3.9}$$

Thus

$$\mathfrak{K}v \approx M^{-1}\mathfrak{v}. \tag{2.3.10}$$

To compute (2.3.2), we write

$$\eta = \frac{\rho \int \frac{dk}{(2\pi)^3} \hat{u}(k) \hat{\mathfrak{K}} \hat{v}(k)}{1 - \rho \int \frac{dk}{(2\pi)^3)} (2\hat{u}(k) - \rho \hat{u}^2(k)) \hat{\mathfrak{K}} \hat{v}(k)} \tag{2.3.11}$$

which we approximate as

$$\eta \approx \frac{\left\langle \mathbb{U} M^{-1} \mathfrak{v} \right\rangle}{1 - \left\langle (2\mathbb{U} - \mathbb{U}^2) M^{-1} \mathfrak{v} \right\rangle} \tag{2.3.12}$$

where $\langle \cdot \rangle$ is defined in (2.2.54). We can thus compute η using the solution $\mathbb{U}$ computed by `anyeq`.

To compute (2.3.4), we write

$$C_2 = \rho^2 (1 - u(x)) + \rho^2 \frac{\mathfrak{K} v(x)(1 - u(x)) + \int \frac{dk}{(2\pi)^3} (-2\rho \hat{u} \hat{\mathfrak{K}} \hat{v} + \rho^2 \hat{u}^2 \hat{\mathfrak{K}} \hat{v})}{1 - \rho \int \frac{dk}{(2\pi)^3} \ (2\hat{u}(k) - \rho \hat{u}^2(k)) \hat{\mathfrak{K}} \hat{v}(k)} \tag{2.3.13}$$

which we approximate as

$$C_2 \approx \rho^2 - \rho \left\langle e^{-ik_{l,j}x} \mathbb{U} \right\rangle + \rho^2 \frac{\left\langle M^{-1} \mathfrak{v} \right\rangle (1 - \rho^{-1} \left\langle e^{-ik_{l,j}x} \mathbb{U} \right\rangle) + \left\langle (-2\mathbb{U} M^{-1} \mathfrak{v} + \mathbb{U} M^{-1} \mathfrak{v}) \right\rangle}{1 - \left\langle (2\mathbb{U} - \mathbb{U}^2) M^{-1} \mathfrak{v} \right\rangle}. \tag{2.3.14}$$

We can thus compute C_2 using the solution $\mathbb{U}$ computed by `anyeq`.

2.4. `simpleq-hardcore`

This method is used to solve the Simple equation with a hardcore potential:

$$\begin{cases} (-\Delta + 4e) u(x) = 2e\rho u * u(x) & \text{for } |x| > 1 \\ u(x) = 1 & \text{for } |x| \leqslant 1 \end{cases} \tag{2.4.1}$$

with

$$e = -\frac{4\pi\rho \partial u|_{|x| \searrow 1}}{2(1 - \frac{8}{3}\pi\rho + \rho^2 \int_{|x|<1} dx \ u * u(x))}. \tag{2.4.2}$$

This equation is solved in x-space, and as such is very different from `easyeq`, and significantly longer to run.

2.4.1. Usage

Unless otherwise noted, this method takes the following parameters (specified via the `[-p params]` flag, as a '`;`' separated list of entries).

- `rho` (`Float64`, default: 10^{-6}): density ρ.
- `tolerance` (`Float64`, default: 10^{-11}): maximal size of final step in Newton iteration.
- `maxiter` (`Int64`, default: 21): maximal number of iterations of the Newton algorithm before giving up.
- `P` (`Int64`, default: 11): order of all Chebyshev polynomial expansions (denoted by P below).
- `N` (`Int64`, default: 12): order of all Gauss quadratures (denoted by N below).
- `J` (`Int64`, default: 10): number of splines (denoted by J below).

The available `commands` are the following.

- `energy_rho`: compute the energy e as a function of ρ.
  Disabled parameters: `rho`.
  Extra parameters:
  - `minlrho` (`Float64`, default: 10^{-6}): minimal value for $\log_{10}\rho$.
  - `maxlrho` (`Float64`, default: 10^{2}): maximal value for $\log_{10}\rho$.
  - `nlrho` (`Int64`, default: 100): number of values for ρ (spaced logarithmically).
  - `minrho` (`Float64`, default: 10^{-6}): minimal value for ρ.
  - `maxrho` (`Float64`, default: 10^{2}): maximal value for ρ.
  - `nrho` (`Int64`, default: 0): number of values for ρ (spaced linearly). If `nrho` is $\neq 0$, then the linear spacing will be used, and `minlrho`, `maxlrho`, `nlrho` will be ignored. Otherwise, the logarithmic spacing will be used and `minrho`, `maxrho` will be ignored.
  - `rhos` (`Array{Float64}`): list of values for ρ, specified as a '`,`' separated list. This parameter takes precedence over `minlrho`, `maxlrho`, `nlrho`, `minrho`, `maxrho`, `nrho`.

  Output (one line for each value of ρ): [ρ] [e] [Newton error ϵ].
  Multithread support: yes, different values of ρ split up among workers.
- `condensate_fraction_rho`: compute the uncondensed fraction η as a function of ρ.
  Disabled parameters: same as `energy_rho`.
  Extra parameters: same as `energy_rho`.
  Output (one line for each value of ρ): [ρ] [η] [Newton error ϵ].
  Multithread support: yes, different values of ρ split up among workers.
- `ux`: compute u as a function of $|x|$.
  Extra parameters:
  - `xmin` (`Float64`, default: 0): minimum of the range of $|x|$ to be printed.
  - `xmax` (`Float64`, default: 100): maximum of the range of $|x|$ to be printed.
  - `nx` (`Int64`, default: 100): number of points to print (linearly spaced).

  Output (one line for each value of x): [$|x|$] [$u(|x|)$]

2.4.2. Description

In order to carry out the computation of the solution of (2.4.1) and compute the condensate fraction at the same time, we will consider the equation with an added parameter $\mu > 0$:

$$(-\Delta + 4\epsilon)u = 2e\rho u * u, \quad \epsilon := e + \frac{\mu}{2} \tag{2.4.3}$$

for $|x| > 1$.

2.4.2.1. Energy

To compute the energy e of this equation, with the extra parameter μ, we consider the limit of the soft sphere potential $\lambda \mathbb{1}_{|x|<1}$ (see (2.1.1) with $\beta_K = \beta_L = 0$):

$$(-\Delta + 2\mu + 4e)u(x) = s_\lambda(x) + 2e\rho u * u, \quad s_\lambda(x) := \lambda(1 - u(x))\mathbb{1}_{|x|\leqslant 1}, \quad \frac{2e}{\rho} = \int dx\ s_\lambda(x). \tag{2.4.4}$$

Furthermore, since ∂u need not be continuous at $|x| = 1$, by integrating $-\Delta u$ over a thin spherical shell of radius 1, we find that, for $|x| \leqslant 1$,

$$-\Delta u(x) = -\delta(|x| - 1)\partial u|_{|x|\searrow 1} \tag{2.4.5}$$

so, formally,

$$s_\infty(x) = \mathbb{1}_{|x|\leqslant 1}(2e(2 - \rho u * u(x)) + 2\mu) - \delta(|x| - 1)\partial u|_{|x|\searrow 1} \tag{2.4.6}$$

and

$$\frac{2e}{\rho} = \int dx\ s_\infty(x) = 2e\left(\frac{8\pi}{3} - \rho\int_{|x|<1} u * u(x)\right) + \frac{8\pi}{3}\mu - 4\pi\partial u|_{|x|\searrow 1}. \tag{2.4.7}$$

Therefore,

$$e = \frac{2\pi\rho(\frac{2}{3}\mu - \partial u|_{|x|\searrow 1})}{1 - \frac{8\pi}{3}\rho + \rho^2\int_{|x|<1} dx\ u * u(x)}. \tag{2.4.8}$$

2.4.2.2. Integral equation

We turn the differential equation in (2.4.3) into an integral equation. Let

$$w(|x|) := |x|u(x) \tag{2.4.9}$$

we have, for $r > 1$,

$$(-\partial_r^2 + 4\epsilon)w(r) = 2e\rho r u * u(r). \tag{2.4.10}$$

Furthermore, the bounded solution of

$$(-\partial_r^2 + \alpha^2)w(r) = F(r), \quad w(1) = 1 \tag{2.4.11}$$

is

$$w(r) = e^{-\alpha(r-1)} + \int_0^\infty ds\ \frac{F(s+1)}{2\alpha}\left(e^{-\alpha|(r-1)-s|} - e^{-\alpha((r-1)+s)}\right) \tag{2.4.12}$$

so, for $r > 1$,

$$u(r) = \frac{1}{r}e^{-2\sqrt{\epsilon}(r-1)} + \frac{\rho e}{2r\sqrt{\epsilon}}\int_0^\infty ds\ (s+1)(u * u(s+1))\left(e^{-2\sqrt{\epsilon}|(r-1)-s|} - e^{-2\sqrt{\epsilon}((r-1)+s)}\right). \tag{2.4.13}$$

In order to compute the integral more easily, we split it:

$$\begin{aligned} u(r) = \frac{1}{r}e^{-2\sqrt{\epsilon}(r-1)} + \frac{\rho e}{r\sqrt{\epsilon}}\int_0^{r-1} ds\ (s+1)(u * u(s+1))\sinh(2\sqrt{\epsilon}s)e^{-2\sqrt{\epsilon}(r-1)} \\ + \frac{\rho e}{r\sqrt{\epsilon}}\sinh(2\sqrt{\epsilon}(r-1))\int_{r-1}^\infty ds\ (s+1)(u * u(s+1))e^{-2\sqrt{\epsilon}s}. \end{aligned} \tag{2.4.14}$$

We change variables in the last integral:

$$\begin{aligned} u(r) = \frac{1}{r}e^{-2\sqrt{\epsilon}(r-1)} + \frac{\rho e}{r\sqrt{\epsilon}}\int_0^{r-1} ds\ (s+1)(u * u(s+1))\sinh(2\sqrt{\epsilon}s)e^{-2\sqrt{\epsilon}(r-1)} \\ + \frac{\rho e}{2r\sqrt{\epsilon}}\left(1 - e^{-4\sqrt{\epsilon}(r-1)}\right)\int_0^\infty d\sigma\ (\sigma + r)(u * u(\sigma + r))e^{-2\sqrt{\epsilon}\sigma}. \end{aligned} \tag{2.4.15}$$

2.4.2.3. The auto-convolution term

We split

$$u(r) = \mathbb{1}_{r>1} u_+(r-1) + \mathbb{1}_{r\leqslant 1}. \tag{2.4.16}$$

in terms of which

$$u * u = \mathbb{1}_{r\leqslant 1} * \mathbb{1}_{r\leqslant 1} + 2\mathbb{1}_{r\leqslant 1} * (u_+(r-1)\mathbb{1}_{r>1}) + (\mathbb{1}_{r>1}u_+(r-1)) * (u_+(r-1)\mathbb{1}_{r>1}) \tag{2.4.17}$$

In bipolar coordinates (see lemma A3.1),

$$\mathbb{1}_{r\leqslant 1} * \mathbb{1}_{r\leqslant 1}(r) = \frac{2\pi}{r}\int_0^1 dt\ t\int_{|r-t|}^{r+t} ds\ s\mathbb{1}_{s\leqslant 1} \tag{2.4.18}$$

and, if $r \geqslant 0$ and $t \geqslant 0$, then

$$\int_{|r-t|}^{r+t} ds\ s\mathbb{1}_{s\leqslant 1} = \mathbb{1}_{r-1\leqslant t\leqslant r+1}\int_{|r-t|}^{\min(1,r+t)} ds\ s = \mathbb{1}_{r-1\leqslant t\leqslant r+1}\left(\mathbb{1}_{t\leqslant 1-r}2rt + \mathbb{1}_{t>1-r}\frac{1-(r-t)^2}{2}\right). \tag{2.4.19}$$

Therefore, if $r \geqslant 1$

$$\mathbb{1}_{r\leqslant 1} * \mathbb{1}_{r\leqslant 1}(r) = \mathbb{1}_{r\leqslant 2}\frac{2\pi}{r}\int_{r-1}^1 dt\ t\frac{1-(r-t)^2}{2} = \mathbb{1}_{r\leqslant 2}\frac{\pi}{12}(r-2)^2(r+4) \tag{2.4.20}$$

and if $r < 1$,

$$\mathbb{1}_{r\leqslant 1} * \mathbb{1}_{r\leqslant 1}(r) = \frac{2\pi}{r}\int_0^1 dt\ t\left(\mathbb{1}_{t\leqslant 1-r}2rt + \mathbb{1}_{t>1-r}\frac{1-(r-t)^2}{2}\right) \tag{2.4.21}$$

so

$$\mathbb{1}_{r\leqslant 1} * \mathbb{1}_{r\leqslant 1}(r) = \frac{4}{3}\pi(1-r)^3 + \frac{\pi}{12}r(36-48r+17r^2) = \frac{\pi}{12}(r-2)^2(r+4). \tag{2.4.22}$$

Thus, (2.4.20) holds for all r. Furthermore,

$$2\mathbb{1}_{r\leqslant 1} * (u_+(r-1)\mathbb{1}_{r>1}) = \frac{4\pi}{r}\int_1^\infty dt\ tu_+(t-1)\int_{|r-t|}^{r+t} ds\ s\mathbb{1}_{s\leqslant 1} \tag{2.4.23}$$

so, if $r \geqslant 0$ then, by (2.4.19),

$$2\mathbb{1}_{r\leqslant 1} * (u_+(r-1)\mathbb{1}_{r>1})(r) = \frac{2\pi}{r}\int_{\max(1,r-1)}^{r+1} dt\ tu_+(t-1)(1-(r-t)^2). \tag{2.4.24}$$

Finally, if $r \geqslant 0$ then

$$(\mathbb{1}_{r>1}u_+(r-1)) * (u_+(r-1)\mathbb{1}_{r>1})(r) = \frac{2\pi}{r}\int_1^\infty dt\ tu_+(t-1)\int_{\max(1,|r-t|)}^{r+t} ds\ su_+(s-1). \tag{2.4.25}$$

Thus, by (2.4.20), (2.4.24) and (2.4.25), for $r \geqslant -1$,

$$\begin{aligned} u*u(1+r) = \mathbb{1}_{r\leqslant 1}\frac{\pi}{12}(r-1)^2(r+5) + \frac{2\pi}{r+1}\int_{\max(0,r-1)}^{r+1} dt\ (t+1)u_+(t)(1-(r-t)^2) \\ + \frac{2\pi}{r+1}\int_0^\infty dt\ (t+1)u_+(t)\int_{\max(0,|r-t|-1)}^{r+t+1} ds\ (s+1)u_+(s). \end{aligned} \tag{2.4.26}$$

We then compactify the integrals by changing variables to $\tau = \frac{1-t}{1+t}$ and $\sigma = \frac{1-s}{1+s}$:

$$\begin{aligned} u*u(1+r) = \mathbb{1}_{r\leqslant 1}\frac{\pi}{12}(r-1)^2(r+5) + \frac{8\pi}{r+1}\int_{-\frac{r}{2+r}}^{\min(1,\frac{2-r}{r})} d\tau\ \frac{1}{(1+\tau)^3}u_+(\tfrac{1-\tau}{1+\tau})(1-(r-\tfrac{1-\tau}{1+\tau})^2) \\ + \frac{32\pi}{r+1}\int_{-1}^1 d\tau\ \frac{1}{(1+\tau)^3}u_+(\tfrac{1-\tau}{1+\tau})\int_{\alpha_-(1+r,\tau)}^{\min(1,\beta_+(r,\tau))} d\sigma\ \frac{1}{(1+\sigma)^3}u_+(\tfrac{1-\sigma}{1+\sigma}). \end{aligned} \tag{2.4.27}$$

with

$$\alpha_-(1+r,\tau) := \frac{-r-\frac{1-\tau}{1+\tau}}{2+r+\frac{1-\tau}{1+\tau}}, \quad \beta_+(r,\tau) := \frac{2-|r-\frac{1-\tau}{1+\tau}|}{|r-\frac{1-\tau}{1+\tau}|} \tag{2.4.28}$$

(note that α_- is the same as defined for fulleq). Finally, note that, if $\beta_+ < 1$, that is, if $|r-\frac{1-\tau}{1+\tau}| > 1$, then

$$\beta_+(r,\tau) = \alpha_+(|r-\tfrac{1-\tau}{1+\tau}| - 1 + \tfrac{1-\tau}{1+\tau}, \tau) \tag{2.4.29}$$

where

$$\alpha_+(r,\tau) := \frac{1-|r-\frac{1-\tau}{1+\tau}|}{1+|r-\frac{1-\tau}{1+\tau}|} \tag{2.4.30}$$

is the same as is defined for `anyeq` (2.2.44).

2.4.2.4. Chebyshev polynomial expansion

We use the same interpolation as we used in `anyeq`: (2.2.37)

$$\frac{1}{(1+\tau)^{\nu_u}} u_+(\tfrac{1-\tau}{1+\tau}) = \sum_{l=0}^{J-1} \mathbb{1}_{\tau_l \leqslant \tau < \tau_{l+1}} \sum_{n=0}^{\infty} \mathbf{F}^{(\nu_u)}_{l,n}(u_+) T_n(\tfrac{2\tau-(\tau_l+\tau_{l+1})}{\tau_{l+1}-\tau_l}) \tag{2.4.31}$$

(J is set by the parameter `J`) with

$$\mathbf{F}^{(\nu_u)}_{l,n}(u_+) := \frac{2-\delta_{n,0}}{\pi} \int_0^\pi d\theta \, \frac{\cos(n\theta)}{(1+\frac{\tau_{l+1}-\tau_l}{2}\cos(\theta)+\frac{\tau_{l+1}+\tau_l}{2})^{\nu_u}} u_+(\tfrac{2-(\tau_{l+1}-\tau_l)\cos(\theta)-(\tau_{l+1}+\tau_\alpha)}{2+(\tau_{l+1}-\tau_l)\cos(\theta)+(\tau_{l+1}+\tau_l)}) \tag{2.4.32}$$

and we take ν_u to be the decay exponent of u, which we will assume is $\nu_u = 4$. In particular, by (2.4.27),

$$u * u(1+r) = \mathbb{1}_{r\leqslant 1} B^{(0)}(r) + \sum_l \sum_n \mathbf{F}^{(\nu_u)}_{l,n}(u_+) B^{(1)}_{l,n}(r) + \sum_{l,l'} \sum_{n,m} \mathbf{F}^{(\nu_u)}_{l,n}(u_+) \mathbf{F}^{(\nu_u)}_{l',m}(u_+) B^{(2)}_{l,n;l',m}(r) \tag{2.4.33}$$

with

$$B^{(0)}(r) := \frac{\pi}{12}(r-1)^2(r+5) \tag{2.4.34}$$

$$B^{(1)}_{l,n}(r) := \mathbb{1}_{\tau_l < \frac{2-r}{r}} \mathbb{1}_{\tau_{l+1} > -\frac{r}{2+r}} \frac{8\pi}{r+1} \int_{\max(\tau_l, -\frac{r}{2+r})}^{\min(\tau_{l+1}, \frac{2-r}{r})} d\tau \, \frac{1}{(1+\tau)^{3-\nu_u}} (1-(r-\tfrac{1-\tau}{1+\tau})^2) T_n(\tfrac{2\tau-(\tau_l+\tau_{l+1})}{\tau_{l+1}-\tau_l}) \tag{2.4.35}$$

$$\begin{aligned} B^{(2)}_{l,n;l',m}(r) := \frac{32\pi}{r+1} \int_{\tau_l}^{\tau_{l+1}} d\tau \, \frac{1}{(1+\tau)^{3-\nu_u}} T_n(\tfrac{2\tau-(\tau_l+\tau_{l+1})}{\tau_{l+1}-\tau_l}) \mathbb{1}_{\tau_{l'} < \alpha_+(|r-\frac{1-\tau}{1+\tau}|-\frac{2\tau}{1+\tau},\tau)} \mathbb{1}_{\tau_{l'+1} > \alpha_-(1+r,\tau)} \cdot \\ \cdot \int_{\max(\tau_{l'}, \alpha_-(1+r,\tau))}^{\min(\tau_{l'+1}, \alpha_+(|r-\frac{1-\tau}{1+\tau}|-\frac{2\tau}{1+\tau},\tau))} d\sigma \, \frac{1}{(1+\sigma)^{3-\nu_u}} T_m(\tfrac{2\sigma-(\tau_{l'}+\tau_{l'+1})}{\tau_{l'+1}-\tau_{l'}}). \end{aligned} \tag{2.4.36}$$

Thus, by (2.4.15), for $r > 0$,

$$u_+(r,e,\epsilon) = D^{(0)}(r,e,\epsilon) + \sum_l \sum_n \mathbf{F}^{(\nu_u)}_{l,n}(u_+) D^{(1)}_{l,n}(r,e,\epsilon) + \sum_{l,l'} \sum_{n,m} \mathbf{F}^{(\nu_u)}_{l,n}(u_+) \mathbf{F}^{(\nu_u)}_{l',m}(u_+) D^{(2)}_{l,n;l',m}(r,e,\epsilon) \tag{2.4.37}$$

with

$$\begin{aligned} D^{(0)}(r,e,\epsilon) := \frac{1}{r+1} e^{-2\sqrt{\epsilon}r} + \frac{\rho e}{(r+1)\sqrt{\epsilon}} \int_0^{\min(1,r)} ds \, (s+1) B^{(0)}(s) \sinh(2\sqrt{\epsilon}s) e^{-2\sqrt{\epsilon}r} \\ + \mathbb{1}_{r\leqslant 1} \frac{\rho e}{2(r+1)\sqrt{\epsilon}} \left(1 - e^{-4\sqrt{\epsilon}r}\right) \int_0^{1-r} d\sigma \, (\sigma+r+1) B^{(0)}(\sigma+r) e^{-2\sqrt{\epsilon}\sigma} \end{aligned} \tag{2.4.38}$$

$$D^{(1)}_{l,n}(r,e,\epsilon) := \frac{\rho e}{(r+1)\sqrt{\epsilon}} \int_0^r ds\ (s+1) B^{(1)}_{l,n}(s) \sinh(2\sqrt{\epsilon}s) e^{-2\sqrt{\epsilon}r} + \frac{\rho e}{2(r+1)\sqrt{\epsilon}} \left(1 - e^{-4\sqrt{\epsilon}r}\right) \int_0^\infty d\sigma\ (\sigma + r + 1) B^{(1)}_{l,n}(\sigma + r) e^{-2\sqrt{\epsilon}\sigma}. \tag{2.4.39}$$

and

$$D^{(2)}_{l,n;l',m}(r,e,\epsilon) := \frac{\rho e}{(r+1)\sqrt{\epsilon}} \int_0^r ds\ (s+1) B^{(2)}_{l,n;l',m}(s) \sinh(2\sqrt{\epsilon}s) e^{-2\sqrt{\epsilon}r} + \frac{\rho e}{2(r+1)\sqrt{\epsilon}} \left(1 - e^{-4\sqrt{\epsilon}r}\right) \int_0^\infty d\sigma\ (\sigma + r + 1) B^{(2)}_{l,n;l',m}(\sigma + r) e^{-2\sqrt{\epsilon}\sigma}. \tag{2.4.40}$$

2.4.2.5. Energy

We now compute the approximation for the energy, using (2.4.2).

1 - We start with $\partial u|_{|x|\searrow 1}$. By (2.4.15),

$$\partial u|_{|x|\searrow 1} = -(1 + 2\sqrt{\epsilon}) + 2\rho e \int_0^\infty d\sigma\ (\sigma + 1)(u * u(\sigma + 1)) e^{-2\sqrt{\epsilon}\sigma} \tag{2.4.41}$$

which, by (2.4.33), becomes

$$\partial u|_{|x|\searrow 1} = -(1 + 2\sqrt{\epsilon}) + \gamma^{(0)}(e,\epsilon) + \sum_l \sum_n \mathbf{F}^{(\nu_u)}_{l,n}(u_+) \gamma^{(1)}_{l,n}(e,\epsilon) + + \sum_{l,l'} \sum_{n,m} \mathbf{F}^{(\nu_u)}_{l,n}(u_+) \mathbf{F}^{(\nu_u)}_{l',m}(u_+) \gamma^{(2)}_{l,n;l',m}(e,\epsilon) \tag{2.4.42}$$

with

$$\gamma^{(0)}(e,\epsilon) := 2\rho e \int_0^1 d\sigma\ (\sigma + 1) B^{(0)}(\sigma) e^{-2\sqrt{\epsilon}\sigma} \tag{2.4.43}$$

$$\gamma^{(1)}_{l,n}(e,\epsilon) := 2\rho e \int_0^\infty d\sigma\ (\sigma + 1) B^{(1)}_{l,n}(\sigma) e^{-2\sqrt{\epsilon}\sigma} \tag{2.4.44}$$

and

$$\gamma^{(2)}_{l,n;l',m}(e,\epsilon) := 2\rho e \int_0^\infty d\sigma\ (\sigma + 1) B^{(2)}_{l,n;l',m}(\sigma) e^{-2\sqrt{\epsilon}\sigma}. \tag{2.4.45}$$

2 - Let us now turn to $\int_{|x|<1} dx\ u * u(x)$. We have

$$\int_{|x|<1} dx\ u * u(x) = 4\pi \int_0^1 dr\ r^2 u * u(r) \tag{2.4.46}$$

so, by (2.4.33),

$$\int_{|x|<1} dx\ u * u(x) = \bar{\gamma}^{(0)}(r) + \sum_l \sum_n \mathbf{F}^{(\nu_u)}_{l,n}(u_+) \bar{\gamma}^{(1)}_{l,n}(r) + \sum_{l,l'} \sum_{n,m} \mathbf{F}^{(\nu_u)}_{l,n}(u_+) \mathbf{F}^{(\nu_u)}_{l',m}(u_+) \bar{\gamma}^{(2)}_{l,n;l',m}(r) \tag{2.4.47}$$

with

$$\bar{\gamma}^{(0)} := 4\pi \int_0^1 d\sigma\ \sigma^2 B^{(0)}(\sigma - 1) \tag{2.4.48}$$

$$\bar\gamma^{(1)}_{l,n} := 4\pi \int_0^1 d\sigma\ \sigma^2 B^{(1)}_{l,n}(\sigma-1) \tag{2.4.49}$$

and

$$\bar\gamma^{(2)}_{l,n;l',m} := 4\pi \int_0^1 d\sigma\ \sigma^2 B^{(2)}_{l,n;l',m}(\sigma-1). \tag{2.4.50}$$

3 - Thus,

$$e = 2\pi\rho\cdot \cdot \frac{C(\epsilon) - \gamma^{(0)}(e,\epsilon) - \sum_l \sum_n \mathbf{F}^{(\nu_u)}_{l,n}(u_+)\gamma^{(1)}_{l,n}(e,\epsilon) - \sum_{l,l'}\sum_{n,m}\mathbf{F}^{(\nu_u)}_{l,n}(u_+)\mathbf{F}^{(\nu_u)}_{l',m}(u_+)\gamma^{(2)}_{l,n;l',m}(e,\epsilon)}{1-\frac{8}{3}\pi\rho+\rho^2\left(\bar\gamma^{(0)} + \sum_l\sum_n \mathbf{F}^{(\nu_u)}_{l,n}(u_+)\bar\gamma^{(1)}_{l,n} + \sum_{l,l'}\sum_{n,m}\mathbf{F}^{(\nu_u)}_{l,n}(u_+)\mathbf{F}^{(\nu_u)}_{l',m}(u_+)\bar\gamma^{(2)}_{l,n;l',m}\right)}. \tag{2.4.51}$$

$$C(\epsilon) := \frac{2}{3}\mu + (1+2\sqrt{\epsilon}) = \frac{4}{3}(\epsilon - e) + 1 + 2\sqrt{\epsilon} \tag{2.4.52}$$

2.4.2.6. Newton algorithm

In this paragraph, we set $\epsilon = e$, that is, $\mu = 0$.

1 - As we did for `anyeq`, we discretize the integral in (2.4.32) by using a Gauss-Legendre quadrature, and truncate the sum over Chebyshev polynomials to order `P`. We then reduce the computation to a finite system of equations, whose variables are

$$e, \quad \mathfrak{u}_{l,j} := u_+(r_{l,j}), \quad r_{l,j} := \frac{2 + (\tau_{l+1}-\tau_l)\sin(\frac{\pi x_j}{2}) - (\tau_{l+1}+\tau_\alpha)}{2 - (\tau_{l+1}-\tau_l)\sin(\frac{\pi x_j}{2}) + (\tau_{l+1}+\tau_l)} \tag{2.4.53}$$

where x_j are the abcissa for Gauss-Legendre quadratures (see (2.2.40)). In other words, we define a vector $\mathbb{U}$ in dimension `N`×`J`+1, where the first `N`×`J` terms are $\mathfrak{u}_{l,j}$ and the last component is e. We then write (2.4.37) as, for $l \in \{0,\cdots,J-1\}$, $j \in \{1,\cdots,N\}$,

$$\Xi_{lN+j}(\mathbb{U}) = 0, \quad \Xi_{NJ+1}(\mathbb{U}) = 0 \tag{2.4.54}$$

(note that Ξ_{lN+j} corresponds to the pair (l,j)) with

$$\Xi_{lN+j}(\mathbb{U}) := -\mathfrak{u}_{l,j} + \mathbb{D}^{(0)}(r_{l,j},e) + \sum_{l'}\sum_n \mathfrak{F}^{(\nu_u)}_{l',n}(\mathfrak{u})\mathbb{D}^{(1)}_{l',n}(r_{l,j},e) + \\ + \sum_{l',l''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l',n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathfrak{u})\mathbb{D}^{(2)}_{l',n;l'',m}(r_{l,j},e) \tag{2.4.55}$$

$$\Xi_{NJ+1}(\mathbb{U}) := -e + \\ +2\pi\rho\frac{C(e) - \mathfrak{g}^{(0)}(e) - \sum_l\sum_n \mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{g}^{(1)}_{l,n}(e) - \sum_{l,l'}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l',m}(\mathfrak{u})\mathfrak{g}^{(2)}_{l,n;l',m}(e)}{1-\frac{8}{3}\pi\rho+\rho^2\left(\bar{\mathfrak{g}}^{(0)} + \sum_l\sum_n \mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\bar{\mathfrak{g}}^{(1)}_{l,n} + \sum_{l,l'}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l',m}(\mathfrak{u})\bar{\mathfrak{g}}^{(2)}_{l,n;l',m}\right)}. \tag{2.4.56}$$

where $\mathfrak{F}$ is defined in (2.2.39), and $\mathbb{D}^{(i)}$, $\mathfrak{g}^{(i)}$, $\bar{\mathfrak{g}}^{(i)}$ are defined like $D^{(i)}$, $\gamma^{(i)}$ and $\bar\gamma^{(i)}$ except that the integrals over bounded intervals are approximated using Gauss-Legendre quadratures, and the integrals from 0 to ∞ are approximated using Gauss-Laguerre quadratures. Gauss-Legendre

and Gauss-Laguerre quadratures and their errors are discussed in appendix A2. The orders of the quadratures are given by the variable `N`.

2 - We then solve $\Xi = 0$ using the Newton algorithm, that is, we define a sequence of $\mathbb{U}$'s:

$$\mathbb{U}^{(n+1)} = \mathbb{U}^{(n)} - (D\Xi(\mathbb{U}^{(n)}))^{-1}\Xi(\mathbb{U}^{(n)}) \tag{2.4.57}$$

where $D\Xi$ is the Jacobian of Ξ:

$$(D\Xi)_{\alpha,\beta} := \frac{\partial \Xi_\alpha}{\partial \mathbb{U}_\beta}. \tag{2.4.58}$$

We initialize the algorithm with

$$\mathbb{U}^{(0)}_{lN+j} = \frac{1}{(1+r_{l,j}^2)^2}, \quad \mathbb{U}^{(0)}_{JN+1} = \pi\rho. \tag{2.4.59}$$

3 - We are left with computing the Jacobian of Ξ:

$$\begin{aligned}\frac{\partial \Xi_{lN+j}(\mathbb{U})}{\partial \mathfrak{u}_{l',i}} = -\delta_{l,l'}\delta_{j,i} + \sum_{l''}\sum_{n} \mathfrak{F}^{(\nu_u)}_{l'',n}(\mathbb{1}_{l',i})\mathbb{D}^{(1)}_{l'',n}(r_{l,j},e) + \\ +2\sum_{l'',l'''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l'',n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l''',m}(\mathbb{1}_{l',j})\mathbb{D}^{(2)}_{l'',n;l''',m}(r_{l,j},e)\end{aligned} \tag{2.4.60}$$

where $\mathbb{1}_{l',i}$ is a vector whose only non-vanishing entry is the (l',i)-th, which is 1,

$$\begin{aligned}\frac{\partial \Xi_{JN+1}(\mathbb{U})}{\partial \mathfrak{u}_{l',i}} = \\ = 2\pi\rho\frac{-\sum_l\sum_n \mathfrak{F}^{(\nu_u)}_{l,n}(\mathbb{1}_{l',i})\mathfrak{g}^{(1)}_{l,n}(e) - 2\sum_{l,l''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathbb{1}_{l',i})\mathfrak{g}^{(2)}_{l,n;l'',m}(e)}{1-\frac{8}{3}\pi\rho+\rho^2\left(\bar{\mathfrak{g}}^{(0)}+\sum_l\sum_n\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\bar{\mathfrak{g}}^{(1)}_{l,n}+\sum_{l,l''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathfrak{u})\bar{\mathfrak{g}}^{(2)}_{l,n;l'',m}\right)} \\ -(\Xi_{NJ+1}(\mathbb{U})+e)\frac{\rho^2\left(\sum_l\sum_n\mathfrak{F}^{(\nu_u)}_{l,n}(\mathbb{1}_{l'.i})\bar{\mathfrak{g}}^{(1)}_{l,n}+2\sum_{l,l''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathbb{1}_{l',i})\bar{\mathfrak{g}}^{(2)}_{l,n;l'',m}\right)}{1-\frac{8}{3}\pi\rho+\rho^2\left(\bar{\mathfrak{g}}^{(0)}+\sum_l\sum_n\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\bar{\mathfrak{g}}^{(1)}_{l,n}+\sum_{l,l''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathfrak{u})\bar{\mathfrak{g}}^{(2)}_{l,n;l'',m}\right)}.\end{aligned} \tag{2.4.61}$$

$$\begin{aligned}\frac{\partial \Xi_{lN+j}(\mathbb{U})}{\partial e} = \partial_e\mathbb{D}^{(0)}(r_{l,j},e) + \sum_{l'}\sum_{n}\mathfrak{F}^{(\nu_u)}_{l',n}(\mathfrak{u})\partial_e\mathbb{D}^{(1)}_{l',n}(r_{l,j},e) + \\ +\sum_{l',l''}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l',n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathfrak{u})\partial_e\mathbb{D}^{(2)}_{l',n;l'',m}(r_{l,j},e)\end{aligned} \tag{2.4.62}$$

$$\begin{aligned}\frac{\partial \Xi_{NJ+1}(\mathbb{U})}{\partial e} = -1+ \\ +2\pi\rho\frac{\partial_e C(e) - \partial_e\mathfrak{g}^{(0)}(e) - \sum_l\sum_n\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\partial_e\mathfrak{g}^{(1)}_{l,n}(e) - \sum_{l,l'}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l',m}(\mathfrak{u})\partial_e\mathfrak{g}^{(2)}_{l,n;l',m}(e)}{1-\frac{8}{3}\pi\rho+\rho^2\left(\bar{\mathfrak{g}}^{(0)}+\sum_l\sum_n\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\bar{\mathfrak{g}}^{(1)}_{l,n}+\sum_{l,l'}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l',m}(\mathfrak{u})\bar{\mathfrak{g}}^{(2)}_{l,n;l',m}\right)}.\end{aligned} \tag{2.4.63}$$

To compute $\partial_e\mathbb{D}^{(i)}$ and $\partial_e\mathfrak{g}^{(i)}$, we use $\partial_e = \frac{1}{2\sqrt{e}}\frac{\partial}{\partial\sqrt{e}}$ and

$$\frac{\partial C(e)}{\partial\sqrt{e}} = 2 \tag{2.4.64}$$

$$\frac{\partial D^{(0)}(r,e)}{\partial\sqrt{e}} = -\frac{2r}{r+1}e^{-2\sqrt{e}r}$$
$$+\frac{\rho}{r+1}\int_0^{\min(1,r)} ds\ (s+1)B^{(0)}(s)e^{-2\sqrt{e}r}\left((1-2\sqrt{e}r)\sinh(2\sqrt{e}s)+2\sqrt{e}s\cosh(2\sqrt{e}s)\right)$$
$$+\mathbb{1}_{r\leqslant 1}\frac{\rho}{2(r+1)}\int_0^{1-r} d\sigma\ (\sigma+r+1)B^{(0)}(\sigma+r)\cdot$$
$$\cdot\left((1-2\sqrt{e}\sigma)\left(1-e^{-4\sqrt{e}r}\right)e^{-2\sqrt{e}\sigma}+4\sqrt{e}re^{-2\sqrt{e}(2r+\sigma)}\right) \tag{2.4.65}$$

$$\frac{\partial D^{(1)}_{l,n}(r,e)}{\partial\sqrt{e}} = \frac{\rho}{r+1}\int_0^{r} ds\ (s+1)B^{(1)}_{l,n}(s)e^{-2\sqrt{e}r}\left((1-2\sqrt{e}r)\sinh(2\sqrt{e}s)+2\sqrt{e}s\cosh(2\sqrt{e}s)\right)$$
$$+\frac{\rho}{2(r+1)}\int_0^{\infty} d\sigma\ (\sigma+r+1)B^{(1)}_{l,n}(\sigma+r)\cdot$$
$$\cdot\left((1-2\sqrt{e}\sigma)\left(1-e^{-4\sqrt{e}r}\right)e^{-2\sqrt{e}\sigma}+4\sqrt{e}re^{-2\sqrt{e}(2r+\sigma)}\right) \tag{2.4.66}$$

$$\frac{\partial D^{(2)}_{l,n;l',m}(r,e)}{\partial\sqrt{e}}$$
$$= \frac{\rho}{r+1}\int_0^{r} ds\ (s+1)B^{(2)}_{l,n;l',m}(s)e^{-2\sqrt{e}r}\left((1-2\sqrt{e}r)\sinh(2\sqrt{e}s)+2\sqrt{e}s\cosh(2\sqrt{e}s)\right)$$
$$+\frac{\rho}{2(r+1)}\int_0^{\infty} d\sigma\ (\sigma+r+1)B^{(2)}_{l,n;l',m}(\sigma+r)\cdot$$
$$\cdot\left((1-2\sqrt{e}\sigma)\left(1-e^{-4\sqrt{e}r}\right)e^{-2\sqrt{e}\sigma}+4\sqrt{e}re^{-2\sqrt{e}(2r+\sigma)}\right). \tag{2.4.67}$$

Furthermore,

$$\frac{\partial\gamma^{(0)}(e)}{\partial\sqrt{e}} = 4\rho\sqrt{e}\int_0^1 d\sigma\ (\sigma+1)B^{(0)}(\sigma)(1-\sqrt{e}\sigma)e^{-2\sqrt{e}\sigma} \tag{2.4.68}$$

$$\frac{\partial\gamma^{(1)}_{l,n}(e)}{\partial\sqrt{e}} = 4\rho e\int_0^{\infty} d\sigma\ (\sigma+1)B^{(1)}_{l,n}(\sigma)(1-\sqrt{e}\sigma)e^{-2\sqrt{e}\sigma} \tag{2.4.69}$$

and

$$\frac{\partial\gamma^{(2)}_{l,n;l',m}(e)}{\partial\sqrt{e}} = 4\rho e\int_0^{\infty} d\sigma\ (\sigma+1)B^{(2)}_{l,n;l',m}(\sigma)(1-\sqrt{e}\sigma)e^{-2\sqrt{e}\sigma}. \tag{2.4.70}$$

Finally, to get from D to $\mathbb{D}$ and γ to $\mathfrak{g}$, we approximate the integrate using Gauss-Legendre and Gauss-Laguerre quadratures (see appendix A2), as described above.

4 - We iterate the Newton algorithm until the Newton relative error ϵ becomes smaller than the `tolerance` parameter. The Newton error is defined as

$$\epsilon = \frac{\|\mathbb{U}^{(n+1)}-\mathbb{U}^{(n)}\|_2}{\|\mathbb{U}^{(n)}\|_2} \tag{2.4.71}$$

where $\|\cdot\|_2$ is the l_2 norm. The energy thus obtained is

$$e = \mathbb{U}_{JN+1}. \tag{2.4.72}$$

2.4.2.7. Condensate fraction

To compute the condensate fraction, we use the parameter μ in (2.4.3). The uncondensed fraction is

$$\eta = \partial_\mu e|_{\mu=0}. \tag{2.4.73}$$

To compute $\partial_\mu e$, we use

$$\Xi(\mathbb{U}) = 0 \tag{2.4.74}$$

which we differentiate with respect to μ:

$$\partial_\mu \mathbb{U} = -(D\Xi)^{-1}\partial_\mu \Xi. \tag{2.4.75}$$

We are left with computing $\partial_\mu \Xi$:

$$\begin{aligned}\frac{\partial \Xi_{lN+j}(\mathbb{U})}{\partial \mu} = \partial_\mu \mathbb{D}^{(0)}(r_{l,j}, e, \epsilon) + \sum_{l'}\sum_{n} \mathfrak{F}^{(\nu_u)}_{l',n}(\mathfrak{u})\partial_\mu \mathbb{D}^{(1)}_{l',n}(r_{l,j}, e, \epsilon) + \\ + \sum_{l',l''}\sum_{n,m} \mathfrak{F}^{(\nu_u)}_{l',n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l'',m}(\mathfrak{u})\partial_\mu \mathbb{D}^{(2)}_{l',n;l'',m}(r_{l,j}, e, \epsilon)\end{aligned} \tag{2.4.76}$$

$$\begin{aligned}&\frac{\partial \Xi_{NJ+1}(\mathbb{U})}{\partial \mu} = \\ &= 2\pi\rho \frac{\partial_\mu C(\epsilon) - \partial_\mu \mathfrak{g}^{(0)}(e,\epsilon) - \sum_l \sum_n \mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\partial_\mu \mathfrak{g}^{(1)}_{l,n}(e,\epsilon) - \sum_{l,l'}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l',m}(\mathfrak{u})\partial_\mu \mathfrak{g}^{(2)}_{l,n;l',m}(e,\epsilon)}{1 - \frac{8}{3}\pi\rho + \rho^2\left(\bar{\mathfrak{g}}^{(0)} + \sum_l\sum_n \mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\bar{\mathfrak{g}}^{(1)}_{l,n} + \sum_{l,l'}\sum_{n,m}\mathfrak{F}^{(\nu_u)}_{l,n}(\mathfrak{u})\mathfrak{F}^{(\nu_u)}_{l',m}(\mathfrak{u})\bar{\mathfrak{g}}^{(2)}_{l,n;l',m}\right)}.\end{aligned} \tag{2.4.77}$$

We then use $\partial_\mu = \frac{1}{4\sqrt{\epsilon}}\partial_{\sqrt{\epsilon}}$ and

$$\left.\frac{\partial C(\epsilon)}{\partial\sqrt{\epsilon}}\right|_{\mu=0} = \frac{8}{3}\sqrt{\epsilon} + 2 \tag{2.4.78}$$

$$\begin{aligned}&\left.\frac{\partial D^{(0)}(r)}{\partial\sqrt{\epsilon}}\right|_{\mu=0} = -\frac{2r}{r+1}e^{-2\sqrt{e}r} \\ &\quad + \frac{\rho}{(r+1)}\int_0^{\min(1,r)} ds\ (s+1)B^{(0)}(s)e^{-2\sqrt{e}r}((-1-2\sqrt{e}r)\sinh(2\sqrt{e}s) + 2\sqrt{e}s\cosh(2\sqrt{e}s)) \\ &+\mathbb{1}_{r\leqslant 1}\frac{\rho}{2(r+1)}\int_0^{1-r} d\sigma\ (\sigma+r+1)B^{(0)}(\sigma+r)\cdot \\ &\qquad \cdot\left((-1-2\sqrt{e}\sigma)\left(1-e^{-4\sqrt{e}r}\right)e^{-2\sqrt{e}\sigma} + 4\sqrt{e}re^{-2\sqrt{e}(2r+\sigma)}\right)\end{aligned} \tag{2.4.79}$$

$$\begin{aligned}&\left.\frac{\partial D^{(1)}_{l,n}(r)}{\partial\sqrt{\epsilon}}\right|_{\mu=0} = \frac{\rho}{r+1}\int_0^r ds\ (s+1)B^{(1)}_{l,n}(s)e^{-2\sqrt{e}r}\left((-1-2\sqrt{e}r)\sinh(2\sqrt{e}s) + 2\sqrt{e}s\cosh(2\sqrt{e}s)\right) \\ &+\frac{\rho}{2(r+1)}\int_0^{\infty} d\sigma\ (\sigma+r+1)B^{(1)}_{l,n}(\sigma+r)\cdot \\ &\qquad \cdot\left((-1-2\sqrt{e}\sigma)\left(1-e^{-4\sqrt{e}r}\right)e^{-2\sqrt{e}\sigma} + 4\sqrt{e}re^{-2\sqrt{e}(2r+\sigma)}\right)\end{aligned} \tag{2.4.80}$$

$$\left.\frac{\partial D^{(2)}_{l,n;l',m}(r)}{\partial\sqrt{\epsilon}}\right|_{\mu=0}$$
$$= \frac{\rho}{r+1}\int_0^r ds\ (s+1)B^{(2)}_{l,n;l',m}(s)e^{-2\sqrt{e}r}\left((-1-2\sqrt{e}r)\sinh(2\sqrt{e}s)+2\sqrt{e}s\cosh(2\sqrt{e}s)\right) \tag{2.4.81}$$
$$+\frac{\rho}{2(r+1)}\int_0^\infty d\sigma\ (\sigma+r+1)B^{(2)}_{l,n;l',m}(\sigma+r)\cdot$$
$$\cdot\left((-1-2\sqrt{e}\sigma)\left(1-e^{-4\sqrt{e}r}\right)e^{-2\sqrt{e}\sigma}+4\sqrt{e}re^{-2\sqrt{e}(2r+\sigma)}\right).$$

$$\left.\frac{\partial\gamma^{(0)}}{\partial\sqrt{\epsilon}}\right|_{\mu=0} = -4\rho e\int_0^1 d\sigma\ \sigma(\sigma+1)B^{(0)}(\sigma)e^{-2\sqrt{e}\sigma} \tag{2.4.82}$$

$$\left.\frac{\partial\gamma^{(1)}_{l,n}}{\partial\sqrt{\epsilon}}\right|_{\mu=0} = -4\rho e\int_0^\infty d\sigma\ (\sigma+1)\sigma B^{(1)}_{l,n}(\sigma)e^{-2\sqrt{e}\sigma} \tag{2.4.83}$$

and

$$\left.\frac{\partial\gamma^{(2)}_{l,n;l',m}}{\partial\sqrt{\epsilon}}\right|_{\mu=0} = -4\rho e\int_0^\infty d\sigma\ (\sigma+1)\sigma B^{(2)}_{l,n;l',m}(\sigma)e^{-2\sqrt{e}\sigma}. \tag{2.4.84}$$

2.5. `simpleq-iteration`

This method is used to solve the Simple equation using the iteration described in [CJL20]. The Simple equation is

$$-\Delta u = S - 4eu + 2e\rho u * u \tag{2.5.1}$$

$$S := (1-u)v, \quad \rho := \frac{2e}{\int dx\ (1-u(|x|))v(|x|)}. \tag{2.5.2}$$

for a soft potential v at fixed energy $e > 0$. The iteration is defined as

$$-\Delta u_n = S_n - 4eu_n + 2e\rho_{n-1}u_{n-1} * u_{n-1}, \quad u_0 = 0, \quad \rho_{n-1} = \frac{2e}{\int dx\ (1-u_{n-1}(|x|))v(|x|)}. \tag{2.5.3}$$

2.5.1. Usage

Unless otherwise noted, this method takes the following parameters (specified via the `[-p params]` flag, as a '`;`' separated list of entries).

- `e` (`Float64`, default: 10^{-4}): energy e.
- `maxiter` (`Int64`, default: 21): maximal number of iterations.
- `order` (`Int64`, default: 100): order used for all Gauss quadratures (denoted by N below).

The available `commands` are the following.

- `rho_e`: compute the density ρ as a function of e.
  Disabled parameters: `e`.
  Extra parameters:

- `minle` (`Float64`, default: 10^{-6}): minimal value for $\log_{10} e$.
- `maxle` (`Float64`, default: 10^{2}): maximal value for $\log_{10} e$.
- `nle` (`Int64`, default: 100): number of values for e (spaced logarithmically).
- `es` (`Array{Float64}`, default: $(10^{\texttt{minle}+\frac{\texttt{maxle}-\texttt{minle}}{\texttt{nle}}n})_n$: list of values for e, specified as a '`,`' separated list.

Output (one line for each value of e): $[e]$ $[\rho]$.

- `ux`: compute u as a function of $|x|$.
  Extra parameters:
  - `xmin` (`Float64`, default: 0): minimum of the range of $|x|$ to be printed.
  - `xmax` (`Float64`, default: 100): maximum of the range of $|x|$ to be printed.
  - `nx` (`Int64`, default: 100): number of points to print (linearly spaced).

  Output (one line for each value of x): $[|x|]$ $[u_1(|x|)]$ $[u_2(|x|)]$ $[u_3(|x|)]$ $\cdots$

2.5.2. Description

In Fourier space

$$\hat{u}_n(|k|) := \int dx\ e^{ikx} u_n(|x|) \tag{2.5.4}$$

(2.5.3) becomes

$$(k^2 + 4e)\hat{u}_n(|k|) = \hat{S}_n(|k|) + 2e\rho_{n-1}\hat{u}_{n-1}(k)^2, \quad \rho_n := \frac{2e}{\hat{S}_n(0)} \tag{2.5.5}$$

with

$$\hat{S}_n(|k|) = \hat{v}(|k|) - \frac{1}{(2\pi)^3}\int dp\ \hat{u}_n(|p|)\hat{v}(|k-p|). \tag{2.5.6}$$

We write $\hat{S}_n$ in bipolar coordinates (see lemma A3.1):

$$\hat{S}_n(|k|) = \hat{v}(|k|) - \frac{1}{(2\pi)^3}\int_0^\infty dt\ t\hat{u}_n(t)H(|k|,t) \tag{2.5.7}$$

with

$$H(y,t) := \frac{2\pi}{y}\int_{|y-t|}^{y+t} ds\ s\hat{v}(s) \tag{2.5.8}$$

(note that this agrees with the function (2.1.12) defined for `easyeq`). We also change variables to

$$y = \frac{1}{t+1}, \quad t = \frac{1-y}{y} \tag{2.5.9}$$

$$\hat{S}_n(|k|) = \hat{v}(|k|) - \frac{1}{(2\pi)^3}\int_0^1 dy\ \frac{(1-y)\hat{u}_n(\frac{1-y}{y})H(|k|,\frac{1-y}{y})}{y^3}. \tag{2.5.10}$$

We approximate this integral using a Gauss-Legendre quadrature (see appendix A2) and discretize Fourier space:

$$k_i := \frac{1-x_i}{1+x_i}, \quad y_i := \frac{x_i+1}{2} \tag{2.5.11}$$

where x_i are the Gauss-Legendre abscissa, and

$$\hat{S}_n(k_i) \approx \hat{v}(k_i) - \frac{1}{2(2\pi)^3}\sum_{j=1}^{N} w_j \frac{(1-y_j)\hat{u}_n(\frac{1-y_j}{y_j})H(k_i,\frac{1-y_j}{y_j})}{y_j^3} \tag{2.5.12}$$

so if

$$\mathbb{U}_i(n) := \hat{u}_n(k_i), \quad \mathbf{v}_i := \hat{v}(k_i) \tag{2.5.13}$$

we have

$$\sum_{j=1}^{N} A_{i,j}\mathbb{U}_j(n) = b_i^{(n)} \tag{2.5.14}$$

with

$$A_{i,j} := (k_i^2 + 4e)\delta_{i,j} + \frac{w_j(1-y_j)H(k_i, \frac{1-y_j}{y_j})}{2(2\pi)^3 y_j^3} \tag{2.5.15}$$

and

$$b_i^{(n)} := \mathbf{v}_i + 2e\rho_{n-1}\mathbb{U}_i(n-1)^2 \tag{2.5.16}$$

in terms of which

$$\mathbb{U} = A^{-1}b^{(n)}. \tag{2.5.17}$$

Finally, we compute ρ_n using the second of (2.5.5):

$$\rho_n = \frac{2e}{\hat{S}_n(0)}, \quad S_n(0) \approx \hat{v}(0) - \frac{1}{2(2\pi)^3}\sum_{j=1}^{N} w_j \frac{(1-y_j)\hat{u}_n(\frac{1-y_j}{y_j})H(0, \frac{1-y_j}{y_j})}{y_j^3}. \tag{2.5.18}$$

3. Potentials

In this section, we describe the potentials available in `simplesolv`, and provide documentation to add custom potentials to `simplesolv`.

3.1. Built-in potentials

3.1.1. `exp`

In x-space,

$$v(|x|) = ae^{-|x|}. \tag{3.1.1}$$

The constant a is specified through the `v_a` parameter, and can be any real number. Note that $v \geqslant 0$ if and only if $a \geqslant 0$. This is the potential that is selected by default.

1 - In Fourier space,

$$\hat{v}(|k|) = \int dx\ e^{ikx}v(|x|) = \frac{8\pi a}{(1+k^2)^2}. \tag{3.1.2}$$

In particular, v is of positive type (that is, $\hat{v} \geqslant 0$) if and only if $a \geqslant 0$.

2 - The zero energy scattering solution, that is, the solution of

$$(-\Delta + v)\psi = 0, \quad \lim_{|x|\to\infty} \psi = 1 \tag{3.1.3}$$

is

$$\psi(r) = \frac{cI_0(2\sqrt{a}e^{-\frac{r}{2}}) + 2K_0(2\sqrt{a}e^{-\frac{r}{2}})}{r}, \quad c = -\frac{2K_0(2\sqrt{a})}{I_0(2\sqrt{a})} \tag{3.1.4}$$

where I_0 and K_0 are the modified Bessel functions, and the square root is taken with a branch cut on the negative imaginary axis. In other words, if $a < 0$, $\sqrt{a} = i\sqrt{|a|}$ and [DLMF, (10.27.6), (10.27.9), (10.27.10)]

$$I_0(ix) = J_0(x), \quad K_0(ix) = -\frac{\pi}{2}(Y_0(x) + iJ_0(x)) \tag{3.1.5}$$

where J_0 and Y_0 are the Bessel functions, so

$$\psi(r) = \pi \frac{c' J_0(2\sqrt{|a|}e^{-\frac{r}{2}}) - Y_0(2\sqrt{|a|}e^{-\frac{r}{2}})}{r}, \quad c' = \frac{Y_0(2\sqrt{|a|})}{J_0(2\sqrt{|a|})}. \tag{3.1.6}$$

The scattering length is [DLMF, (10.25.2), (10.31.2)]

$$\mathfrak{a}_0 = \lim_{r\to\infty} r(1-\psi(r)) = \log(a) + 2\gamma + \frac{2K_0(2\sqrt{a})}{I_0(2\sqrt{a})} \tag{3.1.7}$$

where γ is the Euler constant, which, for $a < 0$, is

$$\mathfrak{a}_0 = \log|a| + 2\gamma - \frac{\pi Y_0(2\sqrt{|a|})}{J_0(2\sqrt{|a|})}. \tag{3.1.8}$$

3.1.2. `tent`

In x-space,

$$v(|x|) = \mathbb{1}_{|x|<b} a \frac{2\pi}{3} \left(1 - \frac{|x|}{b}\right)^2 \left(\frac{|x|}{b} + 2\right). \tag{3.1.9}$$

The constants a and b are specified through the `v_a` and `v_b` parameters, and $a \in \mathbb{R}$, $b > 0$. Note that $v \geqslant 0$ if and only if $a \geqslant 0$. Note that

$$v(|x|) = \frac{a}{b^3} \mathbb{1}_{|x|<\frac{b}{2}} * \mathbb{1}_{|x|<\frac{b}{2}} \tag{3.1.10}$$

(this is more easily checked in Fourier space than explicitly computing the convolution).

In Fourier space,

$$\hat{v}(|k|) = \int dx\ e^{ikx} v(|x|) = a\frac{b^3}{8} \left(4\pi \frac{\sin(\frac{|k|b}{2}) - \frac{|k|b}{2}\cos(\frac{|k|b}{2})}{\frac{(|k|b)^3}{8}}\right)^2. \tag{3.1.11}$$

Note that this is $\frac{a}{b^3}(\hat{\mathbb{1}}_{|x|<\frac{b}{2}})^2$. In particular, v is of positive type (that is, $\hat{v} \geqslant 0$) if and only if $a \geqslant 0$.

3.1.3. `expcry`

In x-space

$$v(|x|) = e^{-|x|} - ae^{-b|x|}. \tag{3.1.12}$$

The constants a and b are specified through the `v_a` and `v_b` parameters, and $a \in \mathbb{R}$, $b > 0$. Note that $v \geqslant 0$ if and only if $a \leqslant 1$ and $b \geqslant 1$.

In Fourier space

$$\hat{v}(|k|) = \int dx\ e^{ikx} v(|x|) = 8\pi \left(\frac{1}{(1+k^2)^2} - \frac{ab}{(b^2+k^2)^2}\right) \tag{3.1.13}$$

In particular, $\hat{v}$ is of positive type (that is, $\hat{v} \geqslant 0$) if and only if $ab \leqslant 1$, $a \leqslant b$ and $a \leqslant b^3$. If $a \leqslant 1$, $b \geqslant 1$ and $ab > 1$, then $\hat{v}$ has a unique minimum at

$$|k_*| = \sqrt{\frac{b^2 - (ab)^{\frac{1}{3}}}{(ab)^{\frac{1}{3}} - 1}}, \quad \hat{v}(|k_*|) = -8\pi \frac{((ab)^{\frac{1}{3}} - 1)^3}{b^2 - 1}. \tag{3.1.14}$$

3.1.4. `npt`

In x-space

$$v(|x|) = x^2 e^{-|x|}. \tag{3.1.15}$$

Note that $v \geqslant 0$.

In Fourier space

$$\hat{v}(|k|) = \int dx\ e^{ikx} v(|x|) = 96\pi \frac{1-k^2}{(1+k^2)^4}. \tag{3.1.16}$$

In particular, v is not of positive type (that is, $\hat{v}$ is not $\geqslant 0$).

3.1.5. `alg`

In x-space

$$v(|x|) = \frac{1}{1+\frac{1}{4}|x|^4}. \tag{3.1.17}$$

Note that $v \geqslant 0$.

In Fourier space

$$\hat{v}(|k|) = \int dx\ e^{ikx} v(|x|) = 4\pi^2 e^{-|k|} \frac{\sin(|k|)}{|k|}. \tag{3.1.18}$$

In particular, v is not of positive type (that is, $\hat{v}$ is not $\geqslant 0$).

3.1.6. `algwell`

In x-space

$$v(|x|) = \frac{1+a|x|^4}{(1+|x|^2)^4}. \tag{3.1.19}$$

The constant a can be set using the parameter `v_a` and can be any real number. Note that $v \geqslant 0$ if and only if $a \geqslant 0$. If $a > 8$, this potential has a local minimum at $|x_-|$ and a local maximum at $|x_+|$:

$$|x_\pm| = \sqrt{\frac{1}{2}(1 \pm \sqrt{1-8a^{-1}})}. \tag{3.1.20}$$

In Fourier space

$$\hat{v}(|k|) = \int dx\ e^{ikx} v(|x|) = \frac{\pi^2}{24} e^{-|k|}(a(k^2 - 9|k| + 15) + k^2 + 3|k| + 3). \tag{3.1.21}$$

In particular, v is of positive type (that is, $\hat{v}$ is not $\geqslant 0$) if and only if

$$-\frac{1}{5} \leqslant a \leqslant 3 + \frac{8}{7}\sqrt{7} \approx 6.02. \tag{3.1.22}$$

3.1.7. `exact`

In x-space

$$v(|x|) = \frac{12c(x^6b^6(2e-b^2)+b^4x^4(9e-7b^2)+4b^2x^2(3e-2b^2)+(5e+16b^2))}{(1+b^2x^2)^2(4+b^2x^2)^2((1+b^2x^2)^2-c)} \tag{3.1.23}$$

The constants a, b, c, e can be set using the parameters `v_a`, `v_b`, `v_c`, `v_e`, and $a, e \in \mathbb{R}$, $b \neq 0$, $c > 0$, $c \neq 9$. Note that $v \geqslant 0$ if and only if [CJL21]

$$b > 0, \quad 0 < c < 1, \quad \frac{e}{b^2} \geqslant \frac{-263+23\sqrt{161}}{48} \approx 0.601. \tag{3.1.24}$$

With this potential, the Simple equation has an exact solution:

$$u = \frac{c}{(1+b^2x^2)^2}, \quad \rho = \frac{b^3}{c\pi^2}. \tag{3.1.25}$$

In Fourier space $\hat{v}(|k|) = \int dx\ e^{ikx}v(x)$ is

$$\begin{aligned}\hat{v}(|k|) = \frac{48c\pi^2}{|k|}\Bigg(&\frac{18+3\sqrt{c}-(4-3\alpha)c-(1-2\alpha)c^{\frac{3}{2}}}{4(3+\sqrt{c})^2c^{\frac{3}{2}}}e^{-\sqrt{1-\sqrt{c}}\frac{|k|}{b}} \\ &+\frac{-18+3\sqrt{c}+(4-3\alpha)c-(1-2\alpha)c^{\frac{3}{2}}}{4(3-\sqrt{c})^2c^{\frac{3}{2}}}e^{-\sqrt{1+\sqrt{c}}\frac{|k|}{b}}+\frac{1-\frac{|k|}{b}}{2c}e^{-\frac{|k|}{b}}-\frac{\alpha(3(9-c)\frac{|k|}{b}+8c)}{8(9-c)^2}e^{-2\frac{|k|}{b}}\Bigg)\end{aligned} \tag{3.1.26}$$

with $\alpha := \frac{e}{b^2}$.

3.2. Programming custom potentials

In this section, we provide documentation for programming custom potentials.

The potentials are implemented in the file '`$SIMPLESOLV/src/potentials.jl`', and consist of two functions, one specifying the potential in Fourier space (the "potential function"), and the other returning an approximate value for the scattering length (the "scatterin glength function") (as is explained below, a precise value of the scattering length is not actually needed). For instance, the potential `exp` has two functions: `v_exp` and `a0_exp`. The potential function should take the following arguments:

- `k` (`Float64`): the Fourier momentum
- and any parameters that the potential depends on, such as a in `exp` (can be of any type, provided the appropriate changes are made to `main.jl` as explained below)

and it must return a `Float64`: the value of $\hat{v}$ at `k`. The scattering length function takes the same parameters as an input, and returns a `Float64`: the approximate value for the scattering length.

In addition, the potential must be linked in '`$SIMPLESOLV/src/main.jl`'. In that file, the potential is read from the command line option `U`. The relevant code is in lines 197-222. To add a new potential, add

```
elseif potential=="{name of potential}"
    v=k->{potential function}(k,v_param_a,v_param_b,...)
```

`a0=`{scattering length function}`(v_param_a,v_param_b,...)`

where the number of `v_param` entries should be the number of parameters of the potential. The parameters that are currently read from the parameters list are `a`, `b`, `c` and `e`. To add a parameter, it must first be declared and initialized after line 35, and code to read it should be added after line 172:

```
elseif lhs="v_{name of parameter}"
    v_param_{name of parameter}=parse(Float64,rhs)
```

If the new parameter has a type other than `Float64`, this should be changed in the `parse` function, and in the initialization.

The approximation of the scattering length is only used to initialize the Newton algorithm for `easyeq`, so it is not important that it be exact. In fact, some of the built-in potentials set the scattering length to 1, when it has proved too difficult to compute it exactly.

Appendices

A1. Chebyshev polynomial expansion

In this appendix, we compute the error of Chebyshev polynomial expansions of regular functions. Specifically, we will consider class-s Gevrey functions (which is a generalization of the notion of analyticity: analytic functions are Gevrey functions with $s = 1$). A class-s Gevrey function on $[-1, 1]$ is a $\mathcal{C}^\infty$ function that satisfies, $\forall n \in \mathbb{N}$,

$$\sup_{x\in[-1,1]} \left| \frac{d^n f(x)}{dx^n} \right| \leqslant C_0 C^n (n!)^s. \tag{A1.0.1}$$

Formally, the Chebyshev polynomial expansion of f is

$$f(x) = \frac{c_0}{2} + \sum_{j=1}^{\infty} c_j T_j(x) \tag{A1.0.2}$$

where T_j is the j-th Chebyshev polynomial:

$$T_j(x) := \cos(j \arccos(x)) \tag{A1.0.3}$$

and

$$c_j := \frac{2}{\pi} \int_0^\pi d\theta \; f(\cos\theta) \cos(j\theta). \tag{A1.0.4}$$

Lemma A1.1

Let f be a class-s Gevrey function on $[-1, 1]$ with $s \in \mathbb{N} \setminus \{0\}$. There exist $b_0, b > 0$, which are independent of s, such that the coefficients c_j of the Chebyshev polynomial expansion are bounded as

$$c_j \leqslant b_0 e^{-bj^{\frac{1}{s}}}. \tag{A1.0.5}$$

In particular, the Chebyshev polynomial expansion is absolutely convergent pointwise (and, therefore in any L_p norm), and, for every $N \geqslant 1$,

$$\left| f(x) - \frac{c_0}{2} - \sum_{j=1}^{N} c_j T_j(x) \right| \leqslant \frac{b_0}{1 - e^{-b}} (s-1)! (N^{\frac{1}{s}} + b^{-1})^{s-1} e^{-bN^{\frac{1}{s}}}. \tag{A1.0.6}$$

Proof: Note that (A1.0.2) is nothing other than the Fourier cosine series expansion of $F(\theta) := f(\cos(\theta))$, which is an even, periodic, class-s Gevrey function on $[-\pi, \pi]$, whose j-th Fourier coefficient for $j \in \mathbb{Z}$ is equal to $\frac{1}{2} c_{|j|}$. The bound (A1.0.5) follows from a well-known estimate of the decay of Fourier coefficients of class-s Gevrey functions (see e.g. [Ta87, Theorem 3.3]). The bound (A1.0.6) then follows from $|T_j(x)| \leqslant 1$ and lemma A1.2 below. □

Lemma A1.2

Given $b > 0$ and two integers $N, s > 0$,

$$\sum_{j=N}^{\infty} e^{-bj^{\frac{1}{s}}} \leqslant (s-1)! \left(N^{\frac{1}{s}} + \frac{1}{1 - e^{-b}} \right)^{s-1} \frac{e^{-bN^{\frac{1}{s}}}}{1 - e^{-b}} \tag{A1.0.7}$$

Proof: If $\nu_{N,s}^s := \lfloor N^{\frac{1}{s}} \rfloor^s$ denotes the largest integer that is $\leqslant N$ and has an integer s-th root, then

$$\sum_{j=N}^{\infty} e^{-bj^{\frac{1}{s}}} \leqslant \sum_{j=\nu_{N,s}^s}^{\infty} e^{-bj^{\frac{1}{s}}} \leqslant \sum_{k=\nu_{N,s}}^{\infty} (k^s - (k-1)^s) e^{-bk} \leqslant s \sum_{k=\nu_{N,s}}^{\infty} k^{s-1} e^{-bk}. \tag{A1.0.8}$$

We then estimate

$$\sum_{k=\nu_{N,s}}^{\infty} k^{s-1} e^{-bk} = \frac{d^{s-1}}{d(-b)^{s-1}} \sum_{k=\nu_{N,s}}^{\infty} e^{-bk} \leqslant (s-1)! \left(\nu_{N,s} + \frac{1}{1-e^{-b}} \right)^{s-1} \frac{e^{-b\nu_{N,s}}}{1-e^{-b}} \tag{A1.0.9}$$

which concludes the proof of the lemma. □

A2. Gauss quadratures

Gauss quadratures are approximation schemes to compute integrals of the form

$$\int_a^b dt\ \omega(t) f(t) \tag{A2.0.1}$$

where $\omega(x) \geqslant 0$ is one of several functions that Gauss quadratures can treat. The possible choices of ω and (a, b) are

- $(a, b) = (-1, 1)$, $\omega(t) = 1$: Gauss-Legendre quadratures.
- $(a, b) = (-1, 1)$, $\omega(t) = (1-t)^\alpha (1+t)^\beta$, $\alpha, \beta > -1$: Gauss-Jacobi quadratures.
- $(a, b) = (-1, 1)$, $\omega(t) = (1-t^2)^{-\frac{1}{2}}$: Gauss-Chebyshev quadratures.
- $(a, b) = (0, \infty)$, $\omega(t) = e^{-t}$: Gauss-Laguerre quadratures.
- $(a, b) = (0, \infty)$, $\omega(t) = t^\alpha e^{-t}$, $\alpha > -1$: generalized Gauss-Laguerre quadratures.
- $(a, b) = (0, \infty)$, $\omega(t) = e^{-t^2}$: Gauss-Hermite quadratures.

It is not our goal here to discuss Gauss quadratures in detail, or their relation to orthogonal polynomials. Instead, we will compute the error made when approximating such an integral by a Gauss quadrature.

For each Gauss quadrature, the integral is approximated in the form

$$\int_a^b dt\ \omega(t) f(t) \approx \sum_{i=1}^{N} w_i f(r_i) \tag{A2.0.2}$$

where w_i are called the *weights*, r_i are the *abcissa*, and N is the *order*. The weights and abcissa depend on both ω and the order N. The crucial property of Gauss quadratures is that they are *exact* when f is a polynomial of order $\leqslant 2N-1$.

In this appendix, we compute the error of Gauss quadratures when used to integrate regular functions. Specifically, we will consider class-s Gevrey functions (which is a generalization of the notion of analyticity: analytic functions are Gevrey functions with $s = 1$). A class-s Gevrey function on is a $\mathcal{C}^\infty$ function that satisfies, $\forall n \in \mathbb{N}$,

$$\sup_{x \in [-1,1]} \left| \frac{d^n f(x)}{dx^n} \right| \leqslant C_0 C^n (n!)^s. \tag{A2.0.3}$$

Lemma A2.1

Let f be a class-s Gevrey function with $s \in \mathbb{N} \setminus \{0\}$. There exist $b_0, b > 0$, which are independent of s, and $N_0 > 0$, which is independent of s and f, such that, if $N \geqslant N_0$, then, denoting the Gauss weights and abcissa by w_i and r_i,

$$\left| \int_a^b dt\ \omega(t) f(t) - \sum_{i=1}^{N} w_i f(r_i) \right| \leqslant b_0 (s-1)! ((2N-1)^{\frac{1}{s}} + b^{-1})^{s-1} e^{-b(2N-1)^{\frac{1}{s}}} \left(\int_a^b \omega(t) + \sum_{i=1}^{N} w_i \right). \tag{A2.0.4}$$

In particular, if f is analytic (i.e. $s = 1$), then

$$\left| \int_a^b dt\ \omega(t) f(t) - \sum_{i=1}^{N} w_i f(r_i) \right| \leqslant b_0 e^{-b(2N-1)} \left(\int_a^b \omega(t) + \sum_{i=1}^{N} w_i \right). \tag{A2.0.5}$$

Proof: We expand f into Chebyshev polynomials as in (A1.0.2):

$$f(x) = \frac{c_0}{2} + \sum_{j=1}^{\infty} c_j T_j(x) \tag{A2.0.6}$$

Let

$$g(x) := f(x) - \frac{c_0}{2} - \sum_{j=1}^{2N-1} c_j T_j(x). \tag{A2.0.7}$$

Since order-N Gauss quadratures are exact on polynomials of order $\leqslant 2N-1$, we have

$$\int_a^b dt\ \omega(t) f(t) - \sum_{i=1}^{N} w_i f(r_i) = \int_a^b dt\ \omega(t) g(t) - \sum_{i=1}^{N} w_i g(r_i) \tag{A2.0.8}$$

and, by lemma A1.1,

$$|g(x)| \leqslant (\text{const.})(s-1)! ((2N-1)^{\frac{1}{s}} + b^{-1})^{s-1} e^{-b(2N-1)^{\frac{1}{s}}}. \tag{A2.0.9}$$

□

A3. Bipolar coordinates

Bipolar coordinates are very useful for computing convolutions of radial functions in three dimensions.

Lemma A3.1

For $y \in \mathbb{R}^3$,

$$\int_{\mathbb{R}^3} dx\ f(|x|, |x-y|) = \frac{2\pi}{|y|} \int_0^{\infty} dt \int_{||y|-t|}^{|y|+t} ds\ st f(s,t) \tag{A3.0.1}$$

Proof: Without loss of generality, we assume that $y = (0, 0, a)$ with $a > 0$. We first change to cylindrical coordinates: (ρ, θ, x_3):

$$\int_{\mathbb{R}^3} dx\ f(|x|, |x-y|) = 2\pi \int_0^{\infty} d\rho \int_{-\infty}^{\infty} dx_3\ \rho f(|(\rho, 0, x_3)|, |(\rho, 0, x_3 - a)|). \tag{A3.0.2}$$

Next, we change variables to

$$s = |(\rho, 0, x_3)|, \quad t = |(\rho, 0, x_3 - a)|. \tag{A3.0.3}$$

The inverse of this transformation is

$$x_3 = \pm\frac{a^2+s^2-t^2}{2a}, \quad \rho = \frac{1}{2a}\sqrt{4a^2s^2-(a^2+s^2-t^2)^2} \tag{A3.0.4}$$

and its Jacobian is

$$\frac{2ts}{\sqrt{-2a^4+2a^2(t^2+s^2)-(t^2-s^2)^2}} = \frac{ts}{\rho a}. \tag{A3.0.5}$$

□

The following is a generalization of the previous lemma to functions of four variables.

Lemma A3.2

For $y \in \mathbb{R}^3$,

$$\begin{aligned}&\int_{\mathbb{R}^6} dxdx'\ f(|x|,|x-y|,|x'|,|x'-y|,|x-x'|)\\ &\qquad = \frac{2\pi}{|y|^2}\int_0^\infty dt\int_{||y|-t|}^{|y|+t} ds\int_0^\infty dt'\int_{||y|-t'|}^{|y|+t'} ds'\int_0^{2\pi} d\theta\ sts't' f(s,t,s',t',\xi(s,t,s',t',\theta,|y|))\end{aligned} \tag{A3.0.6}$$

with

$$\begin{aligned}\xi(s,t,s',t',\theta,|y|) := \frac{1}{\sqrt{2}|y|}\Big(&|y|^2(s^2+t^2+(s')^2+(t')^2)-|y|^4-(s^2-t^2)((s')^2-(t')^2)\\ &-\sqrt{(4|y|^2s^2-(|y|^2+s^2-t^2)^2)(4|y|^2(s')^2-(|y|^2+(s')^2-(t')^2)^2)}\cos\theta\Big)^{\frac{1}{2}}.\end{aligned} \tag{A3.0.7}$$

Proof: Without loss of generality, we assume that $y=(0,0,a)$ with $a>0$. We first change to cylindrical coordinates: $(\rho,\theta,x_3;\rho',\theta',x_3')$:

$$\begin{aligned}&\int_{\mathbb{R}^6} dxdx'\ f(|x|,|x-y|,|x'|,|x'-y|,|x-x'|)\\ &= 2\pi\int_0^\infty d\rho\int_{-\infty}^\infty dx_3\int_0^\infty d\rho'\int_{-\infty}^\infty dx_3'\int_0^{2\pi} d\theta'\ \rho\rho' f(s,t,s',t',|(\rho-\rho'\cos\theta',\rho'\sin\theta',x_3-x_3')|).\end{aligned} \tag{A3.0.8}$$

where

$$s := |(\rho,0,x_3)|, \quad t := |(\rho,0,x_3-a)|, \quad s' := |(\rho'\cos\theta',\rho'\sin\theta',x_3')|, t' := |(\rho'\cos\theta',\rho'\sin\theta',x_3'-a)| \tag{A3.0.9}$$

Next, we change variables to (s,t,s',t',θ'). The Jacobian of this transformation is, by (A3.0.5),

$$\frac{tst's'}{\rho\rho' a^2}. \tag{A3.0.10}$$

Furthermore, by (A3.0.4),

$$x_3 - x_3' = \frac{s^2-t^2-(s')^2+(t')^2}{2a} \tag{A3.0.11}$$

and

$$\rho = \frac{\sqrt{4a^2s^2-(a^2+s^2-t^2)^2}}{2a}, \quad \rho' = \frac{\sqrt{4a^2(s')^2-(a^2+(s')^2-(t')^2)^2}}{2a} \tag{A3.0.12}$$

so

$$\begin{aligned}|(\rho-\rho'\cos\theta',\rho'\sin\theta',x_3-x_3')| = \frac{1}{2a}\Big(&4a^2s^2-(a^2+s^2-t^2)^2+4a^2(s')^2-(a^2+(s')^2-(t')^2)^2\\ &-2\sqrt{(4a^2s^2-(a^2+s^2-t^2)^2)(4a^2(s')^2-(a^2+(s')^2-(t')^2)^2)}\cos\theta'+(s^2-t^2-(s')^2+(t')^2)^2\Big)^{\frac{1}{2}}\end{aligned} \tag{A3.0.13}$$

and

$$|(\rho-\rho'\cos\theta',\rho'\sin\theta',x_3-x_3')| = \frac{1}{\sqrt{2}a}\Big(a^2(s^2+t^2+(s')^2+(t')^2)-a^4-(s^2-t^2)((s')^2-(t')^2) \\ -\sqrt{(4a^2s^2-(a^2+s^2-t^2)^2)(4a^2(s')^2-(a^2+(s')^2-(t')^2)^2)}\cos\theta'\Big)^{\frac{1}{2}}. \tag{A3.0.14}$$

□

A4. Hann windowing

In this appendix, we discuss the use of Hann windows to compute Fourier transforms. Consider the Fourier transform

$$\hat{f}(k) = \int dx\ e^{ikx} f(x). \tag{A4.0.1}$$

Evaluating this integral numerically can be tricky, especially at high $|k|$, because of the rapid oscillations at large $|x|$. A trick to palliate such a problem is to multiply f by a *window function* h_L, which cuts off distances of order L. We then compute, instead of $\hat{f}$,

$$\tilde{f}(k) = \int dx\ e^{ikx} h_L(x) f(x). \tag{A4.0.2}$$

We can then evaluate $\tilde{f}$ using standard numerical techniques, such as Gauss quadratures (see appendix A2), without issues at large $|x|$. However, in doing so, we will make an error in the Fourier transform. To quantify this error, note that

$$\tilde{f}(k) = \hat{h}_L \hat{*} \hat{f}(k) \equiv \int \frac{dq}{(2\pi)^d}\ \hat{h}_L(q)\hat{f}(k-q) \tag{A4.0.3}$$

so if we choose h_L in such a way that $\hat{h}_L$ is peaked around the origin, then $\tilde{f}$ will not differ too much from $\hat{f}$:

$$\hat{f}(k) - \tilde{f}(k) = ((2\pi)^d\delta(k) - \hat{h}_L)\hat{*}\hat{f}(k). \tag{A4.0.4}$$

The *Hann* window is defined as

$$h_L(x) = \cos^2(\tfrac{\pi|x|}{L})\mathbb{1}_{|x|<\frac{L}{2}} \tag{A4.0.5}$$

whose Fourier transform is, in $d=3$,

$$\hat{h}_L(k) = \frac{4\pi^3L^2}{|k|}\frac{((|k|L)^3-4|k|L\pi^2)\cos(\frac{|k|L}{2}) - 2(3(|k|L)^2-4\pi^2)\sin(\frac{|k|L}{2})}{((|k|L)^3-4|k|L\pi^2)^2} \tag{A4.0.6}$$

which decays at large $|k|L$ as

$$\hat{h}_L(k) \sim \frac{4\pi^3}{|k|^4L}\cos(\tfrac{|k|L}{2}). \tag{A4.0.7}$$

References

[CHe21] E.A. Carlen, M. Holzmann, I. Jauslin, E.H. Lieb - *Simplified approach to the repulsive Bose gas from low to high densities and its numerical accuracy*, Physical Review A, volume 103, issue 5, number 053309, 2021, doi:10.1103/PhysRevA.103.053309, arxiv:2011.10869.

[CJL20] E.A. Carlen, I. Jauslin, E.H. Lieb - *Analysis of a simple equation for the ground state energy of the Bose gas*, Pure and Applied Analysis, volume 2, issue 3, pages 659-684, 2020, doi:10.2140/paa.2020.2.659, arxiv:1912.04987.

[CJL21] E.A. Carlen, I. Jauslin, E.H. Lieb - *Analysis of a Simple Equation for the Ground State of the Bose Gas II: Monotonicity, Convexity, and Condensate Fraction*, SIAM Journal on Mathematical Analysis, volume 53, number 5, pages 5322-5360, 2021, doi:10.1137/20M1376820, arxiv:2010.13882.

[DLMF] F.W.J. Olver, A.B. Olde Daalhuis, D.W. Lozier, B.I. Schneider, R.F. Boisvert, C.W. Clark, B.R. Miller, B.V. Saunders, H.S. Cohl, M.A. McClain (editors) - *NIST Digital Library of Mathematical Functions*, Release 1.1.3 of 2021-09-15, 2021.

[Ja23] I. Jauslin - *The Simplified approach to the Bose gas without translation invariance*, 2023 http://ian.jauslin.org/publications/23j/.

[Ta87] Y. Taguchi - *Fourier coefficients of periodic functions of Gevrey classes and ultradistributions*, Yokohama Mathematical Journal, volume 35, pages 51-60, 1987.